Informatica® PowerCenter
10.5

Data Validation Option User Guide

Informatica PowerCenter Data Validation Option User Guide
10.5
May 2021

Publication Date: 2021-04-29

Table of Contents

Preface

Read the *PowerCenter Data Validation User Guide* to learn how you can test and validate data across multiple sources. Read about the Data Validation Option architecture, components, and the validation process. Learn how to configure and run validation tests.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction to Data Validation Option

This chapter includes the following topics:

Data Validation Option Overview

Data validation is the process of verifying the accuracy and completeness of data integration operations such as the migration or replication of data. Use Informatica Data Validation Option to create repeatable tests that validate PowerCenter mapping data.

Use Data Validation Option to test large volumes of data in heterogeneous environments. You can create a test that validates data in one or more data sources. You might create a test against one data source to verify the number of unique IDs in a table. You might create a test between two data sources to test referential integrity, such as verifying that every product in a fact table is also in the dimension table. You can also create views on one or more data sources to perform more complex tests.

Use the Data Validation Client to configure validation tests. When you create a test, you configure the conditions through an Expression Editor. After you run the test, you can view the testing summary in the client tool and you can view the result data in reports.

The following types of validation tests are common tests that you might perform on mapping data:

Regression testing

Validate data during development testing to ensure that updates to mapping logic do not introduce regressions. Perform the validation between the development target and the testing target.

Production validation

Validate data after you move from development to production. Perform the validation between the target systems to verify consistency.

Upgrade validation

Validate data after you upgrade the Informatica platform version. Perform the validation between the target systems to verify consistency.

Migration validation

Validate data after you migrate data. Perform the validation between the source and target to verify that the target contains the data and values expected from the integration process.

Important: A data validation test can validate data, but it cannot validate the logic that you use to transform or migrate data. A data validation test can identify inconsistencies in data, but it cannot identify the source of the inconsistencies.

Data Validation Guidelines

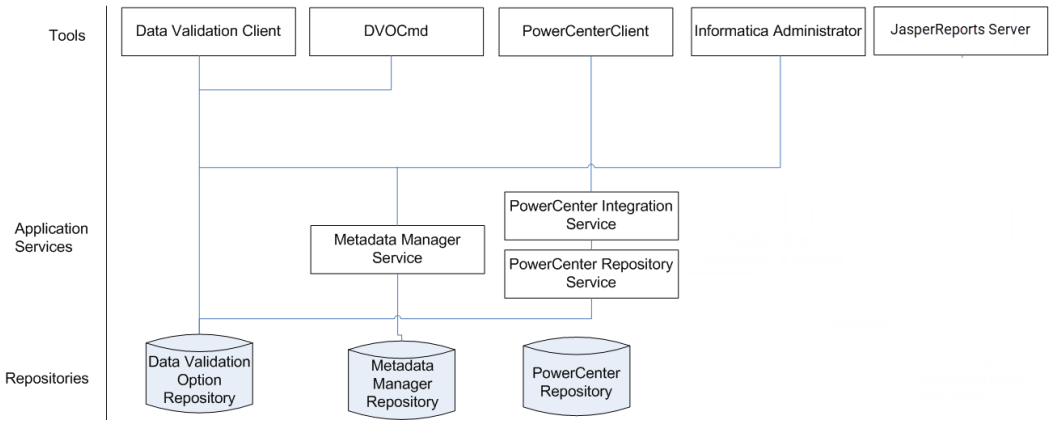
Consider the following guidelines when you work with Data Validation Option:

- Keep the mapping logic and the testing logic separate. Do not copy formulas from the PowerCenter mapping into the data validation test. If the mapping logic contains errors, the test result will replicate the mapping error. Remember that Data Validation Option validates data, but it does not validate the mapping logic.
- Perform simple tests first, such as counts and constraint checks, to identify obvious errors that might be easy to fix.
- Consider splitting complex SQL into more than one table pair object.
- You do not have to perform all testing with Data Validation Option. You might want to perform a particular step with SQL.

Data Validation Option Architecture

Data Validation Option components include client tools, application services, and repositories that Data Validation Option uses to validate mapping data and to report on validation results. The components involved depend on the tasks that you perform.

The following image shows the Data Validation Option architecture:



Data Validation Tools

Use the Data Validation tools to set up Data Validation Option and to configure and run validation tests.

Use the following tools to perform Data Validation Option administrator and user tasks:

Data Validation Client

Use the Data Validation Client to create views, table objects, and tests. Run the tests and view test results from within the Data Validation Client. You can also add PowerCenter repositories and repository folders to the Data Validation Client.

DVOcmd

Use the DVOcmd command line program to complete many of the same tasks that you perform in the Data Validation Client.

PowerCenter Client

Use the PowerCenter Client to create the target folder that stores the validation tests.

Informatica Administrator (the Administrator tool)

The Administrator tool is a web application that you use to manage the Informatica domain. Create application services and Data Validation Option users in the Administrator tool.

JasperReports Server

JasperReports Server is a third-party tool that you can use to generate reports. JasperReports is a high-performance standalone or embeddable Business Intelligence (BI) platform that provides rich reporting and integrated in-memory analysis capabilities.

Data Validation Application Services

Data Validation Option uses application services in the Informatica domain to perform data integration and data validation operations. The application services that you use depend on the operations that you perform.

Data Validation Option uses the following application services in the Informatica domain:

PowerCenter Integration Service

The PowerCenter Integration Service runs the mappings that you create in the PowerCenter Client. When you run a validation test, the PowerCenter Integration Service generates validation mappings and stores them in the Data Validation Option target folder of the PowerCenter repository. It runs the validation mappings and writes the results to the Data Validation Option repository.

PowerCenter Repository Service

The PowerCenter Repository Service manages the PowerCenter repository. It refreshes the folders and objects that you see in the Data Validation Client.

Metadata Manager Service

The Metadata Manager Service manages the Metadata Manager repository. You can integrate Data Validation Option with Metadata Manager. The integration allows you to analyze the impact of test results on data sources. When you integrate Data Validation Option with Metadata Manager, you must create a PowerCenter resource for the PowerCenter repository.

Data Validation Repositories

Data Validation Option uses repository databases to store mapping metadata, testing metadata, and testing results. The databases that you use depend on the functionality that you implement.

The following repositories integrate with Data Validation Option:

PowerCenter repository

The PowerCenter repository contains mappings based on the tests that you create in the Data Validation Client. It also contains all the mappings and connections that you create within the PowerCenter Client. Create a separate folder in the PowerCenter repository to store the tests that you create in the Data Validation Client.

Data Validation Option repository

The Data Validation Option repository contains the test results. You can generate reports based on the repository views.

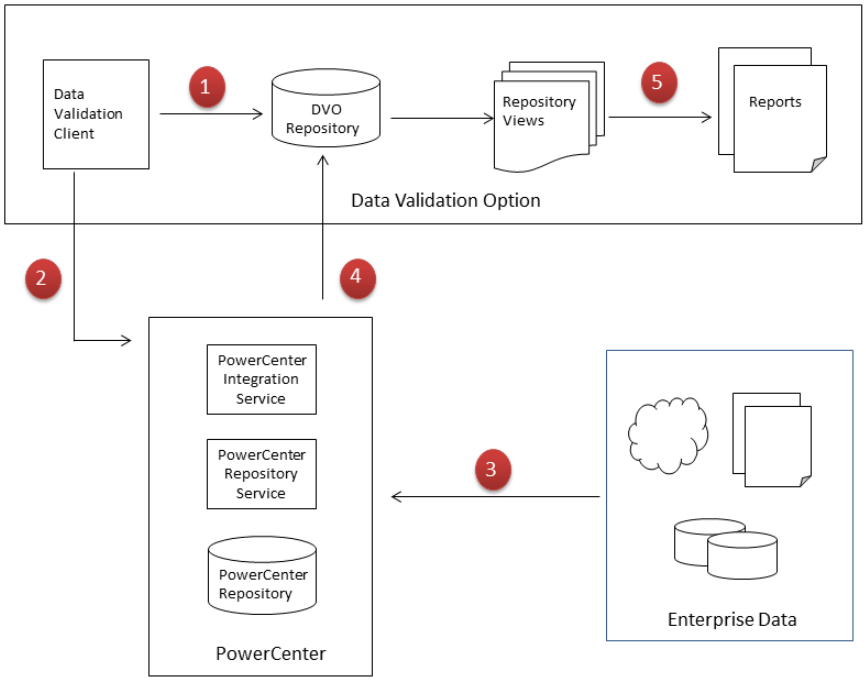
Metadata Manager repository

The Metadata Manager repository stores metadata models. You can run data lineage in Metadata Manager to analyze the data flow and to identify possible causes of invalid data in data validation tests.

Data Validation Option Process Architecture

The Data Validation process flow begins with defining tests and ends with users viewing test results.

The following image shows the process architecture of Data Validation Option:



1. Define a test in the Data Validation Client based on enterprise data that you want to validate.
2. Run the test from the Data Validation Client or from DVOcmd. Data Validation Option generates a mapping and session and it stores them in the PowerCenter Repository.
3. The PowerCenter Integration Service connects to the data sources and applies the test to the data.
4. The PowerCenter Integration Service writes the test results to the Data Validation Option repository.
5. You can run reports against the repository views that contain the test results.

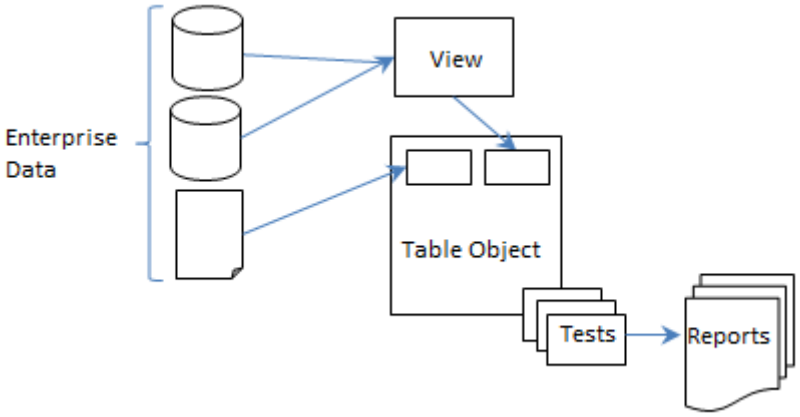
Data Validation Test Components

Use the Data Validation Client to configure validation tests and the test components. Configure a test based on a table object.

You create a table object to define the set of data that you want to validate. To define the set of data, you add one or more PowerCenter data sources or data validation views. Then you configure information such as key fields, WHERE clause, and connections.

After you create a table object, you can create one or more tests to validate the set of data. After you run the test, you can run summary and detailed reports to view test results.

The following image shows the Data Validation Option test components:



Data Validation Process

After you determine the validation requirements, you can create and run validation tests from within the Data Validation Client. You can view test results and run reports from within the Data Validation Client.

Use the following process to perform data validation:

1. Determine the validation requirements.
2. Optionally, create a data validation view to represent one or more data sources.
3. Create a table object that contains one or more data sources or views.
4. Create a test to validate data in the table object.
5. Run the test.
6. View test results.

Step 1. Determine the validation requirements.

Before you create a validation test, you first determine the testing requirements. Identify the data sources in the PowerCenter repository that contain data that you need to validate. If the validation is fairly straightforward, you can add the data source or data sources to a table object. If the validation is more complex, you might need to create a view with one or more data sources and then add the view to a table object.

Step 2. Create a data validation view.

You might need to create a data validation view based on an SQL query or based on one or more data sources.

When you create a data validation view, the PowerCenter Integration Service stores the view definition in the Data Validation Option repository. You can edit the view in the Data Validation Client.

Step 3. Create a table object.

After you create a view, you can add it to a table object. Create a table object that contains one or more table definitions or views. You can create a single table object or a table pair object. If you create a single table object, you specify the primary key. If you create a table pair object, you specify the join condition. You can include a WHERE clause to validate a subset of the data.

When you create a table object, the PowerCenter Integration Service stores the object in the Data Validation Option repository.

Step 4. Create a validation test.

After you create a table object, you can create a test based on the object. When you create the test, you choose the type of test that you want to run. You can choose an aggregate, set, or value test type. You configure test conditions based on the type of test that you choose.

Step 5. Run the test.

After you create a validation test, you can run it. When you run a validation test, the PowerCenter Integration Service creates the mapping and run-time objects in the Data Validation Option folder of the PowerCenter repository. It runs the mapping and loads the results into the Data Validation Option repository.

Step 6. View the test results.

After you run a test, you can view the summary of results in the Data Validation Client. You can see if the test passed or failed. The test fails if any row does not meet the validation condition. You can run reports from within the Data Validation Client. You can also view the session log from within the Data Validation Client.

Data Validation Views

A data validation view is a virtual view generated from one or more data sources that you want to test. Create a data validation view when you need to customize the set of data before you use it in a table object. After you create a data validation view, you can add it to a table object. You can also add some views to another view.

You can create the following types of data validation views:

Aggregate view

A view based on a single data source or another data validation view. You can create an aggregate view when you need to validate aggregate data from a nonrelational source, such as a flat file. When you create the view, you choose the group by columns, and you configure expressions to create the aggregated data.

For example, your company recently acquired another company, and you need to verify the average pay rate. You received a flat file that contains employee information. You create an aggregate view to group the data by employee pay grade. Then you run a test to identify the average pay rate of employees in each grade.

Join view

A view based on two or more heterogeneous data sources joined by key columns. Create a join view to validate data across data sources and to check for possible discrepancies. You can use a join view in an aggregate view or in a table object.

For example, you want to verify that all customer records that you migrated from multiple sources were moved to the target. You create a join view of the sources. You add the view and the target table to a table pair object. Finally, you create a test to verify that the customers in all the sources also exist in the target.

SQL view

A view based on an SQL statement. The view can include multiple data sources and multiple calculations to produce a single data source. You can create an SQL view based on one or more relational tables. If you create a view based on more than one table, all tables must be in the same database and use the same connection object. You can use an SQL view in an aggregate view, a join view, or a table object.

For example, you want a combined result set of two data sources. You create an SQL view that contains a UNION operator.

Lookup view

A view based on source and lookup data that you want to test for key values and orphan records. You can create a lookup view based on flat file and relational data sources. If the view contains two relational sources, the tables must be in the same database and use the same connection object. You can use a lookup view in a table object.

For example, you want to check the validity of contents in a target table against the contents in the source. You create a lookup view with the source table and the lookup table. You add the lookup view and the target table to a table pair object. Finally, you create a test to compare the IDs in the view and the target.

Data Validation Table Objects

A data validation table object (table object) is a data validation object that defines the set of data that you want to validate. It can contain one or more data sources or data validation views. You can create multiple validation tests based on one table object.

When you create a table object, you configure properties used to perform tests against the data. For example, you can choose to test a sampling of the data if you do not need to perform tests against the entire data set. You might also choose to configure an external ID that you can use to run tests from DVOcmd.

You can create the following types of table objects:

Single table object

A single table object represents one data source or validation view that you want to test. You must configure a primary key in a single table object. You might create a single table object to verify data, such as average sales, when you do not need to reconcile it to another data set.

Table pair object

A table pair object represents two data sources or validation views. Create a table pair object to compare one data source to another. You must configure the key relationships between the sources in a table pair object. You might create a table pair object to compare migrated data against the source of the migration.

Data Validation Tests

After you create a table object, you can create one or more tests to validate the data. You can configure test functions and conditions when you create a test. The type of test that you can create depends on the table object that you use. For example, you can test for null values on a single table object, but not on a table pair object.

When you create a test, you choose the type of test that you want to perform and the type of operation, such as equals or does not equal. If you want to set a threshold for a margin of error, you can choose the approximate operator. You can also configure functionality such as case sensitivity or an allowance for the maximum number of bad records. You can create multiple tests on a table object. When you run a test on a table object, you run all tests that you created on that object.

You can create the following types of data validation tests:

Aggregate tests

Retrieves summarized information about data contained in the data source. Use an aggregate test to verify whether all records were moved or to identify incorrect logic in WHERE clauses.

An aggregate test can detect the following problems:

- Lack of referential integrity in the source
- Rejected rows by the target system
- Incorrect logic in the mapping WHERE clauses

The following table describes the aggregate test functions:

Aggregate Test Functions	Description
Count	Counts non-null values.
Count Distinct	Counts distinct non-null values.
Count Rows	Counts all values including nulls.
Min	Calculates the minimum value for a field.
Max	Calculates the maximum value for a field.
Average	Calculates the average of a numeric field.
Sum	Calculates the total of a numeric field.

Perform aggregate functions in the following order:

1. COUNT and COUNT ROWS to count the number of rows
2. SUM to sum numeric fields
3. COUNT DISTINCT to compare detail and aggregate tables

For example, the target data for daily sales contains shows only total sales of $15,000. You know that your company sells at least 1,000 items a day and that the total sales should be between $100,000 and $200,000. You configure the following test functions:

```
COUNT_ROWS (any fld) > 1000
SUM (Amount) Between 100000, 200000
```

Set tests

Compares the distinct values of one field to the distinct values of another field. Determines if the set of values is equal in both data sources or if values are missing in one data source. You might use a set test to identify records in a fact table that are not in the dimension table. You can check target tables to find a lack of referential integrity in the target, either a child without a parent, or a fact record without a corresponding dimension table.

For example, you configure the fact table to be Table A and you configure the dimension table to be Table B. You configure the test `Set A in B` to perform the validation test. This test will verify that the fact foreign keys are in the parent table. You create an expression that concatenates composite keys, and you run the tests on that expression.

The following table describes the set test functions:

Set Test Functions	Description
Set A in B	Determines whether all distinct values of A are in B.
Set B in A	Determines whether all distinct values of B are in A.
Set A Equal B	Determines whether A and B have the same distinct values.
Set A Not In B	Determines whether A and B share no common values.

Value tests

Evaluates data based on values. Errors in complex transformations can manifest themselves in simple ways such as NULLs in the target, missing rows, or incorrect formats. To test for these types of errors, you can enforce constraints on target tables.

You might use an outer value test type to check the lack of referential integrity in the source and identify orphan records.

The following table describes the value and expression test functions:

Value Test Functions	Description
Value	Tests individual values across data sets. For example, `Value (FldA) Between 10,50 .`
Outer Value	Compares values across tables row-by-row to identify orphans across data sets.
Not Null	Determines whether values in fields are not null. For example, `NOT_NULL (FldD).`
Not Blank	Determines whether values in fields are not blank.
Unique	Determines whether values in fields are unique. For example, `UNIQUE (PK).`
Format	Determines whether the values in the field match a pattern or constraint value.

Data Validation Reports

Data Validation Option stores all test definitions and test results in the Data Validation Option repository. You can run reports to display test definitions and results.

You can use the BIRT reporting engine to generate the reports on one or more table objects. For example, you can run the Summary of Testing Activities report to determine the number of table pairs or single tables, the number of tests for each table pair or single table, and the overall test results.

You can use JasperReports Server to generate the reports and dashboards. A dashboard displays the test results from one or more reports.

Data Validation Example

You upgraded to the current version of PowerCenter, and you need to verify that mapping output has not changed. You compare the mapping results of the pre-upgrade mappings with the results of the upgraded mappings.

Step 1. Run a session.

Run a session in PowerCenter and use separate connections to write the target data for the upgraded mapping.

Step 2. Create a table pair object.

Create a table pair object in the Data Validation Client. You perform the following configuration:

- Add the tables. One table in the table pair contains the mapping results from the pre-upgraded mapping, and one table contains the mapping results from the upgraded mapping.
- Select the connections associated with each of the tables.
- Create a join relationship based on a primary key, CUST_ID.

Step 3. Create validation tests.

Create the following tests based on the table pair:

- VALUE test. The VALUE test compares the CUST_ID values row-by-row in each table and determines whether they are the same. Any row that exists in one table and not the other is disregarded. Add the following test condition:

  ```
  CUST_ID = CUST_ID
  ```

- COUNT_ROW test. The COUNT_ROW test compares the total number of values for the CUST_ID column.

Step 4. Run the tests.

When you run the test, the PowerCenter Integration Service creates and runs a mapping based on the test. It writes the test results to the Data Validation repository.

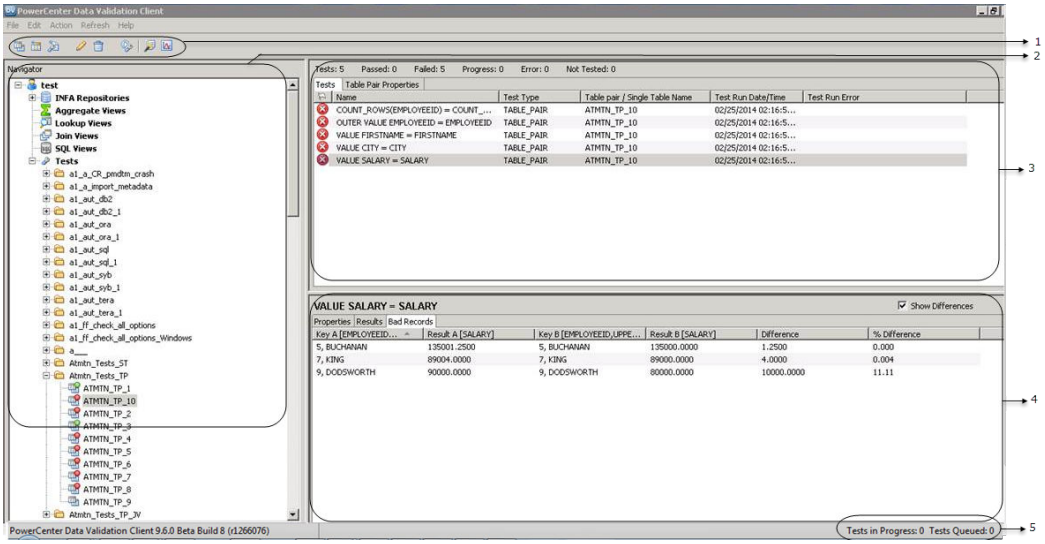
You view the results in the Data Validation Client to verify that the tests passed, indicating that both tables contain the same number of rows and that the customer IDs match.

Data Validation Client

The Data Validation Client is the client application that you use to configure and perform data validation operations.

When you log in to the Data Validation Client, your user workspace shows objects that you created or that have been copied into your user workspace. You can change objects in your user workspace.

The following image shows the Data Validation Client user interface:



1. Shortcuts
2. Navigator
3. Details Area
4. Properties Area
5. Status Bar

The Data Validation Client can display the following areas:

Shortcuts

Enable you to perform common tasks such as adding table pairs and single tables, adding and running tests, editing and deleting objects, generating reports, and running dashboards.

Navigator

Displays and organizes the following type of objects:

- Data Validation Option objects. You can view all Data Validation option folders, views, table objects, and tests objects for the Data Validation Option user that is logged in to Data Validation Option. The default folder contains table objects that you create in this folder. All folders, including the default folder, appear under the Tests node in the Navigator.
- PowerCenter objects. You can view PowerCenter repositories, folders, sources, and targets. The Navigator contains all PowerCenter repositories that you add to Data Validation Option. Expand INFA Repositories to view the PowerCenter repositories. Expand a repository to view the repository folders and the sources and targets in each folder.

Details area

Displays details about the objects that you create in the Data Validation Client, such as tests, table pairs, single tables, and views.

When you select a folder, table pair, or single table, the details area displays the following information about the tests associated with the object:

- Number of tests
- Number of tests passed
- Number of tests failed
- Number of tests in progress

- Number of tests not run because of errors
- Number of tests not run by the user

Properties area

Displays the following views based on the object that you select in the Navigator:

- **Properties** view. When you select an object in the details area, displays properties about the object.
- **Results** view. When you select a test or table object in the details area, displays test summary information, such as the test status, total number of records, and the number of bad records.
- **Bad Records** view. When you select a failed test, displays the bad records written to the Data Validation Option repository.

Statistics area

Appears when you select the Data Validation Option user in the Navigator. This area displays the number of repositories, table pairs, single tables, tests, views, and data sources that exist in the Data Validation Option repository. This area also displays information about running tests.

Status bar

Displays the number of tests in progress and the number of tests in queue to run.

CHAPTER 2

Repositories

This chapter includes the following topics:

Repositories Overview

Data Validation Option connects to a PowerCenter repository to import metadata for PowerCenter sources, targets, folders, and connection objects. Data Validation Option also connects to a PowerCenter repository to create mappings, sessions, and workflows in the Data Validation Option target folder.

When you add a repository to Data Validation Option, you add either a source or target repository. You can add one target repository and multiple source repositories.

The target repository contains metadata for PowerCenter sources, targets, folders, and connection objects. It also contains the Data Validation Option target folder. The target folder stores the mappings, sessions, and workflows that Data Validation Option creates when you run tests. Do not store other PowerCenter mappings, sessions, or workflows in this folder.

A source repository contains metadata for PowerCenter sources, targets, and folders. Add source repositories to Data Validation Option if you want to compare tables from different repositories. When you add a source repository, you must verify that all connection objects in the source repository also exist in the target repository. Data Validation Option uses the connection objects in the target repository when you run tests on table pairs.

The version number for a source repository can differ from the version number for the target repository. The version numbers for two source repositories can also differ.

Adding a Repository

You can add one target PowerCenter repository and multiple PowerCenter source repositories. Source repositories contain objects that you want to validate. Data Validation Option stores workflow objects for tests in the target repository. The associated PowerCenter user account must have Read permission on source repository connections and Read, Write, and Execute permissions on the target repository connection.

Verify that the PowerCenter Repository Service is running for each PowerCenter repository that you want to add. The corresponding PowerCenter Repository Service must be running to test the connection to a PowerCenter repository.

1. Right-click **INFA Repositories** in the Navigator.
2. Select **Add Repository**.

   The **Repository Editor** dialog box appears.
3. Enter the repository properties.

   Set **Contains Target Folder** to false when you add a source repository. Set to true when you add a target repository.
4. Click **Test** to test the repository connection.

   Data Validation Option verifies the connection properties. If the repository is a target repository, Data Validation Option also verifies the PowerCenter Integration Service and verifies that the Data Validation Option target folder and option results warehouse connection exist in the repository.

Editing Repositories

- To edit a repository, right-click the repository, and select **Edit Repository**, or double-click the repository.

  The **Repository Editor** dialog box appears. You can update any property that is enabled.

Deleting Repositories

- To delete a repository from Data Validation Option, right-click the repository, and select **Delete Repository**.

  Data Validation Option deletes the repository and all table pairs, single tables, tests, and views based on the repository data.

Refreshing Repositories

You refresh a source repository when the contents of the PowerCenter repository have changed. You usually refresh a target repository only when there are additions or changes to connection objects.

When you refresh a repository, Data Validation Option imports the source, target, folder, and connection metadata from the PowerCenter repository. Therefore, Data Validation Option objects that use changed or deleted PowerCenter objects might no longer be valid after you refresh a repository. If you created table

pairs, single tables, or tests with tables that were deleted from the PowerCenter repository, Data Validation Option deletes them when you refresh the repository.

1. To refresh all repositories at once, right-click **INFA Repositories** in the Navigator, and select **Refresh All Repositories**. To refresh one repository, right-click the repository, and select **Refresh Repository.**
2. When you refresh one repository, you can select the objects to refresh. Select one of the following options:

   Everything

   Data Validation Option reimports all source, target, folder, and connection metadata. It updates the folder list, and the Sources and Targets folders in the Navigator.

   Connections

   Data Validation Option reimports connection metadata. Select this option when a PowerCenter user adds, removes, or updates connection objects.

   Folder List

   Data Validation Option reimports folder metadata. It updates the folder list in the Navigator. Select this option when a PowerCenter user adds or removes folders.

   Folders (Sources and Targets)

   Data Validation Option reimports source and target metadata. It refreshes the contents of the **Sources** and **Targets** folders in each folder in the repository. Select this option when a PowerCenter user adds, removes, or modifies sources or targets in folders. If the PowerCenter folder contains a shortcut to an object in another folder, you must to refresh the other folder also.

   You can also refresh repository folders individually. You might refresh a folder after you refresh the folder list and Data Validation Option imports a new folder. To refresh a repository folder, right-click the folder in the Navigator, and select **Refresh Folder (Sources and Targets)**. Data Validation Option refreshes the contents of the **Sources** and **Targets** folders within the folder you refresh.

   Note: Refreshing everything in a repository or refreshing all repositories can take several minutes to several hours, depending on the size of the repositories. If you work with a small number of repository folders, you can shorten refresh time by refreshing the folders individually.

Troubleshooting a Repository Refresh

An error might occur when you refresh a repository using Data Validation Client or the DVOCmd RefreshRepository command.

Review the following troubleshooting tip for refreshing a repository:

An error occurs when you refresh a large number of connections from a PowerCenter repository.

If you try to refresh a large number of connections, the following error appears:

```
The pipe is being closed.
```

To resolve the error, reduce the number of PowerCenter connections to be imported into Data Validation Option.

To reduce the number of connections, complete the following steps:

1. Close the Data Validation Client or any open command lines.
2. Create a text file that specifies the connections to exclude from the refresh.

   Note: You can run the *pmrep* listConnections command to list all connections in a PowerCenter repository.

3. Name the text file as ExcludeConns.txt.

   Note: Data Validation Option searches for this file name before importing the connections.

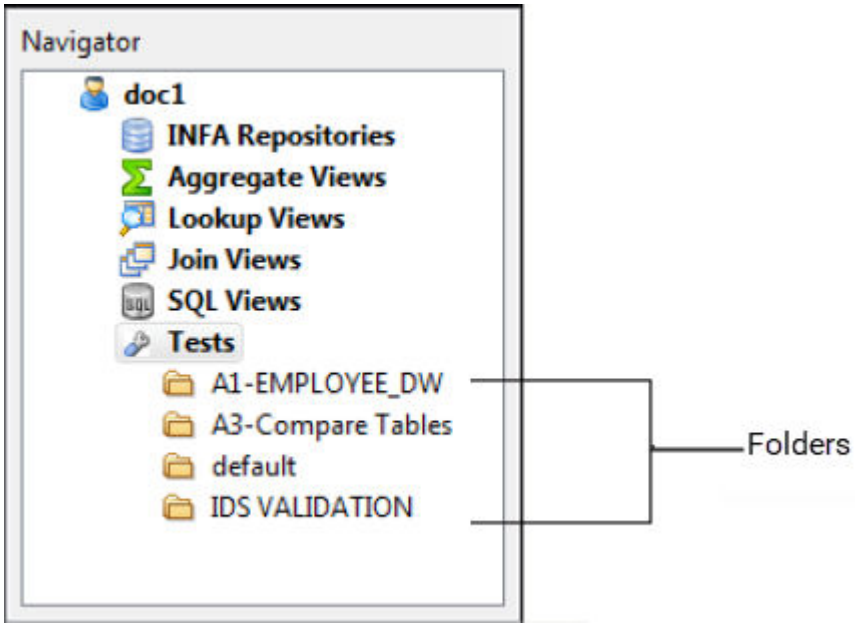
4. Add the file to the root directory where Data Validation Option is installed. For example, add the ExcludeConns.txt file to the following directory: `C:\ProgramFiles\Informatica10.4.0\DVO`
5. Open the Data Validation Client or the command line.
6. Refresh the connections again.

Folders

Folders contain single tables and table pairs that are created in the Data Validation Option repository.

By default, Data Validation Option places single tables and table pairs in the default folder. You can create folders to store and organize single tables and table pairs. You can create folders within other folders. You can move single tables or table pairs between folders. Within a folder, you can expand a single table or table pair to view the tests associated with it. Folder names are case sensitive.

The following figure shows folders in the Navigator:



Folders are not shared among different user workspaces. However, you can copy objects from the workspace of one user to the workspace of another user.

Folder Copy

You can copy the objects of a folder to another folder in your user workspace or the workspace of another user. Copy the folder objects to back up the objects or to share objects between user workspaces.

When you copy all objects in a folder, Data Validation Option creates a target folder and copies the table pairs and single tables from the source folder to the target folder. Data Validation Option also copies the tests that are associated with each table pair and single table. Data Validation Option does not copy test runs or the external IDs associated with table pairs or single tables.

If you copy a table pair or single table that is based on an SQL view or a lookup view to another user workspace, Data Validation Option copies the view to the target user workspace. If the target user workspace contains a view with the same name, you can use the view in the target user workspace or rename the copied view. Data Validation Option renames the copied view as "Copy <number> <original view name>."

Before Data Validation Option copies the folder objects, it verifies that the PowerCenter repository and all data sources associated with the objects exist in the target user workspace. If the PowerCenter repository or any required data source does not exist in the target user workspace, Data Validation Option does not copy the objects. Object names in Data Validation Option are case sensitive.

Copying a Folder

You can copy the contents of a folder to a different folder in your workspace or to a folder in another user workspace.

1. Select **Edit** > **Copy Folder**.

   The **Copy Folder Contents** dialog box opens.
2. Enter the source and target information.

   The following table describes the source and target properties:

Property	Description
Copy from User	Name of the source user. Data Validation Option copies the folder in this user workspace.
Copy from Folder	Name of the source folder. Data Validation Option copies this folder.
Copy to User	Name of the target user. Data Validation Option copies the folder to this user workspace. The source user and the target user can be the same.
Copy to Folder	Name of the target folder. The target folder must be unique in the target workspace.

3. Click **OK**.

Copying Objects

You can copy objects in a folder in your workspace to a different folder in your workspace or to a folder in another user workspace.

1. Select the objects that you want to copy.

   You can select objects of the same type from different folders.
2. Right-click the object and select **Copy**.

   The **Copy Object(s)** dialog box appears.
3. Enter the target information.

The following table describes the target properties:

Property	Description
User	Name of the target user. Data Validation Option copies the folder to this user workspace. The source user and the target user can be the same user.
Folder	Name of the target folder. The target folder must be unique in the target workspace.

Exporting Repository Metadata

You can export repository metadata to a file. You might want to export repository metadata when migrating from a development to a production environment or if you are asked to do by Informatica Global Customer Support.

To export repository metadata from Data Validation Option, right-click the repository, and select **Export Metadata**. Data Validation Option prompts you for a file name and file path.

Metadata Export and Import

Data Validation Option allows you to export and import test metadata from the repositories. Metadata import and export allows users to share tests and allows rapid generation of tests through scripting.

Scripting is particularly useful in the following scenarios:

- You have a very large number of repetitive tests for different table pairs. In this situation, it might be faster to generate the tests programmatically.
- The source-to-target relationships and rules are defined in a spreadsheet. This often happens during data migration. You can script the actual Data Validation Option tests from the spreadsheets.

You can import and export the following metadata:

- Table Pairs
- Single Tables
- PowerCenter Sources
- Aggregate views
- SQL views
- Lookup views
- Join views

Related Topics:

- "Metadata Import Syntax" on page 222

Exporting Metadata

You can export Data Validation Option metadata objects to an XML file. You can export particular objects or all objects. You can also export dependent objects. When you export objects, you can skip objects that are not valid.

1. Right-click the object and select **Export Metadata**.
2. To export all objects, select **File** > **Export All Metadata**.
3. Enter the name of the export file, and then click **Save**.
4. If any objects are not valid, specify whether to skip the objects.

Importing Metadata

You can import Data Validation Option metadata objects. When you import metadata objects, you can edit the metadata, overwrite metadata in the Data Validation Option repository, and generate value tests.

Before you import metadata, you can configure the import file to generate value tests. For example, to generate value tests for the CustDetail_CustStage table pair, add the following lines at the end of the table pair definition in the import XML file:

```
<Commands>
generate-tests("CustDetail_CustStage");
```

When you import metadata from an XML file, you can overwrite repository objects that have the same type and same name as the objects you import. If you do not overwrite objects and an object with the same name and same type exists in the repository, Data Validation Option does not import the metadata.

When you import the metadata, you can also edit the metadata. You might change the metadata based on the environment in to which you import the metadata. For example, you can change the connections, flat file locations, Data Validation Option repository, and PowerCenter repository that stores Data Validation Option objects.

1. To generate value tests, add the generate-tests command to the end of the metadata definition for the table pair in the import XML file.
2. To import and overwrite repository objects, select **File** > **Import Metadata (Overwrite)**.

   The **Import Metadata** dialog box appears.
3. To import metadata without overwriting repository objects, select **File** > **Import Metadata**.

   The **Import Metadata** dialog box appears.
4. Specify whether you want to edit the metadata before importing the metadata.

   If you do not edit the metadata, the Data Validation Option imports the metadata. If you edit the metadata, the **Import Metadata** dialog box appears.
5. Change the metadata, and then click **Import**.

Metadata Manager Integration

You can analyze the impact of test results on data sources if you enable Metadata Manager integration. You can view the metadata of the data source in the PowerCenter repository. To view the metadata of data sources, you must setup a Metadata Manager Service in the Informatica domain. You must create a PowerCenter resource for the PowerCenter repository that contains the data source.

Metadata Manager Properties

Configure the Metadata Manager properties to view the metadata of the data source in a PowerCenter repository.

The following table describes the Metadata Manager properties:

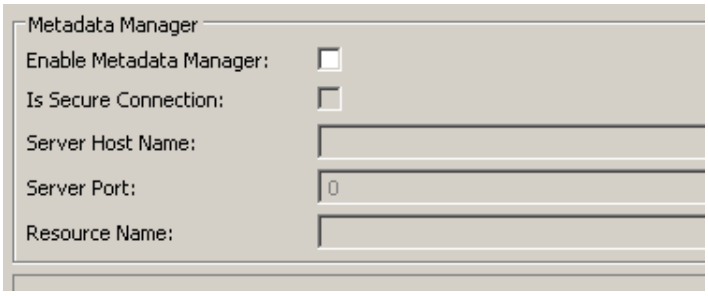
Property	Description
Enable Metadata Manager	Optional. Select to integrate Data Validation Option with the Metadata Manager Service.
Is secure connection	Optional. Indicator that determines whether the Metadata Manager Service runs over a secure connection.
Server Host Name	Optional. Host name of the machine on which the Metadata Manager Service runs.
Server Port	Optional. Port of the Metadata Manager Service.
Resource Name	Optional. Name of the PowerCenter resource created in Metadata Manager.

Configuring Metadata Manager Integration

Configure the connection to Metadata Manager to view additional metadata about PowerCenter data sources and to view the upstream and downstream impact summaries of data sources.

1. Right-click the repository for which you want to enable Metadata Manager Integration and select **Edit Repository**.

   The following figure shows the Metadata Manager properties:

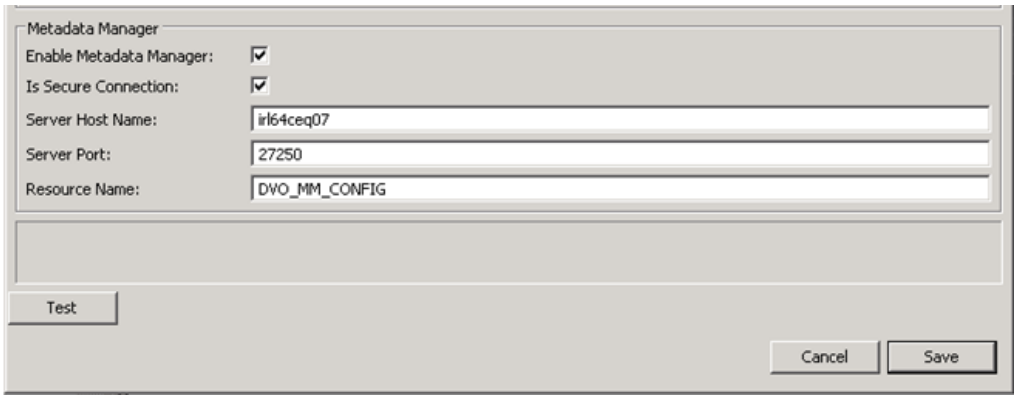
   

2. Select **Enable Metadata Manager Integration**.
3. Configure the Metadata Manager properties.
4. Click **Save**.

Integrating Metadata Manager with Data Validation Option - An Example

1. In Metadata Manager, create a resource and configure it to use the PowerCenter repository that Data Validation Option is using.
2. Enter the following properties in the **New Resource Wizard** of Metadata Manager, when you create the resource:
   - Name and description of the resource
   - Authentication and database configuration details
   - Folders in the PowerCenter repository that contains the data sources
3. In Metadata Manager, load the resource that you created.
4. In the Data Validation Client, configure the Metadata Manager properties to view the metadata of the data source in a PowerCenter repository.

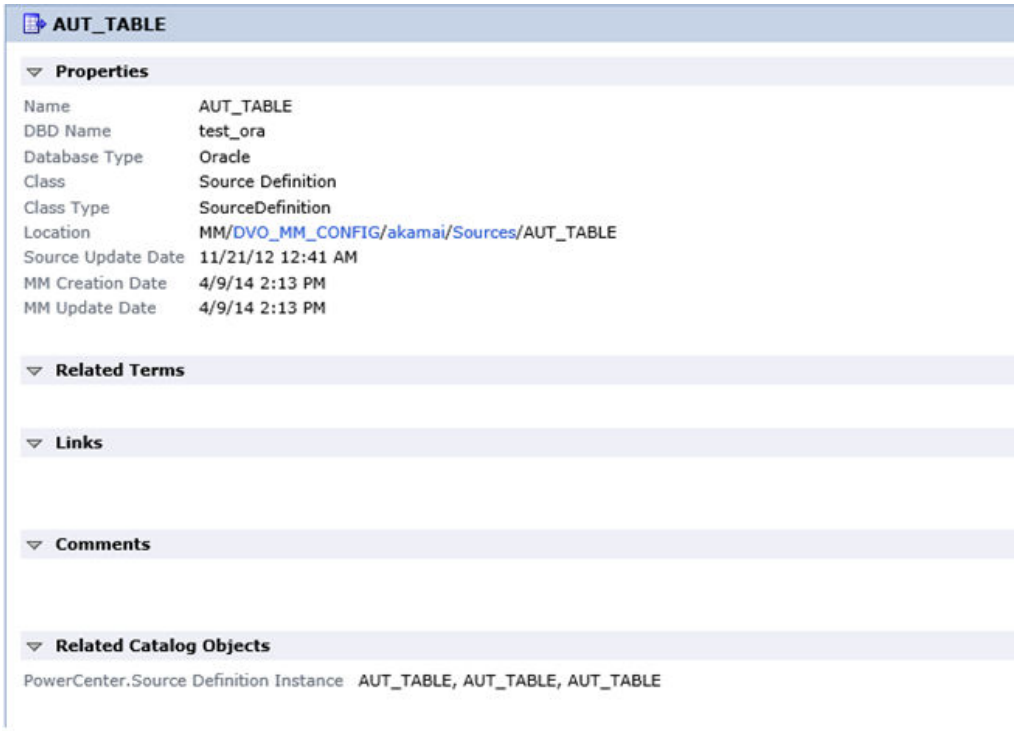
   The following image shows sample configuration options:



5. To view the metadata of an object in the Data Validation Option repository, right-click the data source and select **Get Metadata**.

   The Metadata Manager window appears with metadata of the data source.

   The following image shows the **Metadata Manager** window:



CHAPTER 3

XML Data Source

This chapter includes the following topics:

XML Data Source Overview

XML is a commonly used format for moving data within and across enterprises. It is easily transported, machine independent, and can handle complex data relationships. The hierarchical nature of XML makes parsing and processing XML data more complex than table or flat file data. Testing XML data in ETL or other application scenarios is much more tedious, time consuming, and error prone than flat file or relational data.

XML hierarchies need to be traversed and often flattened in order to process the underlying XML data. For example, when you test whether XML data was loaded correctly into a relational table, the data needs to be read, flattened, and loaded into tables and then tested against the relational structures. Developers typically write code or use tools like PowerCenter or SQL with XML extensions to flatten the XML hierarchy.

Data Validation Option removes the complexity of testing XML data and provides the flexibility to test in both XML and non-XML data scenarios. You can use Data Validation Option to flatten XML without staging, and then create and execute tests quickly and easily. You can test applications that process XML in the same way that you test applications with relational data. You can also test XML data with other types of data, such as flat files, mainframe data, or data on premise or on the cloud.

You can work with XML data in Data Validation Option in the following ways:

- You can include PowerCenter XML sources and targets as tables in single-table constraints and table pairs.
- You can include one or more XML groups in the XML hierarchy in a single-table constraint or table pair. By default, you can include one XML group in a single-table constraint or table pair.
- To include multiple XML groups, create a join view based on the XML file. The join view converts the hierarchical XML data to a relational structure. You can specify which XML groups to include in the join view. You can use the join view as a table in a single-table constraint or table pair.
- You can reuse the same join view for multiple XML files that have the same structure. For example, you create a join view based on an XML file and add the join view to a table pair. You create another table pair based on the same join view and override the XML connection properties. You change the XML connection to point to a different XML file in a different location.

XML Groups

An XML group is a related set of elements with the same parent element. The elements in an XML group are at the same hierarchy level in the XML definition.

For example, you have the following XML structure:

```
<Catalog>
  <Book>
    <Author></Author>
    <Title></Title>
  </Book>
</Catalog>
```

The XML structure has a Catalog XML group and a Book XML group. The Catalog XML group contains the <Book> element. The Book XML group contains the <Author> and <Title> elements.

When you add an XML file to a single-table constraint or table pair, you select an XML group. The child elements of the XML group appear as table columns in the single-table constraint or table pair. For example, you add the Book XML group to a single-table constraint. Author and Title appear as columns in the single-table constraint. You can create single-table constraint tests on these columns.

Note: An XML group in Data Validation Option is equivalent to an XML view in PowerCenter.

XML Data Conversion to a Relational Structure

You can create a join view based on the XML file to converts the hierarchical XML data to a relational structure. You can then use the join view as a table in a single-table constraint or table pair.

To create the join view, you create a self join based on the XML file. You can manually create or automatically generate a join view based on an XML file.

When you create the join view, you can specify which XML groups to include in the join view. An error occurs if you include multiple XML groups that are not connected.

For example, you have the following XML file:

```
<Root>
  <Customers>
    <Customer CustomerID="GLFM">
      <CompanyName>Great Lakes Food Market</CompanyName>
      <ContactName>Howard Snyder</ContactName>
      <ContactTitle>Marketing Manager</ContactTitle>
      <Phone>(503) 555-7555</Phone>
      <FullAddress>
        <Address>32 Baker Blvd.</Address>
        <City>Eugene</City>
        <Region>OR</Region>
        <PostalCode>97403</PostalCode>
        <Country>USA</Country>
      </FullAddress>
    </Customer>
    <Customer CustomerID="HCIS">
      <CompanyName>Hungry Coyote Import Store</CompanyName>
      <ContactName>Yoshi Latimer</ContactName>
      <ContactTitle>Sales Representative</ContactTitle>
      <Phone>(503) 555-6874</Phone>
      <Fax>(503) 555-2376</Fax>
      <FullAddress>
        <Address>165 Main St.</Address>
```

```
        <City>Elgin</City>
        <Region>OR</Region>
        <PostalCode>97827</PostalCode>
        <Country>USA</Country>
      </FullAddress>
    </Customer>
    <Customer CustomerID="LCCS">
      <CompanyName>Little Cap Country Store</CompanyName>
      <ContactName>Martha Lamponelli</ContactName>
      <ContactTitle>Marketing Manager</ContactTitle>
      <Phone>(509) 555-7969</Phone>
      <Fax>(509) 555-6221</Fax>
      <FullAddress>
        <Address>412 Orchestra Terrace</Address>
        <City>Walla Walla</City>
        <Region>WA</Region>
        <PostalCode>99362</PostalCode>
        <Country>USA</Country>
      </FullAddress>
    </Customer>
  </Customers>
</Root>
```

The XML file has the following XML groups and hierarchy: Root > Customers > Customer > Full Address.

An error appears when you automatically generate a join view and include the Customers XML group and the Full Address XML group, but exclude the Customer XML group. The Customer XML group is the link between the Customers XML group and Full Address XML group.

Join View Properties for an XML File

Join view properties include table definitions and join conditions for each of the data source.

The following table describes the join view properties:

Description

Description of the join view for the XML file.

Source Dir

Directory that contains the XML file. The path is relative to the machine that runs the PowerCenter Integration Service.

Source File

File name with the file name extension.

File Type

Type of file. Select **File Contains Source Data**.

Available Groups

The XML groups that you can include in the join view.

Selected Groups

The XML groups that you include in the join view.

Automatically Generating a Join View for an XML File

You can automatically generate a join view based on an XML file to convert the hierarchical XML data to a relational structure.

1. In the Navigator, select the folder that contains the XML file.

   XML files contained in the folder appear in the details area.

2. Right-click the XML file, and then select **Create Join View**.

   The **XML Join View Creator** dialog box appears.
3. Enter the join view properties for the XML file.
4. Add the XML groups that you want to include in the join view.
5. Click **OK**.

   A message states whether Data Validation Option created the join view successfully.
6. Click **OK**.

   The join view appears under Join Views in the Navigator.

You can edit the join view to change the join view properties and configure the output fields of the join view.

Rules and Guidelines for XML Files

Certain rules and guidelines apply to XML files that you include in table pairs and single-table constraints.

Use the following rules and guidelines for XML files:

- Data Validation Option ignores content in a CDATA section of an XML file.
- Data Validation Option cannot process complex XML files, such as those that require the Unstructured Data transformation in a PowerCenter mapping.
- The XML files must conform to the PowerCenter rules and guidelines for XML sources and targets.
- You might encounter memory and row-size errors when you run a test on an XML file that contains columns with unlimited lengths. To resolve the errors, re-create the XML file definition in PowerCenter to override all infinite lengths with a value. Next, refresh the latest XML file definition in Data Validation Option.

For more information about PowerCenter rules and guidelines for XML sources and targets, see the *Informatica PowerCenter XML Guide*.

CHAPTER 4

Tests for XML Data Sources

This chapter includes the following topics:

Test for XML Data Sources Overview

You can import XML source and target definitions into Data Validation Option from PowerCenter. You can use the XML sources and targets in views, table pairs, and single tables for testing.

Data Validation Option flattens the XML hierarchy to make it accessible for testing in views, table pairs, and single tables.

You can use the following methods to flatten an XML hierarchy in Data Validation Option:

- Select a single XML Group in the XML hierarchy and perform tests on the fields in that group.
- Create self-joins of XML Groups in a Join View to flatten the XML.
- Automate the creation of a Join View to flatten XML directly for an XML Source.

After the XML hierarchy is flattened, you can test the XML data in Data Validation Option like any other type of data.

PowerCenter must be able to parse and process any XML definitions and data from Data Validation Option. The rules that apply to XML data in PowerCenter also apply to XML data in Data Validation Option.

When you test XML data in Data Validation Option, consider the following rules and guidelines:

- XML schema size must be smaller than 400 elements and less than 100KB in size.
- XML data file size must be less than 10MB.
- Complexity profile is limited to three hierarchy levels with support for complexType elements Sequence, Any, and Choice.
- The XML import wizard can create a maximum of 400 XML groups.

- PowerCenter strictly follows the XML structure. For example, the W3C XSD rules allow attribute values in any order and not as they are defined in the XSD. In PowerCenter, the attribute values must be in the order that they are specified in the XSD.

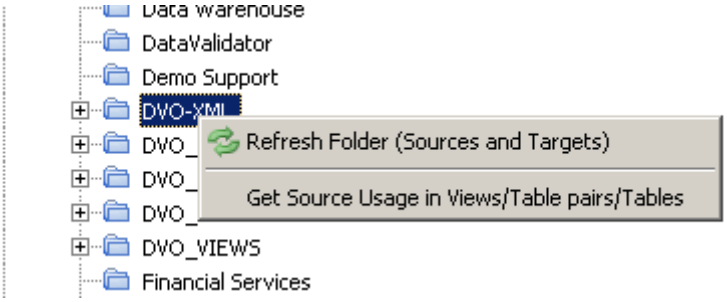
You can use Data Validation Option to test XML data in the following ways:

- Import XML definitions and view them in Data Validation Option.
- Use XML sources directly in table pair or single table tests.
- Flatten XML using a join view.
- Edit table and column names in a join view.
- Override XML source file information.

Importing XML Definitions and Viewing them in Data Validation Option

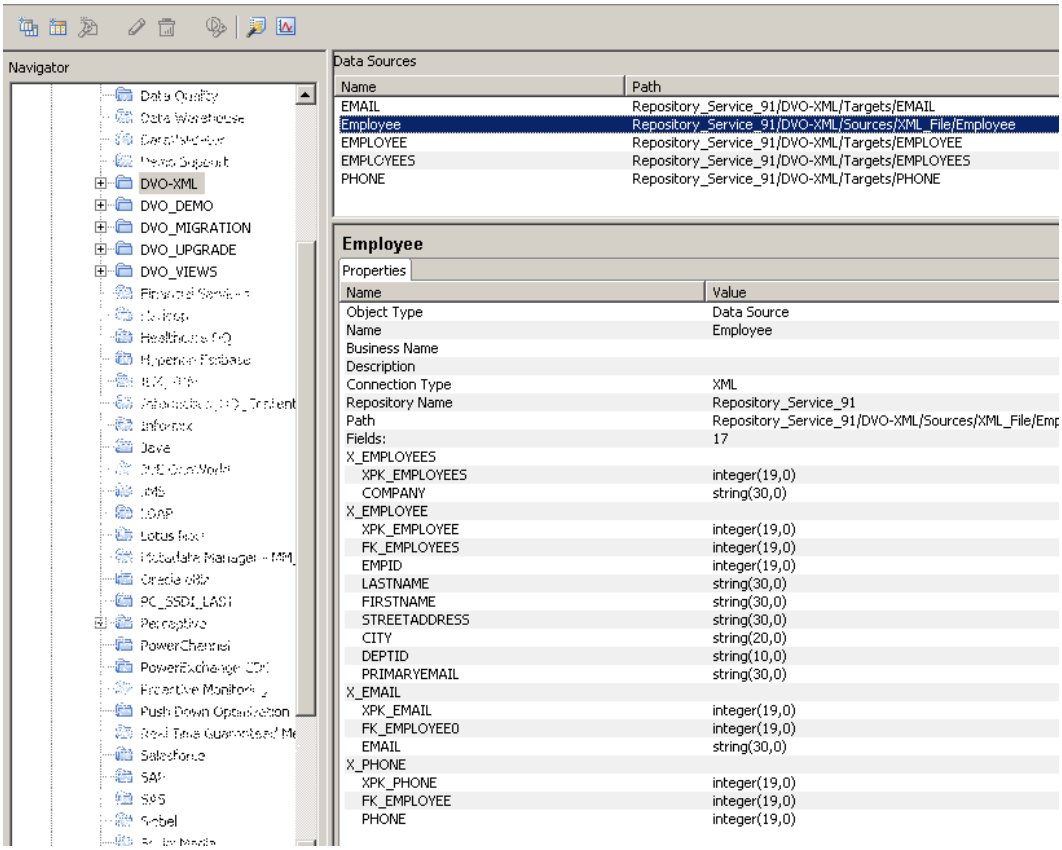
You import XML definitions into Data Validation Option the same way you import any other supported source or target. You can import sources or targets at the folder level or at the repository level.

The following image shows the option to import a source or target at the folder level:



Once imported, the XML metadata can be viewed by clicking on a Repository source or target on the navigator.

The following image shows the details of a source imported from PowerCenter displayed in the Properties tab:



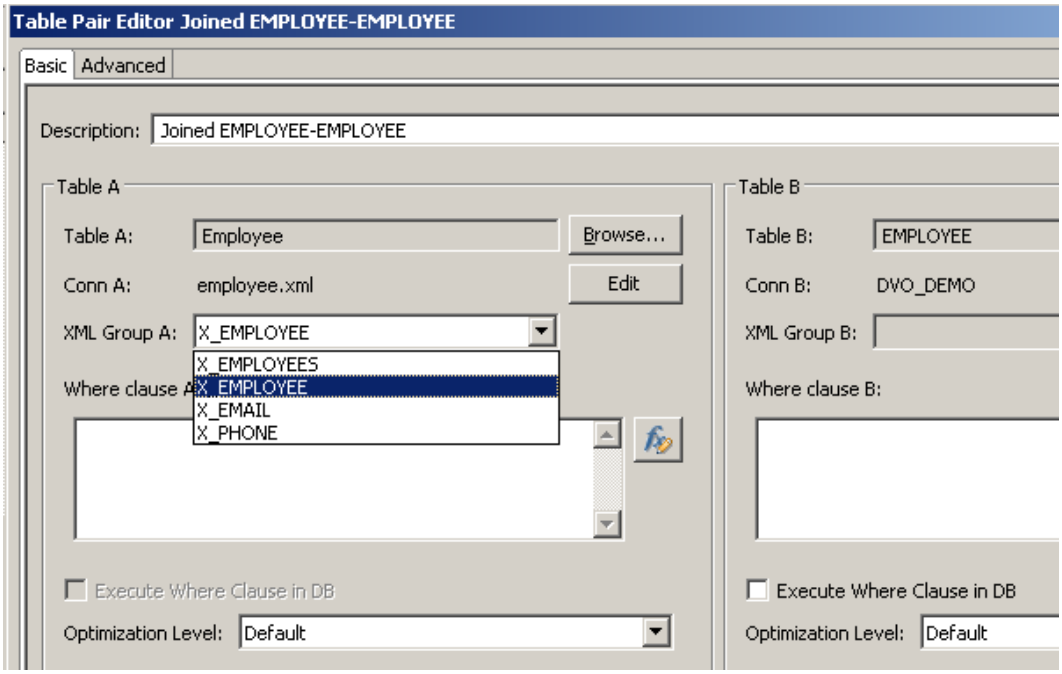
In this example, the XML definition for Employee has four groups: X_EMPLOYEES, X_EMPLOYEE, X_EMAIL, X_PHONE. For each group, the elements in the group along with their type, precision and scale are displayed.

Using XML Sources Directly in Table Pair or Single Table Tests

In a Table Pair or Single Table, when an XML definition is used, the XML Group drop-down is enabled, showing the groups for that XML definition.

If you are using an XML definition directly in a Table Pair or Single Table, select the XML Group from the drop down to identify which data you want to test.

The following image shows how to select an XML group:



Once the XML Group is selected, the elements in that group are available for testing just like any other data source.

Note that this method restricts you to only the elements in the particular group you select. In the above example, if the XML file has the structure shown below, only those fields under X_EMPLOYEE , including EMPID, LASTNAME, FIRSTNAME, are accessible. To access fields under one of the other groups, select one of those groups instead.

The following image shows the fields for each XML group:

X_EMPLOYEES	
XPK_EMPLOYEES	integer(19,0)
COMPANY	string(30,0)
X_EMPLOYEE	
XPK_EMPLOYEE	integer(19,0)
FK_EMPLOYEES	integer(19,0)
EMPID	integer(19,0)
LASTNAME	string(20,0)
FIRSTNAME	string(20,0)
STREETADDRESS	string(30,0)
CITY	string(20,0)
DEPTID	string(3,0)
PRIMARYEMAIL	string(20,0)
X_EMAIL	
XPK_EMAIL	integer(19,0)
FK_EMPLOYEE0	integer(19,0)
EMAIL	string(20,0)
X_PHONE	
XPK_PHONE	integer(19,0)
FK_EMPLOYEE	integer(19,0)
PHONE	string(20,0)

Note: The columns starting with XPK_, such as XPK_EMPLOYEES and XPK_EMAIL and FK_, such as FK_EMPLOYEES and FK_EMPLOYEE, are columns that are automatically added when the XML definition is

imported into PowerCenter. These are primary and foreign key columns used to associate related data across different groups. For example, primary and foreign key columns associate X_EMPLOYEE to X_EMAIL.

XML group definitions have the following limitations:

- An XML definition can have up to 400 groups.
- A group in a source definition does not require a key.
- A group can have one primary key.
- A group can be related to several other groups, and a view can have multiple foreign keys.
- A column cannot be both a primary key and a foreign key.
- A foreign key always refers to a primary key in another group. You cannot use self-referencing keys.
- A generated foreign key column always refers to a generated primary key column.

For more information, see the *PowerCenter XML Guide*.

Flattening XML using a Join View

To select and test columns across groups, the XML hierarchy must be flattened by joining across the XML groups.

In PowerCenter, this is normally done manually by developers when they create mappings sourcing XML.

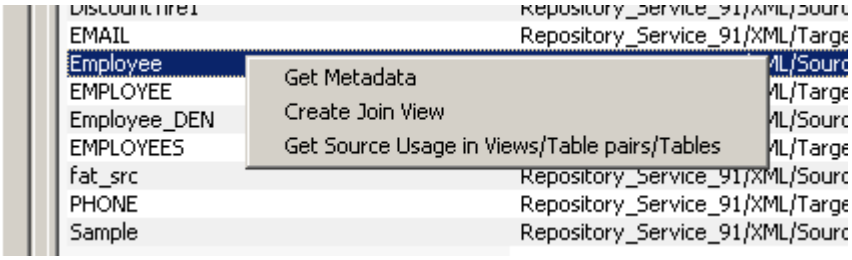
In Data Validation Option, this can be done using the Join View to create joins across selected XML groups in the XML structure.

There are two ways to create these Join Views for XML data:

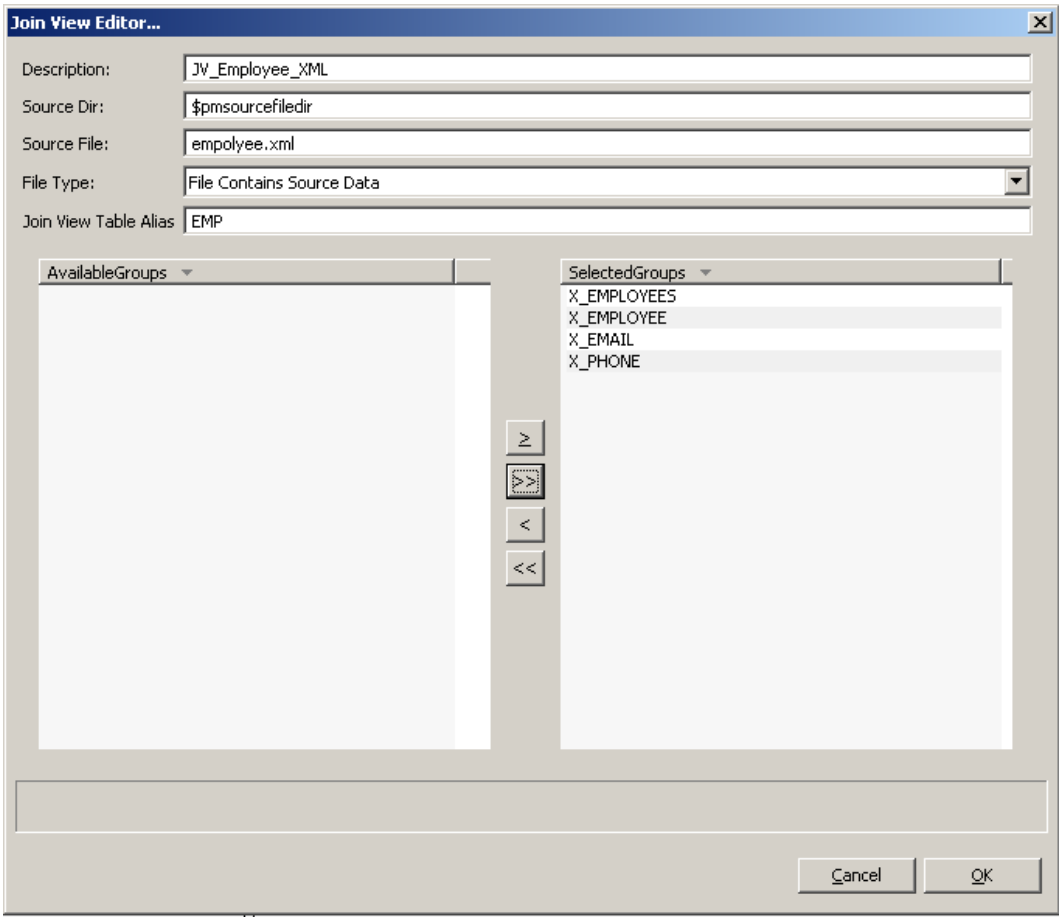
- Manually creating joins using the standard Join View dialog.
- Using the Create Join View dialog specifically available to flatten XML sources/targets.

This example explains how to do create a join view using the Create Join View dialog.

Right-click on an XML definition and select**Create Join View**.



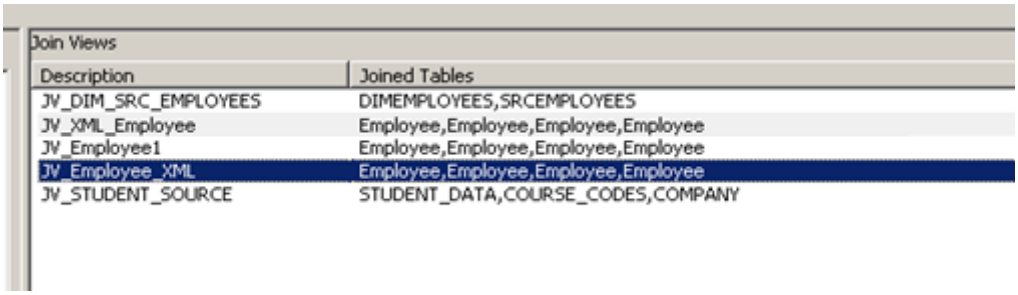
The XML Join View creation dialog appears. Fill in the fields in the dialog and select the groups you want to Join.



Note: The Join View Table Alias and Group names will be used by default to prefix the actual column names of the elements in the generated Join View. Thus an element called EMAIL in a Group called CONTACT with Join View Table Alias called EMPLOYEE will be EMPLOYEE_CONTACT_EMAIL. You can override this in the Join View Table Alias field.

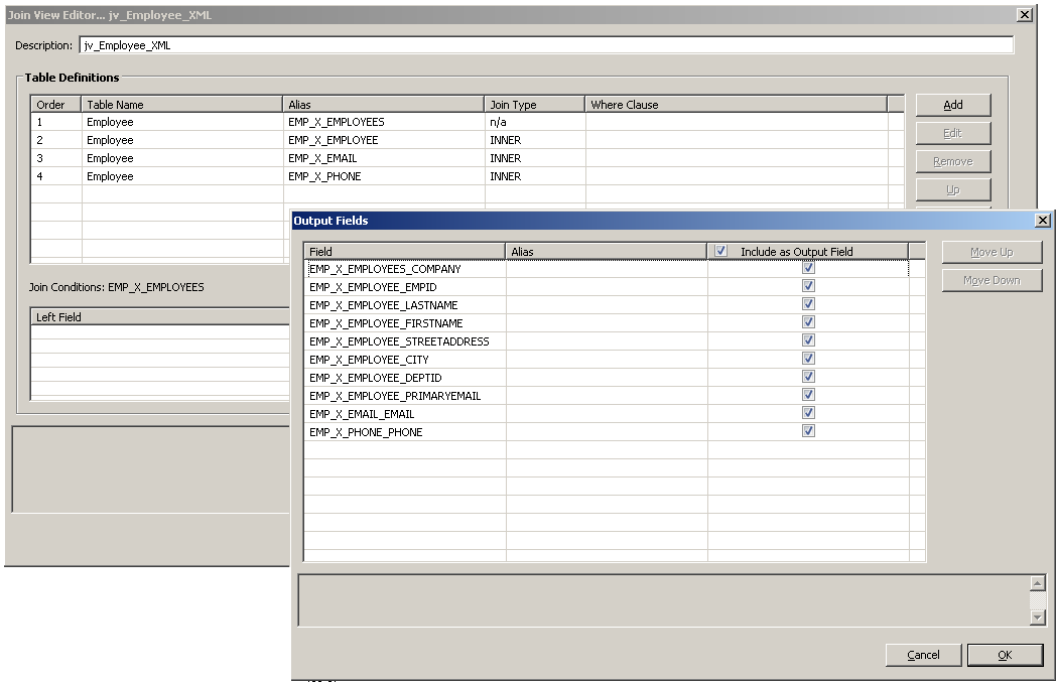
Once you have entered all the information, click OK to create the Join View.

The Join View can be seen in the Join View section of the Navigator.



Looking at the Join View, there are 4 "tables" joined, one for each group in the XML definition. All fields are selected for output.

You can edit any aspect of the view to suit your needs, such as reducing the number of output fields, changing their order or adding joins with other data sources. For example, you can add a join if one is needed between the XML and a relational table for testing.

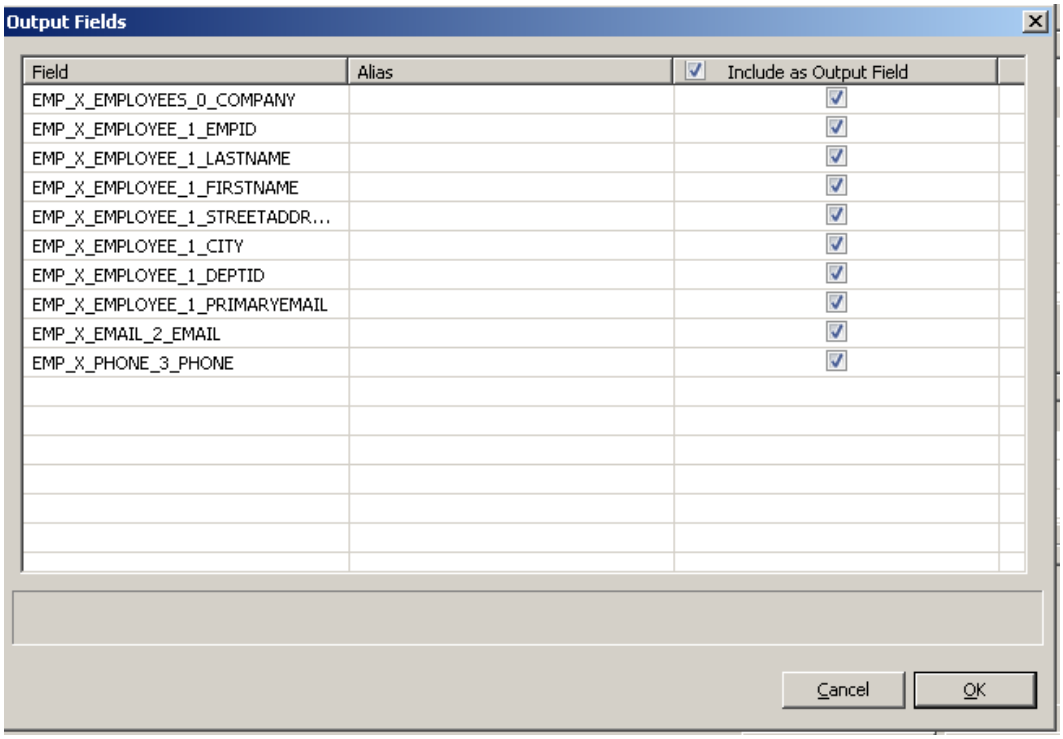


Editing Table and Column Names in the Join View

After you create a Join View to flatten the XML, you can edit the column names that are exposed by the Join View.

Note: Editing table and column names is a general Join View functionality and is not specifically designed to be used with XML. But given the default naming convention when using XML, it is a common scenario to rename columns to make testing simpler.

The following image shows the columns in the **Select Output Columns** screen:



The Alias column lets you override the name of the fields. Use this column to create shorter, more "friendly" names.

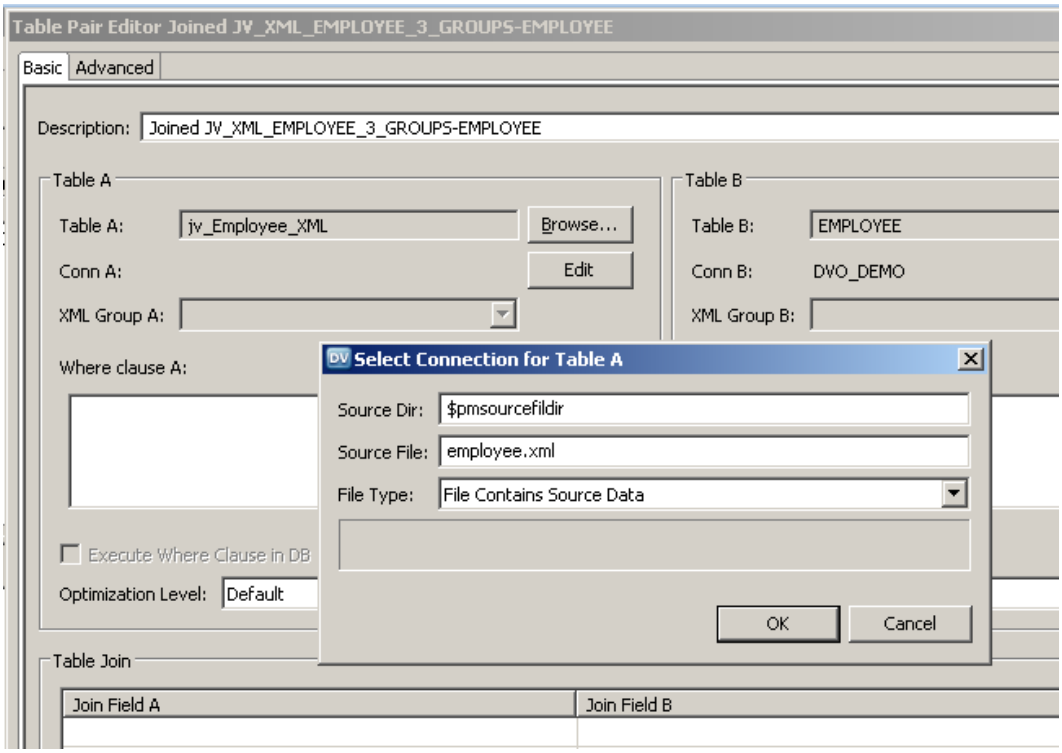
The **Include as Output Field** column lets you select which columns to expose to Table Pairs and Single Tables when the Join View is used.

Overriding XML Source File Information

There can be multiple XML files that match an XML definition and need to be tested, similar to flat files. There may be one flat file source or target definition, but there may be multiple flat files with data that are used for testing.

When a Join View is used to flatten an XML structure, it becomes a general definition to process any XML file that conforms to that structure. Thus, the file location and the file name can change even if they are specified in the original dialog used to create the Join View. It is possible to use the same Join View definition to process any file that conforms to that structure.

To use the same name, the path and file name specified in the connection can be over-ridden when the Join View is used in a Table Pair or Single Table. Click the Conn A or Conn B button to show and edit the Connection details.



Note: Only Join Views that flatten a single XML structure and do not include any other tables or files can be over-ridden this way. If the Join View includes other objects or tables, the over-ride is not possible.

Troubleshooting

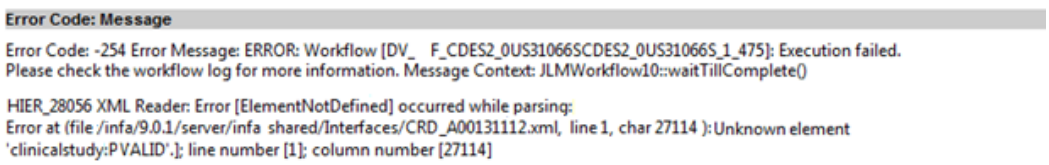
Unlike relational tables, which are relatively straight forward, XML structures can be quite complex, and span multiple files or locations. In short, there are more "moving parts" when dealing with XML, and each part has to be correct and consistent with the other parts.

When XML errors are encountered, there are several immediate areas to consider and verify.

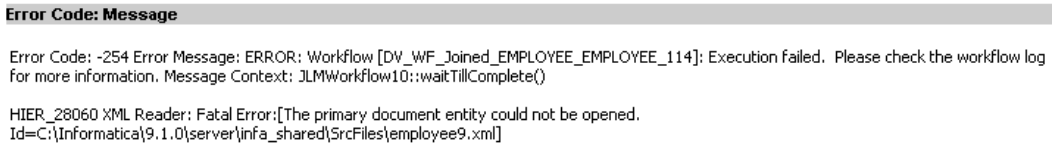
- Does the XML data match the schema?
- Does the data match the PowerCenter metadata definition?
- For multi-file schemas, are the files and relative file path references matching?
- Are all the expected files present?

The error messages displayed in the Data Validation Option screens are the first place to look.

For example, the following error message indicates that the data does not match the schema. An element in the XML file is not defined in the XML definition.



In the following example, the message indicates that the expected XML file is not present. This means the file could be missing, the file name could misspelled, or the directory or path was entered incorrectly.



Use of Multi-file schemas could also cause these types of errors if the relative file paths are not correct.

These are the easier problems to solve because they follow well defined patterns and error messages are reasonably specific.

Maximum Row Size Limit

Often, when importing schema definitions into PowerCenter, string fields are imported with "infinite" length. This is simply a convenience when the maximum size of the field is not known in advance.

"Infinite" length strings are given a physical length in PowerCenter of 898989 characters.

In most cases, having many fields this size does not cause errors, because the maximum row size that PowerCenter can process is 100MB. But some transformations have restrictions on their maximum row size. The Sorter can handle a maximum row size of 8MB, which is a significantly smaller than the maximum row size of PowerCenter in general.

Sorters are used in Data Validation Option mappings to sort data before Joins. This is a best practice when using PowerCenter as Joiners process sorted inputs almost an order of magnitude more efficiently than Joiners with unsorted data.

If an XML definition has multiple "infinite" string fields, the following error may appear in the session log when a Data Validation Option job fails:

```
>SORT_40414 : Error: Total row size [12586408] in transformation [NAME_6] is more than
the allowed maximum [8388607].
```

This error is typically caused by "infinite" length strings in the PowerCenter source or target definition.

To resolve the problem, update the PowerCenter XML definition that is causing the error, and replace "infinite" field lengths of 898989 with field lengths based on the size of the actual data.

CHAPTER 5

Connections

This chapter includes the following topics:

Connections Overview

Data Validation Option uses connections to access the underlying data source for single tables, table pairs, and views. Data Validation Option connections refer to the PowerCenter connection objects.

The following table describes data sources that are applicable to Data Validation Option:

Data Sources	Types of Data Sources
Enterprise application	- Salesforce - SAP - SAS
File	- Flat files - XML
PowerExchange	- Adabas - DB2 for i5/OS - DB2 for z/OS - IMS - Sequential files - VSAM
Relational	- IBM DB2 - Microsoft SQL Server - Netezza - ODBC - Oracle - Sybase - Teradata

Data Validation Option uses a native connection for all types of data sources. Data Validation Option connects to these data sources through ODBC.

The connection properties vary based on the type of data source. You do not have to configure connection properties for all connection types. For example, when you select a Salesforce or SAS connection for a table pair, you do not have to configure any connection properties.

Note: You cannot override the owner name for Salesforce, SAS, and SAP data sources.

File Connections

You must create a connection before you can use file sources in Data Validation Option.

You can connect to different file sources Data Validation Option. You can create connections to file sources in the following ways:

- Flat File and XML Connection Properties
- Mainframe Flat File Connection Properties

Flat File and XML Connection Properties

You must configure the connection properties when you use a flat file or XML data source in a table-pair object, single-table object, join view, lookup view, or an aggregate view.

The following table lists the connection properties for a flat file or XML data source:

Property	Description
Source Dir	Directory that contains the file. The path is relative to the machine that runs the PowerCenter Integration Service.
Source File	File name with the file name extension.
File Type	Type of file. Select one of the following values: - File Contains Source Data. Select if the file contains data. - File Contains a List of Files. Select if the file contains a list of flat or XML files. When you select a file list, Data Validation Option processes all files in the list.

Mainframe Flat File Connection Properties

You must configure the connection properties when you use a flat file or XML data source in a table-pair object, single-table object, join view, lookup view, or an aggregate view.

The following table lists the connection properties for a mainframe flat file data source:

Property	Description
Connection	PowerCenter connection object to connect to the mainframe flat file data source.
Source File	File name with the file name extension.

Property	Description
File Type	Type of file. Select one of the following values: - File Contains Source Data. Select if the file contains data. - File Contains a List of Files. Select if the file contains a list of flat or XML files. When you select a file list, Data Validation Option processes all files in the list.
Override Owner Name	If you use a PowerExchange data source as a target, you must override the owner name.

File Lists

You can create a connection to a group of flat files, mainframe flat files, or XML files.

Before you create a connection to a group of files, you must create a file that lists the names of all mainframe flat files, or XML files in the group. If the file contains a list of files, you must create a line feed after the last filename in the list.

When you create the connection, you specify the name and location of the file that contains the list of files.

Relational Connection Properties

You must configure the connection properties when you use a relational data source in a table-pair object, single-table object, join view, lookup view, SQL view, or an aggregate view.

The following table lists the connection properties for a relational data source:

Property	Description
Connection	PowerCenter connection object to connect to the relational data source.
Override Owner Name	Override for the owner or name of the relational table. If you are using PowerExchange data source as a target, you must override the owner name.

Owner Name and Table Name Override

You can override the owner name or table name for a relational table in a single table or table pair.

You may need to override the owner name if the table exists in a different schema. You may need to override the table name if the table name changes or you want to use a different table that has the same table metadata.

To perform the override, enter the new owner name, table name, or both in the Override Owner Name connection property for the relational data source. When you enter the new owner name or table name, you must use a valid syntax.

Note: You must have Execute permission on the owner name to override the owner name.

The following table specifies the required syntax for each type of override:

Type of Override	Syntax
Change owner name.	<Owner Name>. For example, enter `DEV_ENVIRONMENT`.
Change table name.	/<Table Name>. For example, enter `/CUSTOMERS`.
Change owner name and table name.	<Owner Name>/<Table Name>. For example, enter `DEV_ENVIRONMENT/CUSTOMERS`.

SAP Connection Properties

You must configure the connection properties when you use an SAP data source in a table pair, single-table constraint, or join view. You cannot override the owner name for an SAP data source.

Data Validation Option uses stream mode for the installation of ABAP programs. It does not use FTP mode. SAP data sources must not contain the backslash (/) character in field names.

The following table lists the connection properties for an SAP data source:

Property	Description
Connection	PowerCenter connection object to connect to the SAP data source.
SAP User Name	SAP source system connection user name. Must be a user for which you created a source system connection.
SAP Password	Password for the user name.
SAP Client	SAP client number.
SAP Language	Language you want for the mapping. Must be compatible with the PowerCenter Client code page. Data Validation Option does not authenticate the value. Ensure that you enter the correct value so that the tests run successfully.

SAP Connect String	Type A or Type B DEST entry in saprfc.ini. For example, enter `GE6` for the following Type A entry:
	`DEST=GE6`
	`TYPE=A`
	`ASHOST=gcssapec6.informatica.com`
	`SYSNR=00`
	`RFC_TRACE=1`

CHAPTER 6

Expressions

This chapter includes the following topics:

Expressions Overview

You can create expressions that represent field values and filters.

Create a field expression to change the value, datatype, scale, or precision of a field. For example, you create a case-sensitive join condition between two columns that have the same values. However, the values have different cases. To match the values, you can include a field expression for each column in the join condition to convert all values to uppercase letters.

Create different types of filter expressions to increase test performance. A filter reduces the number of records that are processed in single-table or table-pair tests. For example, you can create a WHERE clause expression in a single table definition to process a subset of the records in a relational table.

You create expressions in the Expression Editor. You can manually enter an expression. Or, you can select fields, PowerCenter functions, and Data Validation Option parameters from the Expression Editor. If you create an expression for a test condition, you can also select an operator to compare two values.

When you create an expression, you must use the proper syntax based on the component that processes the expression. By default, PowerCenter processes expressions. However, you can configure the data source to process some expressions.

You must also enter a valid expression. You can validate PowerCenter expressions in the Expression Editor.

You can use one or more parameters in the WHERE clauses of table pairs and single tables. Use parameters in the WHERE clause to dynamically change the WHERE clause after each test run.

Expression Types

You can create multiple types of expressions in Data Validation Option.

You can create the following types of expressions:

- Field
- WHERE clause
- Test condition

Field Expressions

A field expression represents the value of a field. Create a field expression to change the value, datatype, scale, or precision of a field.

For example, you perform a case-sensitive comparison between the field values of the LAST_NAME fields in two tables. The first table contains values with lowercase letters. The second table contains values with uppercase letters. To match the field values, you convert the uppercase letters in the second table to lowercase. You enter the following field expression for the field of the second table: `lower(LAST_NAME)`.

You can include field expressions in join conditions for table pairs, join views or aggregate views or in source-to-lookup relationships for lookup views.

When you create a field expression, you also specify the datatype, precision, and scale of the expression result. You must ensure that the datatype, precision, and scale of the field expression are valid. For example, you create a join condition between a field and a field expression. The datatype of the field expression must be compatible with the datatype of the other field in the join condition.

Rules and Guidelines for Field Expressions

Certain rules and guidelines apply when you create a field expression.

Use the following rules and guidelines for field expressions:

- If you do not specify the correct datatype, precision, or scale, the test produces an error.
- The datatypes that you can select are PowerCenter transformation datatypes.
- The precision and scale of the field expression must match the precision and scale used in PowerCenter for the specified datatype.

WHERE Clause

A WHERE clause filters the records processed by all tests in a single table or table pair. To increase the performance of all tests defined for a single table or table pair, add a WHERE clause.

You can use a parameter in the WHERE clause to dynamically filter the records that Data Validation Option tests.

Test Conditions

A test condition filters the records processed by a test in a single table or table pair. To increase the performance of a particular test defined for a single table or table pair, add a test condition.

For example, you want to exclude telephone extension numbers from a VALUE test for a table pair. You want to exclude telephone extension numbers that contain fewer than three characters.

Create the VALUE test based on the following criteria:

- Table A.EXT = Table B.EXT
- Condition A = LENGTH(EXT)<3
- Condition B = LENGTH(EXT)<3

Add a test condition instead of a WHERE clause when you want to apply the filter on a particular test. If you enter a test condition and WHERE clause in a single table or table pair, Data Validation Option applies the WHERE clause to all tests before it applies the test condition.

If you create a test condition and WHERE clause on a joined table pair, Data Validation Option applies the WHERE clause before it joins the tables. Data Validation Option applies the test condition after it joins the tables.

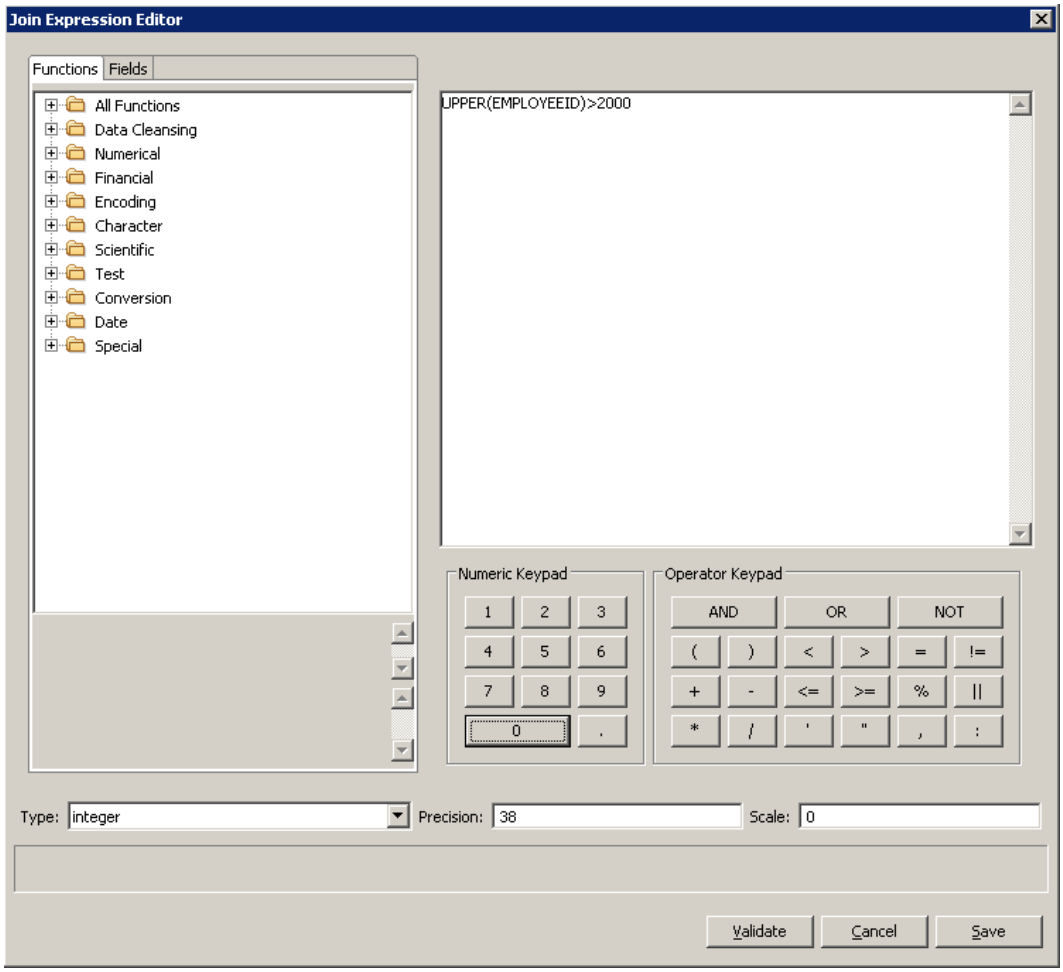
Expression Editor

You can use the Expression Editor to create expressions. Based on the type of expression that you create, you can select fields, PowerCenter functions, and Data Validation Option parameters in the Expression Editor.

To prevent expressions that are not valid because of misspellings and syntax, you can select fields, PowerCenter functions, and Data Validation Option parameters in the Expression Editor. You can select PowerCenter functions from the **Functions** tab when you create a PowerCenter expression. If you create an expression that is processed by the data source, the **Functions** tab does not appear when you create the expression.

You can select a parameter from the **Parameters** tab when you create a WHERE clause for a single table or table pair. The **Parameters** tab shows all parameters that are defined in the parameter file that is associated with the single table or table pair. If you do not configure a parameter file for the single table or table pair, the **Parameters** tab does not appear when configure the WHERE clause.

The following figure shows a sample WHERE clause expression in the Expression Editor:

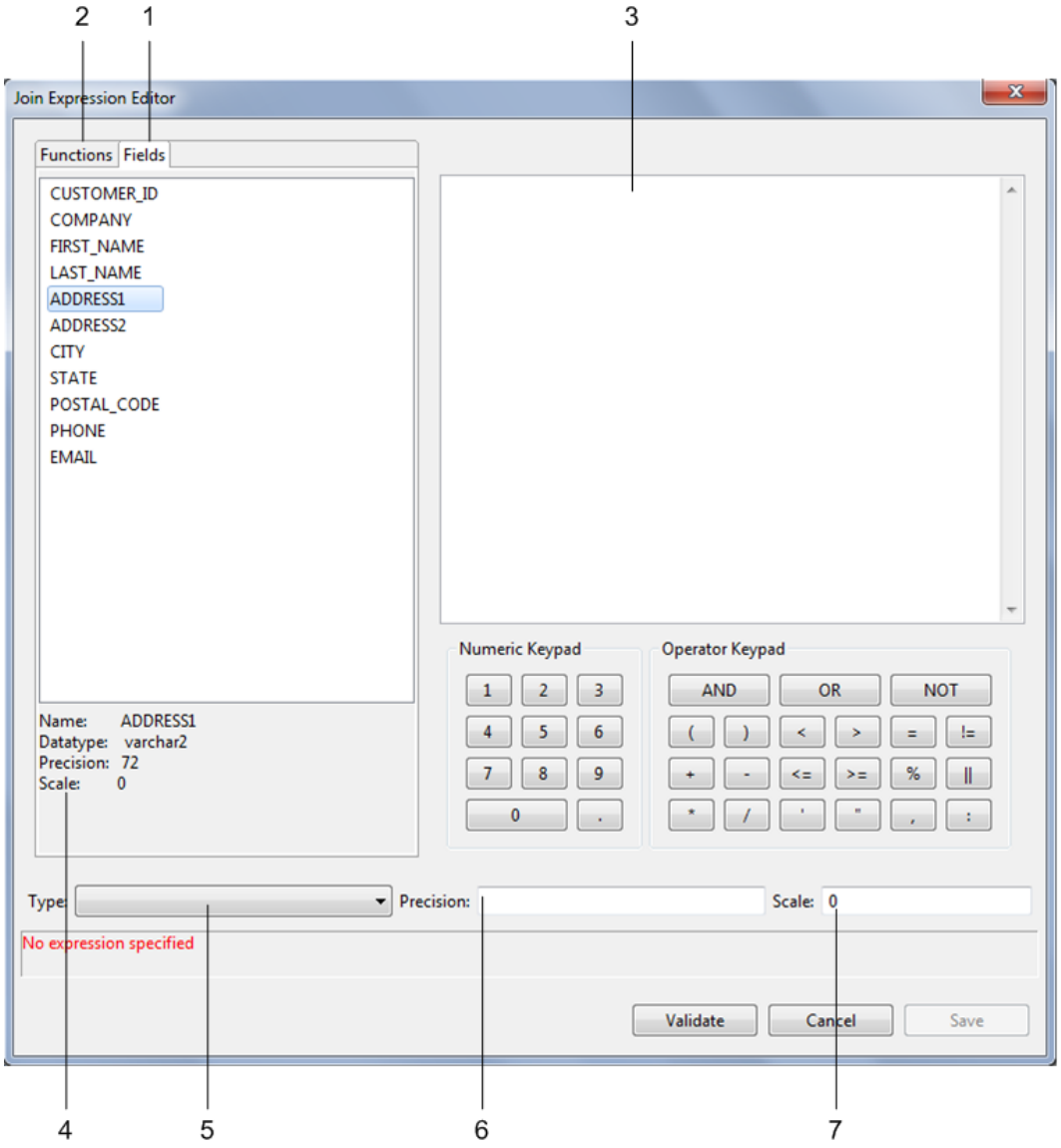


You can click **Validate** to validate a PowerCenter expression. The Expression Editor does not validate datatypes, precisions, scales, or expressions that the data source processes.

Data Type, Precision, and Scale of Field Expressions

In the Expression Editor, you add fields as an expression operand or as a parameter to a PowerCenter function. The field data types and the function return types must be compatible with the data type of the expression result. Additionally, the precision and scale of the expression must be adequate to accommodate the expression result. Incompatibilities can result in a truncated result or test errors.

The following image shows the **Expression Editor** dialog box:



1. Fields tab
2. Functions tab
3. Expression text box
4. Field properties
5. PowerCenter transformation data type for the expression
6. Precision of the PowerCenter transformation data type
7. Scale of the PowerCenter transformation data type

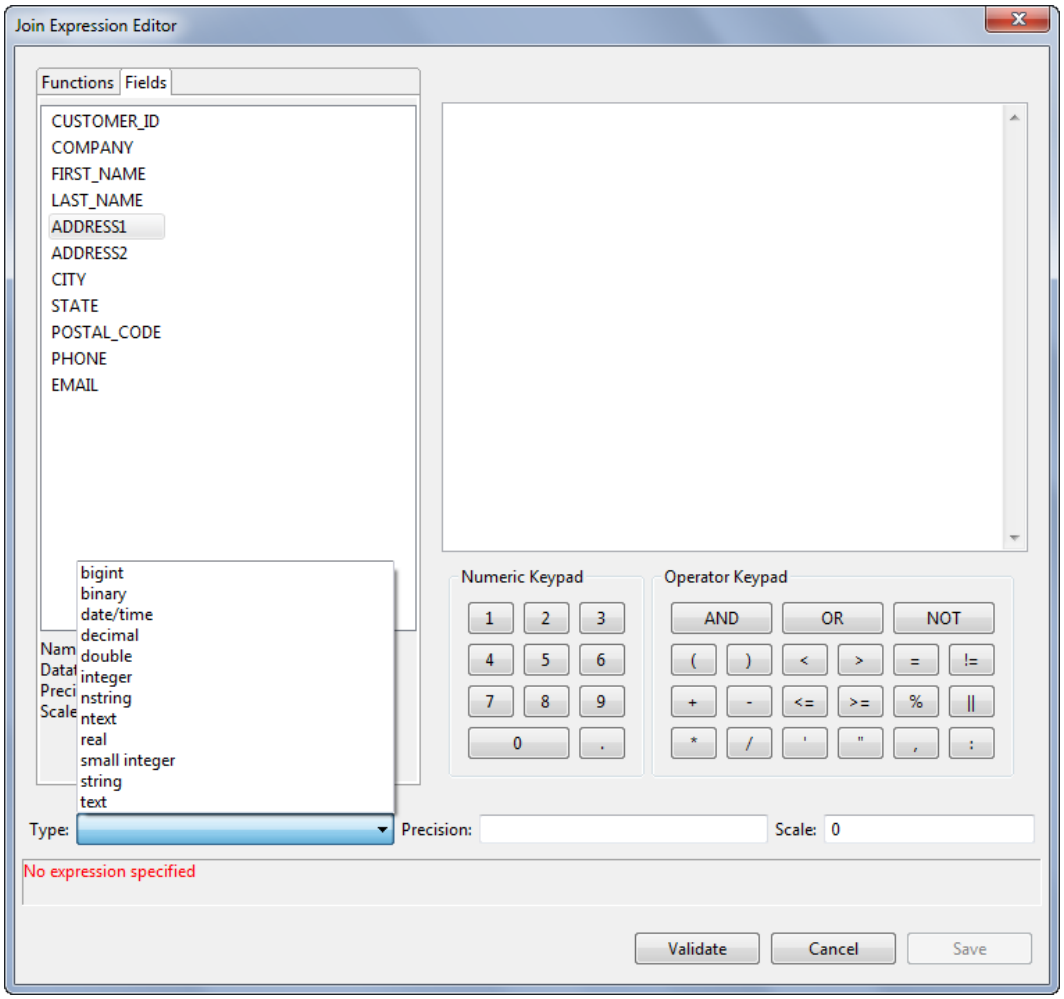
The Expression Editor retrieves fields from the table on which you are running a test. The Expression Editor also displays the properties that the data source associated with the fields when PowerCenter retrieved the fields from the source. The field properties include the data type, precision, and scale of the field.

Precision is the maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. Precision includes scale.

Scale is the maximum number of digits after the decimal point for numeric values. Therefore, the value 11.47 has a precision of 4 and a scale of 2. The string `Informatica` has a precision (or length) of 11.

After Data Validation Option processes an expression, it stores the expression result in a field that has a PowerCenter transformation data type. Data Validation Option also applies the precision and scale of the PowerCenter data type to the expression result. Therefore, when you create an expression, you must know the properties of the fields that you add, and you must set the data type of the expression to a compatible PowerCenter data type. You must also set the precision and scale to values associated with the PowerCenter transformation data type.

The following image shows the PowerCenter transformation data types:



After you create an expression, you can it.

Note: You can validate the expression and not the type, precision, or scale.

Related Topics:

- "Selecting a Valid PowerCenter Transformation Data Type for an Expression" on page 199
- "Properties of PowerCenter Transformation Data Types" on page 201

Creating an Expression

Use the Expression Editor to create an expression that consists of fields, functions, and parameters. You can use the Expression Editor wherever you can use an expression instead of a field name. When you create an expression, you can view the definitions of PowerCenter functions. You can also view the properties of the fields in the table that you are testing.

1. In the dialog box in which you are creating a test, select the **Field is Expression** option.
2. Click the **Expression Editor** button.

   The **Expression Editor** dialog box appears.
3. To add a function to the expression, click the **Functions** tab, and then perform the following tasks:
   a. Expand **All Functions** or expand a function category.
   b. To view the description of a function, click the function.

      The description of the function appears on the **Functions** tab. Use the information in the function description to determine what fields you can use in the function.
   c. To add a function at a particular location in the expression text box, place the insertion point at the location.
   d. On the **Functions** tab, double-click the function.
4. To add a field to the expression, click the **Fields** tab, and then perform the following tasks:
   a. To view the properties of a field, click the field.

      The properties of the field appear on the **Fields** tab.
   b. To add a field at a particular location in the expression text box, place the insertion point at the location.

      To pass a field to a function, place the insertion point within the parentheses that follow the function name, and then add the field.
   c. On the **Functions** tab, double-click the field.
5. To add an operator, perform the following tasks:
   a. In the expression text box, place the insertion point where you want to add the operator.
   b. In the **Operator Keypad**, click the operator.
6. To add a number, perform the following tasks:
   a. In the expression text box, place the insertion point where you want to add the number.
   b. In the **Numeric Keypad**, click the number.
7. In **Type**, select the PowerCenter data type that you want to apply to the result.

   The PowerCenter data type must be compatible with the data types of the expression components.
8. In **Precision**, enter the precision for the PowerCenter data type that you selected for the expression.
9. In **Scale**, enter the scale for the PowerCenter data type that you selected for the expression.

   The default value of scale is 0.
10. To validate the expression, click **Validate**.
11. Click **Save**.

Related Topics:

- "Selecting a Valid PowerCenter Transformation Data Type for an Expression" on page 199
- "Properties of PowerCenter Transformation Data Types" on page 201

Expression Syntax

The syntax of an expression varies based on the component that processes the expression.

By default, the PowerCenter Integration Service processes all expressions. You can use the PowerCenter transformation language to create PowerCenter expressions. Enter the correct case in the WHERE clause because the PowerCenter Integration Service is case sensitive. You can validate PowerCenter expressions, including expressions with parameters. If the PowerCenter syntax is not valid, a mapping installation error occurs.

You can configure the data source to process WHERE clauses. If the data source processes a WHERE clause, you must use the syntax of the data source.

For more information about PowerCenter transformation language, see the *PowerCenter Transformation Language Reference*.

Functions

You can include PowerCenter transformation language functions in a PowerCenter expression.

Data Validation Option does not support the following transformation language functions in PowerCenter expressions:

- LOOKUP
- Variable
- Custom

Note: You cannot use PowerCenter functions if the data source processes the expression.

Operators

The operator defines how to compare values in a test. For example, you can use an equality operator to test whether the salaries in two columns are the same.

For a table pair, the operator defines how to compare each value in Field A with the corresponding value for Field B. For a single-table constraint, the operator defines how to compare the each value in the field with the constraint value.

The operators that you can choose vary based on the type of test and whether the test is for a table pair or single table. An error appears when you try to create a test that contains an operator that is not supported.

Parameters

You can use parameters in the WHERE clause of a table pair or single table. Use parameters in the WHERE clause to dynamically change the WHERE clause after each test run.

When you run a test on a single-table constraint or table pair that has a parameterized WHERE clause, Data Validation Option looks up the value of the parameter from the parameter file and passes the WHERE clause filter to the PowerCenter Integration Service.

To look up the parameter value, the Data Validation Client or the DVOCmd command line that runs the tests must have access to the parameter file.

Rules and Guidelines for Parameters

Certain rules and guidelines apply when you add a parameter to the WHERE clause of a single-table or table-pair.

Use the following rules and guidelines when you add a parameter to a WHERE clause:

- You must add the parameters to the parameter file before you can add the parameters to a table pair or single table definition.
- The parameters in the parameter file must be in the format `$$<parameter name>=<value>`.
- Place the parameter file in a location that is accessible by the Data Validation Client or DVOCmd that runs the tests.
- If you place the parameter file on a UNIX machine, the Data Validation Client on Windows cannot access the parameter file. In this case, install DVOCmd on the UNIX machine and verify that DVOCmd has access to the parameter file.
- When you specify the location of the parameter file in the Data Validation Client, enter the parameter file location as a network path. For example, if the Informatica services runs on Windows, use the following syntax: `\\<Host name of the Informatica services machine>\<shared folder path>\<parameter file name>`
- You can use a parameter file for one or more table pairs and single tables.

Using a Parameter in a WHERE Clause

You can use a parameter in the WHERE clause to dynamically change the WHERE clause after each test run.

1. Create the parameter file and add the parameters.
2. Place the parameter file in a location that is accessible by the Data Validation Client or the DVOCmd command line that runs the tests that require the parameters.
3. If you place the parameter file on a UNIX machine, install DVOCmd on the UNIX machine and verify that DVOCmd has access to the parameter file.
4. On the Advanced tab of a table pair or single table definition, specify the location and name of the parameter file.
5. On the Advanced tab of a table pair or single table definition, specify the name, datatype, scale, and precision of each parameter.

Use Case: Performing Incremental Validation

You can use parameters to perform incremental validation. Incremental validation is when you validate a subset of the records that are loaded into a target.

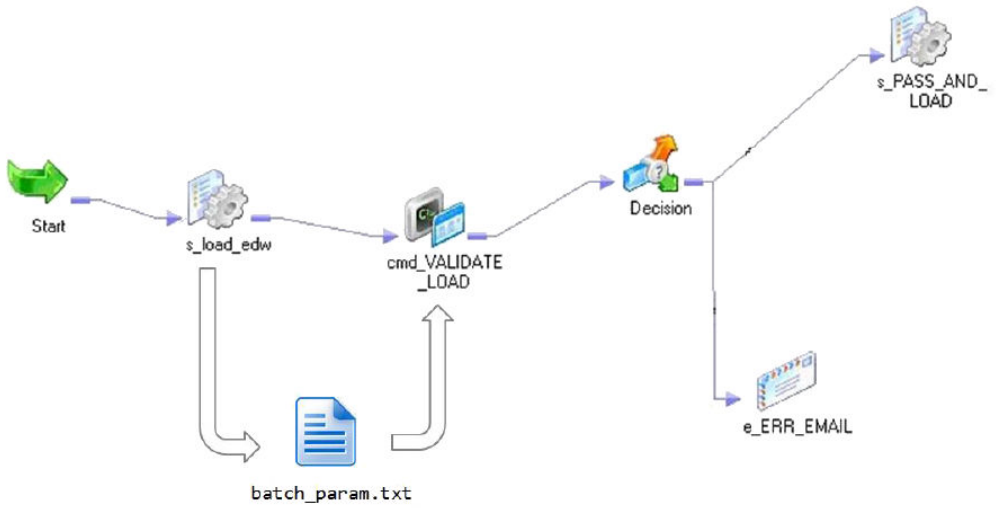
Every day, you run a PowerCenter session that loads approximately 50,000 records into a target. The target currently has 300 million records. You want to create a table pair and tests to validate that all records are loaded into the target every day. However, you do not want to validate the source records against the 300 million target records.

To increase Data Validation Option performance, you perform incremental validation. Each time you run the PowerCenter session, you validate the source records with the latest records that are loaded into the target. You use a $$batchID parameter in the WHERE clause of the table pair to dynamically filter the target records based on the batch ID of the latest load. The PowerCenter session generates a unique batch ID for all records in each load and writes the batch ID of the latest load to the Data Validation Option parameter file.

You perform the following tasks to implement incremental validation:

1. Create the PowerCenter session that loads the source records into the target and populates the BATCH_ID target column with the batch ID of each record.
2. Add a post-session command to the PowerCenter session to write the current batch ID to the parameter file.
3. Create a PowerCenter workflow and add the PowerCenter session to the workflow.
4. Add a Command task to the PowerCenter workflow to run the table pair tests from the DVOCmd command line after the load completes.
5. Add a Decision task and additional logic to the PowerCenter workflow to determine what to do based on the results of the table pair tests. If all tests pass, you configure the workflow to run the s_PASS_AND_LOAD session. Otherwise, the workflow runs the s_FAIL_AND_RECOVER task, and then sends an email notification.
6. Place the parameter file in a location that is accessible by the Data Validation Client that runs the tests.
7. Specify the location of the parameter file in the table pair definition in the Data Validation Client.
8. Add the following WHERE clause in the table pair definition in the Data Validation Client: `BATCH_ID=$$batchID`. `BATCH_ID` is the column in the target table that stores the batch ID of each record.

The following figure shows the PowerCenter workflow:



Test Logic Processing in the Data Source

To increase performance, you can configure a single table or table pair so that the data source processes some of the test logic. The data source can process the WHERE clause, sorting, aggregation, and data sampling.

Data Validation Option generates a PowerCenter mapping that contains the test logic. By default, when you run a table-pair test, the PowerCenter Integration Service processes the test logic. In some cases, you can significantly increase test performance if the data source processes the test logic.

The data source can process the following test logic:

- WHERE clause
- Sorting and aggregation
- Data sampling

Note: The load on the data source increases when it processes test logic. Verify that the net gain is beneficial when the data source processes the test logic.

Where Clause Processing

A WHERE clause filters the records processed by all tests in a single table or table pair. The PowerCenter Integration Service and some data sources can process WHERE clauses.

A relational, Salesforce, SAP, or SAS data source can process the WHERE clause. Configure the data source to process the WHERE clause to reduce the number of rows that the PowerCenter Integration Service reads when it runs a Data Validation Option test. For example, Table A contains information about all U.S. customers. You want to test data for California customers only. In the table pair definition, you enter the WHERE clause `STATE = 'CA'`.

If the PowerCenter Integration Service processes the WHERE clause, the PowerCenter Integration Service reads all U.S. customers, and then applies a filter to the records that were read. If the data source processes the WHERE clause, the data source filters the customer records before the PowerCenter Integration Service reads the records. The test may run faster because the PowerCenter Integration Service reads a subset of the data.

Rules and Guidelines for WHERE Clause Processing

You can configure a relational, Salesforce, SAP, or SAS data source to process the WHERE clause.

Consider the following rules and guidelines if the data source processes the WHERE clause:

- If a relational data source processes the WHERE clause, the WHERE clause must be a valid SQL statement. If the SQL statement is not valid, a run-time error occurs.
- If a relational data source processes the WHERE clause, you can enter a nested SQL WHERE clause. For example, to apply a filter based on employees who have disciplinary issues, use the following WHERE clause:

  ```
  emp_id IN (SELECT DISTINCT emp_id FROM table_discipline)
  ```
- If a Salesforce data source processes the WHERE clause, the WHERE clause must be a valid SOQL filter condition.
- If an SAP data source processes the WHERE clause, the WHERE clause must be a valid SAP filter condition.
- If a SAS data source processes the WHERE clause, the WHERE clause must be a valid Whereclause Overrule condition.

Sorting and Aggregation Processing

The PowerCenter Integration Service or a relational data source can sort and aggregate records. To sort and aggregate records, the relational data source must also process the WHERE clause.

Enable the data source to sort the records when you run tests on tables that have large volumes of data. To minimize disk input and output, sort the records before the PowerCenter Integration Service reads them.

Enable the data source to aggregate data to reduce the number of rows that the PowerCenter Integration Service reads from the database. For example, you use a COUNT test to compare the number of non null records in two Customer_ID columns. If the database processes the sorting and aggregation, the PowerCenter Integration Service does not have to import all customer records to count them.

Data Sampling Processing

Data sampling is when Data Validation Option runs tests on a sample of data rather than the entire source. The PowerCenter Integration Service or a relational data source can generate the data sample.

By default, the PowerCenter Integration Service generates the data sample. The PowerCenter Integration Service reads all records from the source when it generates the data sample.

To increase run-time performance, you can configure IBM DB2, Microsoft SQL Server, Oracle, and Teradata to generate the data sample. After the database generates the data sample, the PowerCenter Integration Service reads the sample data only.

Expression Examples

The sample field expressions show how to use PowerCenter functions to create expressions.

The following examples show how to use field expressions when comparing column values:

Concatenation example

You want to compare the full names in two tables. One table has a FULL_NAME column where the full names are in uppercase letters. The other table has FIRST_NAME and LAST_NAME columns where the first and last names are in lowercase letters. To compare the full names, you must concatenate the FIRST_NAME and LAST_NAME columns, and convert the lowercase names to uppercase.

To compare the full names, you create a Value test and specify the following expressions for the table columns:

- Field A: `FULL_NAME`
- Expression B: `UPPER(FIRST_NAME || ' ' || LAST_NAME)`

IIF statement example

You want to compare total sales in two sales order transaction tables. You want to analyze USA sales and international sales.

Table A contains the following columns:

Column	Description
SALES_USA	Stores the order amount for a USA sales transaction.
SALES_INTL	Stores the order amount for an international sales transaction.

Table B contains the following columns:

Column	Description
REGION	Stores the region. Valid values are 'USA' or 'INTL.'
SALES	Stores the order amount for the sales transaction.

To compare the total USA sales, you create a SUM test and specify the following expressions for the table columns:

- Field A: `SALES_USA`
- Expression B: `IIF(REGION='USA', SALES, 0)`

To compare the total international sales, you create a SUM test and specify the following expressions for the table fields:

- Field A: `SALES_INTL`
- Expression B: `IIF(REGION='INTL', SALES, 0)`

Reusable Expressions

You can create customized or reusable code or functions that can be used across many different tests or table pairs in Data Validation Option.

You can use the PowerCenter expression language to create the user-defined functions for Data Validation Option tests. You can use the same user-defined functions in multiple tests.

Rules and Guidelines for User-Defined Functions

Use the following guidelines when you create user-defined functions and use them in Data Validation Option:

- User-defined functions are defined in PowerCenter at folder level. They exist in a specific folder in the PowerCenter repository.
- You must define a user-defined function in the target folder where Data Validation Option jobs are run so that Data Validation Option can access the function. In the Data Validation Client, you can view the target folder in the list of target repository properties when you edit the PowerCenter repository.

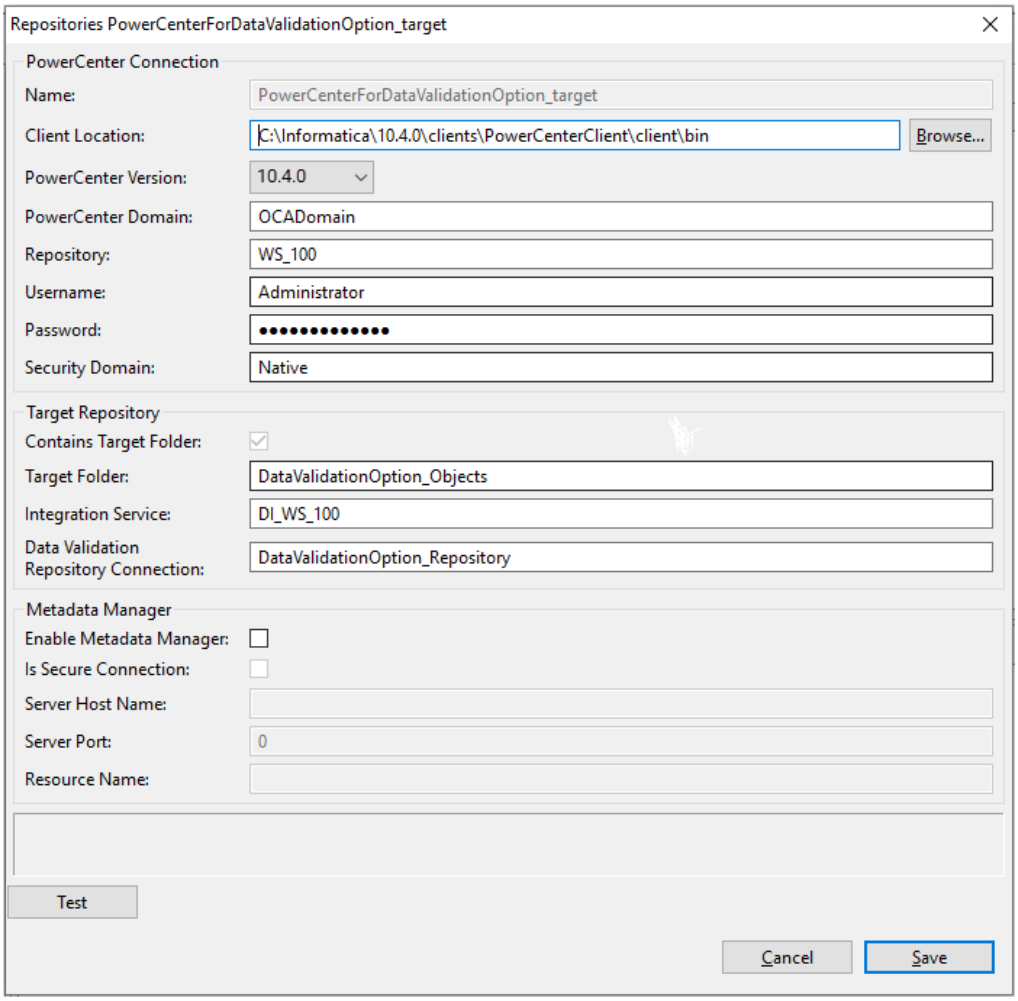
- You cannot validate an expression that uses a user-defined function. The Data Validation Client cannot validate user-defined functions that are defined in the PowerCenter repository. If the expression is a valid PowerCenter expression, it will run correctly.
  If you validate an expression that uses a user-defined function, the Data Validation Client display an error message similar to the following message:

Using PowerCenter User-Defined Functions in Data Validation Option

You can create a user-defined function in PowerCenter and use it in Data Validation Option.

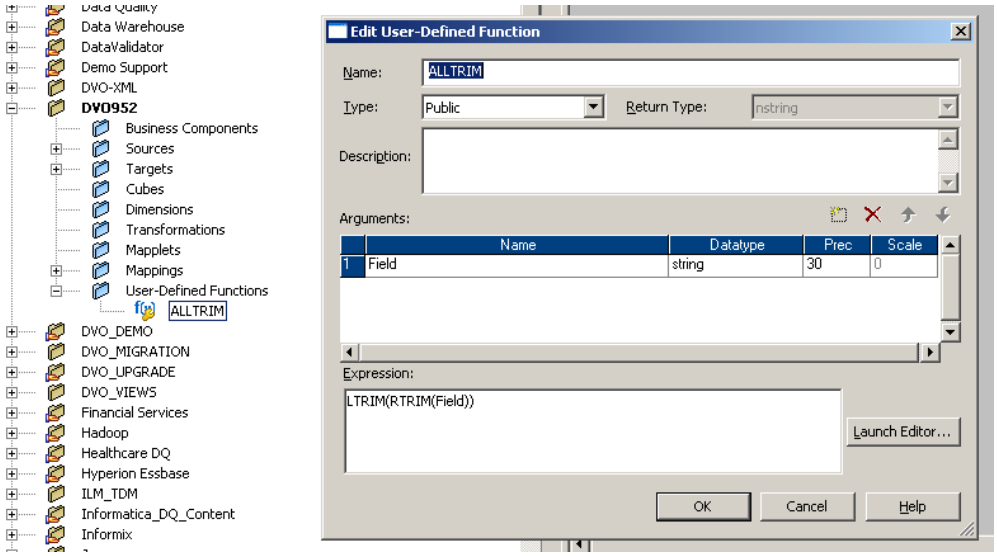
1. In the Data Validation client, right-click the target PowerCenter repository and select **Edit Repository** to view the name of the folder where you can create the user-defined function.

   The following image shows the target folder for the PowerCenter repository:



2. In PowerCenter, go to the target folder of the repository that you use for Data Validation Option.

The following image shows the properties of the user-defined function:



3. Save the user-defined function and disconnect from the target folder.

   Data Validation Option generates and writes objects to the target folder. If the target folder is open in the PowerCenter repository, a locking error occurs.

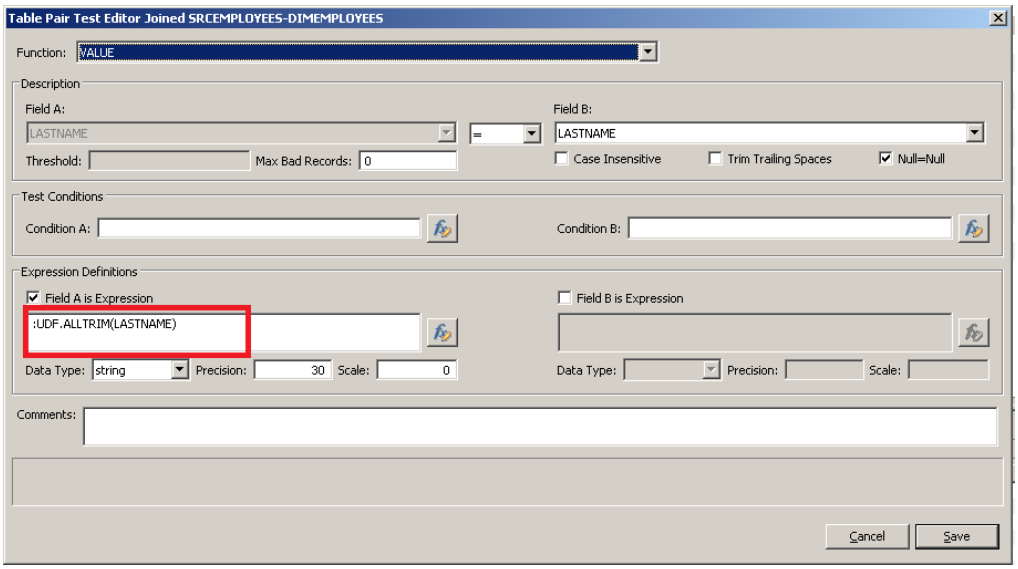
4. In Data Validation Option, go to the location where you want to use the user-defined function.

   You can call the user-defined function from any valid location, such as a WHERE clause, an expression in a value test, or a test condition.

   To call the user-defined function, use the syntax `:UDF.<function name>(<parameters>)`

   For example, if the name of the function is ALLTRIM, use the expression `:UDF.ALLTRIM(FIRSTNAME)`

   The following image shows the user defined function called from an expression in a value test:



5. Ensure the data type, precision, and scale are appropriate for the function and match the return type correctly.

6. Save and run the test.

CHAPTER 7

Table Pairs

This chapter includes the following topics:

Table Pairs Overview

A table pair is a pair of tables on which you can create and run data validation tests. Run data validation tests to compare the tables.

Each table in the table pair must be a PowerCenter source or target, join view, lookup view, or SQL view.

You can use the following types of PowerCenter sources and targets as tables in a table pair:

- Flat file
- Relational table
- Salesforce
- SAP
- SAS
- XML

You can create a single table pair or generate multiple table pairs. Generate table pairs to create table pairs faster.

When you create a table pair, you specify Table A and Table B. Data Validation Option considers Table A as the master table and Table B as the detail table. To increase test performance, specify the largest table as Table B and the smallest table as Table A.

You can configure the table pair to store all error records from each table-pair test. You can also use a parameter in the WHERE clause of a table pair definition to filter the records that are tested. Filter the records to increase test performance.

Basic Properties for a Table Pair

You can view and configure the basic properties for a table pair. The properties vary based on the types of objects you select for the table pair.

Description

Description

Table pair description. By default, Data Validation Option uses "Joined <Table A>-<Table B>" for joined table pairs. It uses "<Table A>-<Table B>" for table pairs that are not joined.

Table A/B

You can configure the following properties for Table A and Table B:

Table A/B

The first or second table in the table pair. Select **Browse** to select a table. You can search for a data source by name or path. You can search for a lookup view, SQL view, join view, or aggregate view by name.

Conn A/B

Connection properties for the table. Select **Edit** to edit the connection properties.

XML Group A

The XML group name for an XML data source. Only child elements of the XML group are available to use in the WHERE clause, table join field, and tests.

Where Clause A/B

The WHERE clause filters the records that are read from the data source. If the PowerCenter Integration Service processes the WHERE clause, enter a valid PowerCenter expression. If a relational, Salesforce, SAP, or SAS data source processes the WHERE clause, enter a WHERE clause that is supported by the data source.

Execute Where Clause in DB

If the data source is relational, Salesforce, SAP, or SAS, enable if you want the data source to process the WHERE clause. You must select this option to select the **Where Clause, Sorting and Aggregation in DB** optimization level.

Optimization Level

Controls the test logic processed by the PowerCenter Integration Service and the data source. Select one of the following options:

- Default. The PowerCenter Integration Service sorts the input data and processes all test logic.

- WHERE clause, Sorting, and Aggregation in DB. If the data source is a relational data source, the database processes the WHERE clause, sorts records for joins, and aggregates data for aggregate tests. The PowerCenter Integration Service processes all remaining test logic that the data source does not process.
- Already Sorted Input. Indicates that the records in the data source are already sorted. If the records in the data source are not sorted, tests might fail.

Table Join

Join Field A/B

Join condition for the table pair. Select **Expression** to insert an expression for a join field. If you enter an expression for a join field, click **Validate** to validate the expression.

Note: If you enter an expression for a join field, you cannot save the table pair until you enter a valid expression.

External ID

External ID

Identifier for the table pair that you can use when you run Data Validation Option tests from the command line.

Related Topics:

- "XML Groups" on page 39

Advanced Properties for a Table Pair

You can configure bad records, a parameter file, sampling, and caching for a table pair. The properties vary based on the types of objects you select for the table pair.

Bad Records

You can configure the following properties for bad records:

Save All Bad Records for Test Execution

Saves all bad records for value, outer value, and set tests for table pairs.

Write Bad Records to:

The location where Data Validation Option writes bad records. You can select one of the following options:

- Flat File. Data Validation Option writes bad records to the specified flat file.
- DVO Schema. Data Validation Option writes bad records to the Data Validation Option repository.

File Name

The location and name of the flat file to which Data Validation Option writes bad records. Default location is `$INFA_HOME/server/infa_shared/BadFiles/<single table name /table pair name>/<file name>`.

Parameters

Parameter File

Name of the parameter file. Click **Add** to add a parameter file. Click **Delete** to delete the selected parameter file.

Data Sampling

You can configure the following properties for data sampling:

Enable Data Sampling

Indicator that determines if data sampling is enabled.

Apply Sampling on:

The table on which you want to apply sampling.

Percentage of Rows

Percentage of rows that you want to sample. Minimum value is .000001. Maximum value is less than 100.

Seed Value

The starting value used to generate a random number for sampling. You can enter a seed value for the following platforms:

- PowerCenter. Minimum is 0. Maximum is 4294967295.
- IBM DB2. Minimum is 1 - 10^{31}. Maximum is 10^{31} - 1.
- Oracle. Minimum is -2,147,483,648. Maximum is 2,147,483,647.

Use Native Sampling

The database performs the sampling. If you do not enable this option, PowerCenter performs the sampling.

Cache Size

Cache Size

Total cache used to run a PowerCenter session associated with the table pair. By default, the units are bytes. Append KB, MB, or GB to the value to specify other units. For example, you can specify 1048576, 1024 KB, or 1 MB. Select **Automatic** to enable PowerCenter to compute the cache size.

Rules and Guidelines for WHERE Clauses in a Table Pair

Certain rules and guidelines apply when you add a WHERE clause to a table pair.

Use the following rules and guidelines when you add a WHERE clause:

- You can configure a WHERE clause for each table or file in a table pair. If you configure a WHERE clause such that one table in a table pair has signicantly more rows than the other, the OUTER_VALUE and aggregate tests are affected. You can use different WHERE clauses in a table pair to equalize data sets. For example, you can add a WHERE clause on a production environment table has three years of data when the development environment table only has two weeks.

- Because the filter condition you enter in the WHERE clause applies to all tests in the table pair, Data Validation Option applies the WHERE clause before it joins the tables. This can improve performance when the WHERE clause filters a large percentage of rows from the source table because the PowerCenter Integration Service processes fewer rows later in the mapping. If you want to enter a condition that filters a small percentage of rows, or you want to apply different filters for different tests, you can enter a filter condition in the Table Pair Test Editor dialog box.
- For an XML data source, you can include only child elements of the selected XML group in the WHERE clause.

Table Joins

You can join or unjoin a table pair. Join tables if you want to run a VALUE or an OUTER_VALUE test. Data Validation Option ignores joins for all set and aggregate tests.

To create a joined table pair, define one or more conditions based on equality between the tables or files. For example, if both tables in a table pair contain employee ID numbers, you can select EMPLOYEE_ID as the join field for one table and EMP_ID as the join field for the other table. Data Validation Option performs an inner equijoin based on the matching ID numbers.

You can create a join with one set of fields or with multiple sets of fields from each table if the table requires this to produce unique records. Note that additional sets of fields increase the time necessary to join two sources. The order of the fields in the join condition can also impact the performance of Data Validation Option tests. If you use multiple sets of fields in the join condition, Data Validation Option compares the ports in the order you specify.

When you create a join, you can select a field from each table or enter an expression. Enter an expression to join tables with key fields that are not identical. For example, you have two customer tables that use different cases for the LAST_NAME field. Enter the following expression for one of the join fields: `lower(LAST_NAME)`.

When you enter an expression for one or both join fields, you must specify the datatype, precision, and scale of the result. The datatype, precision, and scale of both join fields must be compatible. The expression must use valid PowerCenter expression syntax. Data Validation Option does not check the expression syntax.

Rules and Guidelines for Table Joins

Certain rules and guidelines apply when you create a table join in a table pair.

Use the following rules and guidelines when you create a table join:

- Join tables if you want to run a VALUE or an OUTER_VALUE test. Data Validation Option ignores joins for all set and aggregate tests.
- The fields included in a join condition must have compatible datatypes, precisions, and scales. For example, you can join an INT field to a DECIMAL field, but not to a DATETIME or VARCHAR field.
- Data Validation Option supports joins on fields with the following datatypes: datetime, numeric, and string. Joins are not allowed on binary/other datatypes.
- Data Validation Option joins tables in a table pair based on an inner equijoin by default.
- For an XML data source, only child elements of the selected XML group appear as possible join fields.
- You can create multiple join conditions in a table join for a joined table pair.

- If you specify a file list connection and the fields in the join condition are not unique among all files in the file list, the join fails.

Table Join Optimization

When you include a large table in a joined table pair, you can increase performance of the table-pair tests.

Data Validation Option uses joins in VALUE and OUTER_VALUE tests. To run VALUE and OUTER_VALUE tests on a table pair, you must join the tables based on the related keys in each table.

By default, Data Validation Option joins the tables with an inner equijoin and sorts the rows in the table. The join condition uses the following `WHERE` clause syntax:

```
Table A.column_name = Table B.column_name
```

If one of the tables in a table pair contains a large volume of data compared to the other table, you can increase test performance by designating the larger table as the detail source and the smaller table as the master source. Select the smaller table as Table A and the larger table as Table B. When the PowerCenter Integration Service determines whether to join two records, it compares each row of the master source against the detail source.

To increase performance of a table pair with two large tables, sort the input. When you run tests on tables with sorted records, the PowerCenter Integration Service minimizes disk input and output.

To further increase performance for large relational tables, configure the table pair so that the database sorts the records.

Bad Records Configuration for a Table Pair

You can store up to 16,000,000 bad records per test to perform advanced analysis and up to 1000 bad records for reporting.

For reporting, you can set the maximum number of bad records in the mapping properties in the Preferences dialog box. The default number of bad records is 100. You can set up to 1000 bad records for reporting.

For advanced analysis, you can set the maximum number of bad records for detailed analysis and the file delimiter. You set the properties in the Detailed Error Rows Analysis section in the mapping properties. The default number of bad records is 5000. You can set up to 16,000,000 bad records.

You can store all the bad records from the tests for a table pair or single table. Select Save all bad records for test execution on the Advanced tab when you create or edit a table pair or single table.

Data Validation Option stores the following tests for table pairs:

- Value
- Outer Value
- Set

You must select whether you want to store the bad records in a flat file or a table in the Data Validation Option schema. If you store bad records in a flat file, you can optionally enter the name of the file. Data Validation Option appends test information to the name and retains the file extension.

Note: If you modify the file delimiter in the preferences file, run the InstallTests command with the forceInstall option for the existing table pairs or single tables that you already ran. You can also edit and save

the table pair or single table from the Data Validation Client before you run the test. If you modify the bad records value, you need not reinstall the tests.

Bad Records in Flat File

If you configure to store bad records in flat file, Data Validation Option creates the flat file in the machine that runs Informatica services.

The flat files that Data Validation Option generates after running the tests are stored in the following folder:`<PowerCenter installation directory>\server\infa_shared\TgtFiles`

You can modify the folder from the Administrator tool. Edit the $PMTargetFileDir property for the PowerCenter Integration Service.

Data Validation Option generates a folder for each of the table pair or single table. The name of the table pair or single table is in the following format: `TablePairName_TestRunID or SingleTableName_TestRunID`

Data Validation Option creates flat files for each test inside the folder. The name of the flat file is in the following format: `<user defined file name>_ TestCaseType_TestCaseColumn A_TestCaseColumnB_TestCaseIndex.<user defined file extension>`

You can get the Test Case Index from the Properties tab and the Test Run ID from the results summary of the test in the detail area. You can get the Table Pair ID/Single Table ID from the Table Pair or Single Table properties.

For example, you enter the file name as BAD_ROWS.txt when you configure the table pair or single table and you run an outer value test on fields FIRSTNAME and FIRSTNAME. The test-case index is 1 and the test-fields are expressions. The bad records file after you run the test is of the format, BAD_ROWS_OUTER_VALUE_ExprA_ExprB_1.txt.

Data Validation Option supports all the file delimiters that PowerCenter supports. When you enter non-printable characters as delimiters, you should enter the corresponding delimiter code in PowerCenter. When you import these files in PowerCenter, you have to manually create the different data fields since the code appears in the place of delimiter in the bad records file.

Caution: Data Validation Option uses comma as a delimiter if there a multiple primary keys. You must not use comma as a file delimiter.

When you run the tests for the table pair or single table, Data Validation Option stores the details of the bad records along with the following format:

```
Table Pair Name
Table A Name
Table A Connection
Table B Name
Table B Connection
Test Definition
Test Run Time
Test Run By User
Key A[],Result A[],Key B[],Result B[]
```

If the tests pass, Data Validation Option still creates a flat file without any bad records information.

Bad Records in Database Schema Mode

If you choose to save detail bad records in the Data Validation Option schema, the bad records are written into the detail tables.

The following table describes the tables to which Data Validation Option writes the bad records based on the type of test:

Test Type	Table	Columns
Value, Outer Value, Value Constraint	ALL_VALUE_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, KEY_A, VALUE_A, KEY_B, VALUE_B
Unique Constraint	ALL_UNIQUE_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, VALUE
Set	ALL_SET_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, VALUE_A, VALUE_B

Note: You must ensure that the database table has enough table space to hold all the bad records.

The test details of all the tests that you ran in Data Validation Option is available in the TEST_CASE table. The test installation details of the tests are available in the TEST_INSTALLATION table. You can obtain the TEST_ID of a test from the TEST_INSTALLATION table. You need TEST_ID of a test to query the complete details of a test from the TEST_CASE table.

You can get the Test Case Index from the Properties tab and the Test Run ID from the results summary of the test in the detail area. You can get the Table Pair ID/Table ID from the Table Pair or Single Table properties.

For example, you ran a table pair with a value test and outer value test.

The following SQL query is a sample query to retrieve the information of bad records of a test with Test Case Index as 5.

```
select ALL_VALUE_RESULT_DETAIL.*,TEST_CASE.* from ALL_VALUE_RESULT_DETAIL , TEST_RUN,
TEST_INSTALLATION, TEST_CASE
where
ALL_VALUE_RESULT_DETAIL.TEST_RUN_ID=TEST_RUN.TEST_RUN_ID
and TEST_RUN.TEST_INSTALLATION_ID=TEST_INSTALLATION.TEST_INSTALLATION_ID
and TEST_INSTALLATION.TABLE_PAIR_ID=TEST_CASE.TEST_ID
and ALL_VALUE_RESULT_DETAIL.TEST_CASE_INDEX=TEST_CASE.TEST_CASE_INDEX
and ALL_VALUE_RESULT_DETAIL.TEST_RUN_ID=220
```

Data Sampling for a Table Pair

You can use data sampling to run tests on a subset of a data set. You might use data sampling when the data set is large. You can perform data sampling on table pairs and single tables.

You can perform data sampling on one table in a table pair. When you run a table-pair test, Data Validation Option runs the test based on a percentage of the rows in the sampled table and all of the rows in the other table.

You can use a seed value to repeat the same sample data set in multiple runs of a test. Data Validation uses the seed value as the starting value to generate a random number. If you do not enter a seed value, Data Validation Option generates a random seed value for each test run. You might use a seed value to replicate a Data Validation Option test.

By default, PowerCenter performs the sampling. If you sample data from IBM DB2, Microsoft SQL Server, Oracle, or Teradata, you can perform native sampling in the database. Push sampling to the database to increase performance.

If you add the WHERE clause and enable sampling, the order of operations depend on where you perform sampling and execute the WHERE clause. Generally, PowerCenter and the database performs sampling before executing the WHERE clause. However, when you configure the database to execute the WHERE clause and PowerCenter to perform sampling, the database executes the WHERE clause before PowerCenter performs the sampling.

Note: Because data sampling reduces the number of rows in one table of the table pair, some tests might have different results based on whether you enable data sampling. For example, a Count test might fail after you enable data sampling on one table in a table pair because Data Validation Option processes less rows in one of the tables in the table pair. An Outer Value test might fail after you enable data sampling because the test encounters orphan rows when it performs a full outer join between the tables in the table pair.

Rules and Guidelines for Data Sampling

Certain rules and guidelines apply when you perform data sampling on a table pair.

Use the following rules and guidelines when you perform data sampling on a table pair:

- The sample percentage that you specify may differ from the actual sample percentage during a test run. The sample percentage that you specify represents the chance that each data is included in the sample.
- You cannot use native sampling if the underlying Oracle tables or views from which you are sampling do not have unique primary keys. Instead, use PowerCenter sampling.
- If you run sampling on a table pair, you select one table to sample. In a master-detail relationship, sample the detail table because it is the bigger table. When you sample the bigger table, the joiner processes a smaller set of data. Table A is the master table and table B is the detail table.
- Some tests might have different results based on whether you enable data sampling.

PowerCenter Cache for a Table Pair

You can configure the total cache memory for a table pair. If you enable caching, the PowerCenter Integration Service processes the Lookup, Joiner, Aggregator, and Sorter transformations faster. The PowerCenter Integration Service creates in-memory index and data caches to temporarily store data for each transformation while processing the transformation.

If you enable caching and run a test for a table pair, the PowerCenter Integration Service divides the specified cache size among all Aggregator, Joiner, Lookup, and Sorter transformations in the underlying PowerCenter session. The PowerCenter Integration Service allocates cache memory based on the transformation requirements, such as the number of rows processed, number and types of tests, and precision and scale of the table columns.

For optimal session performance, configure the cache size so that the PowerCenter Integration Service processes each transformation in memory without paging to disk. Session performance decreases when the Integration Service pages to disk. To determine the required cache size for each transformation, review the cache size specified in the session log after the session completes.

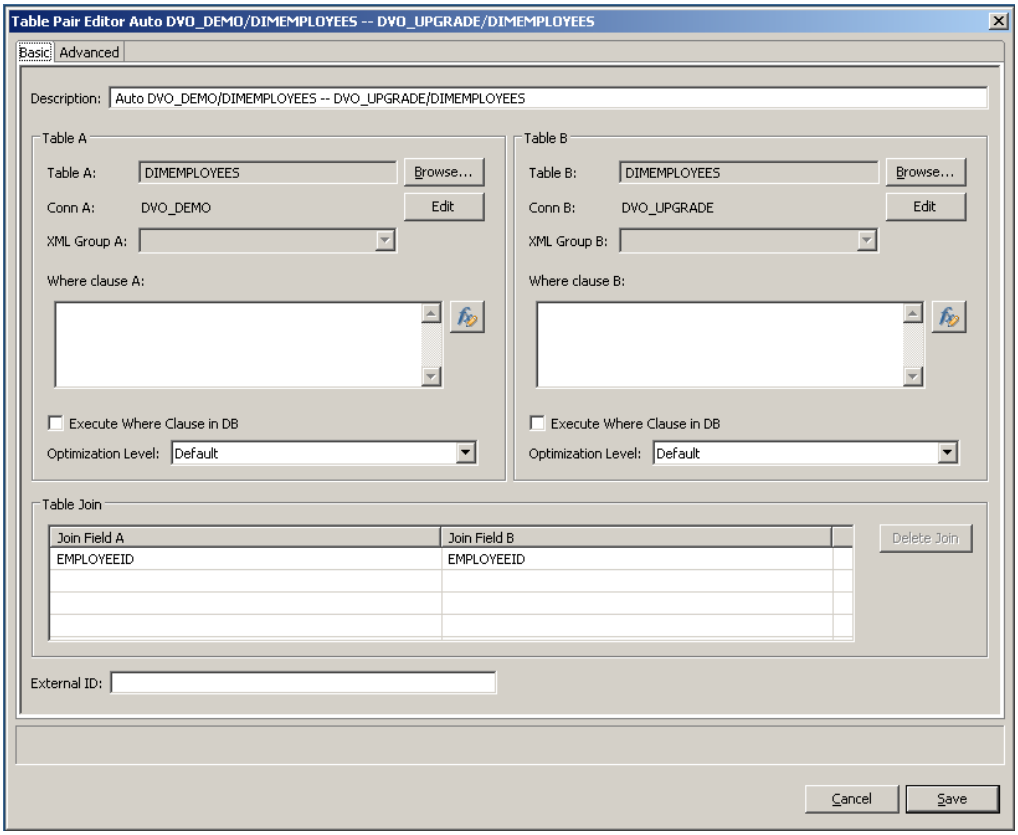
Adding a Table Pair

You can create a table pair from the file menu or from the shortcut in the menu bar.

1. Select the folder to which you want to add the table pair.
2. Right-click the folder and select **Add Table Pair**.

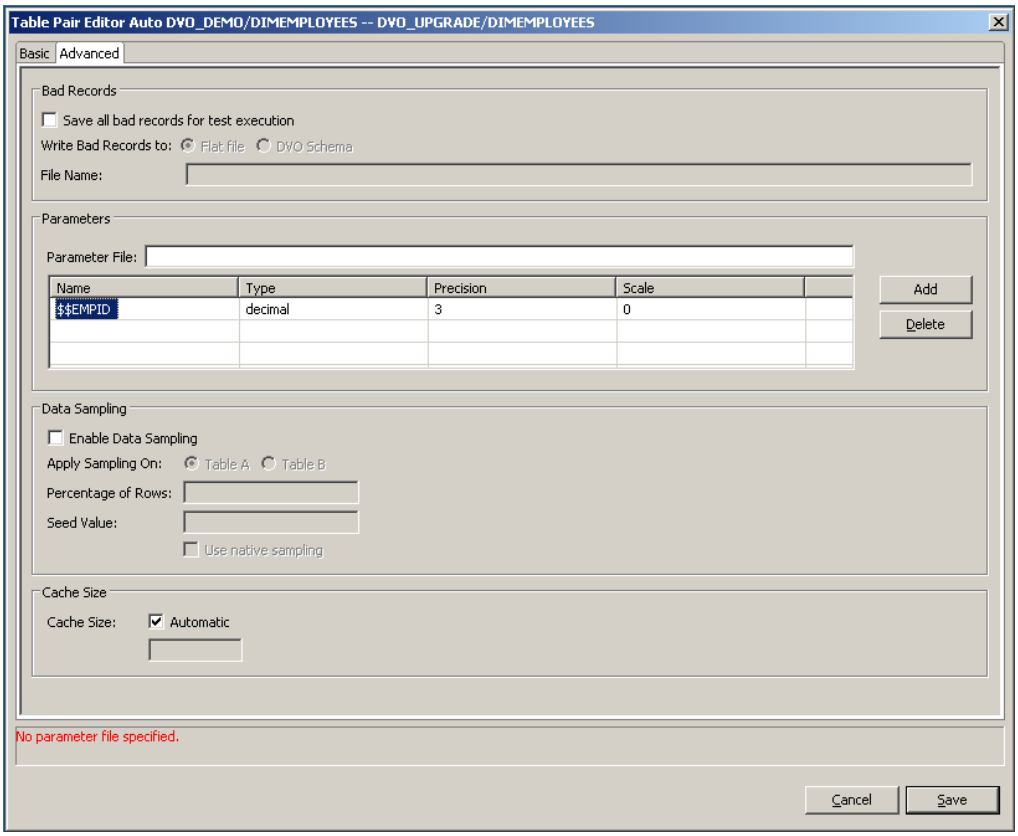
   The **Table Pair Editor** window appears.
3. On the **Basic** tab, enter the basic properties of the table pair.

   The following figure shows the **Basic** tab for a relational connection:



4. Enter values for the basic properties on the **Basic** tab.
5. Click **Edit** to configure the connection properties for a table.
6. Click the **Advanced** tab and enter values for the advanced properties.

The following figure shows the **Advanced** tab:



7. Click **Save** to save the table pair.

Table Pair Generation from Table Comparisons

A table comparison compares tables in two different folders. Run a table comparison to generate table pairs and tests for each table pair. Use the test results to determine if the data is the same in both tables or to determine what the differences are.

For example, run a table comparison after you upgrade PowerCenter or migrate from a PowerCenter development environment to a PowerCenter production environment. To verify that an upgrade is successful, take a specified set of input data and run it through the pre-upgrade mappings/workflows and then the post upgrade mappings/workflows. Use Data Validation Option to run a full regression test by comparing the pre-upgrade target tables to the post-upgrade target tables to identify any potential errors in the data. If the upgrade process completes successfully, the two data sets should be identical. Similarly, to verify a migration is successful, use Data Validation Option to compare the pre-migration data against the post migration data, incorporating any expected or required migration data changes into tests in Table Pairs.

You can run a table comparison on all data source types. If you compare database tables, you must select a default connection for the tables. If you compare flat files, you must enter a default directory for the files. If a table or file requires a different connection, update the connection for the applicable table pair after Data Validation Option generates the table pairs and tests.

When you run a table comparison, by default, Data Validation Option generates table pairs for tables that have matching names. To generate table pairs for tables that have different names, define the table pairs in a spreadsheet and import the table pairs from the spreadsheet.

You might also use spreadsheets to define table pairs for other reasons. If you defined PowerCenter ETL rules in spreadsheets, you can reuse the ETL rules that map PowerCenter sources to targets. You might also use spreadsheets to define the table pair definitions when you work offline or are not a Data Validation Option user. A Data Validation Option user can import the table pairs from the spreadsheet.

Data Validation Option generates value and outer value tests for a table pair based on the primary keys of the tables. You must specify the primary keys for both tables in the table pair definition. By default, you can select one column as the primary key for a table. If a relational table has a composite primary key or a flat file does not have a column header, you can create a primary key file that specifies the primary key for the relational table or flat file.

When Data Validation Option generates table pairs, it also generates tests for each generated table pair.

Tests Generated from Table Comparisons

When you run a table comparison, Data Validation Option generates count and value tests for all columns that have compatible datatypes and matching columns. When you create a table comparison, you configure whether to match columns based on the column name or column position.

When you run a table comparison, Data Validation Option generates OUTER_VALUE, VALUE, and COUNT_ROWS tests.

Note: Data Validation Option does not generate OUTER_VALUE and VALUE tests for binary columns.

Related Topics:

- " Column Comparison by Position" on page 93

Primary Key File

To use a table comparison to generate value and outer value tests for a table pair, you must specify the primary keys for both tables. If a relational table has a composite primary key or if a flat file does not have a column header, you can create a primary key file that specifies the primary key for the relational table or flat file.

In the Data Validation Client, you can select one column as the primary key for each table in a table pair. If the primary key of a relational data source is a composite primary key, you must create a primary key file that specifies all columns that comprise the primary key.

If a flat file does not contain column headers, then you must create a primary key file that specifies the column that is the primary key for the flat file. If the primary key is a composite key, then you can specify all columns that comprise the primary key.

The file must contain the name of the flat file or table and the primary key separated by comma.

For example:

```
flatfile_dictionary_10rows,FIELD1
```

To specify multiple columns of a composite primary key, enter each column on a new line.

For example:

```
flatfile_dictionary_10rows,FIELD1
flatfile_dictionary_10rows,FIELD2
```

```
flatfile_dictionary_10rows_str,FIELD1
```

Spreadsheet Requirements for Table Pair Definitions

When you run a table comparison, you can generate table pairs based on table-pair definitions in a spreadsheet. Define table pairs in a spreadsheet to generate table pairs for tables that have different names.

Consider the following requirements for the spreadsheet:

- Create the spreadsheet based on the spreadsheet template in the following Data Validation Option installation directory:

  ```
  <Data Validation Option installation directory>\samples\CompareTables-template.xlsx
  ```

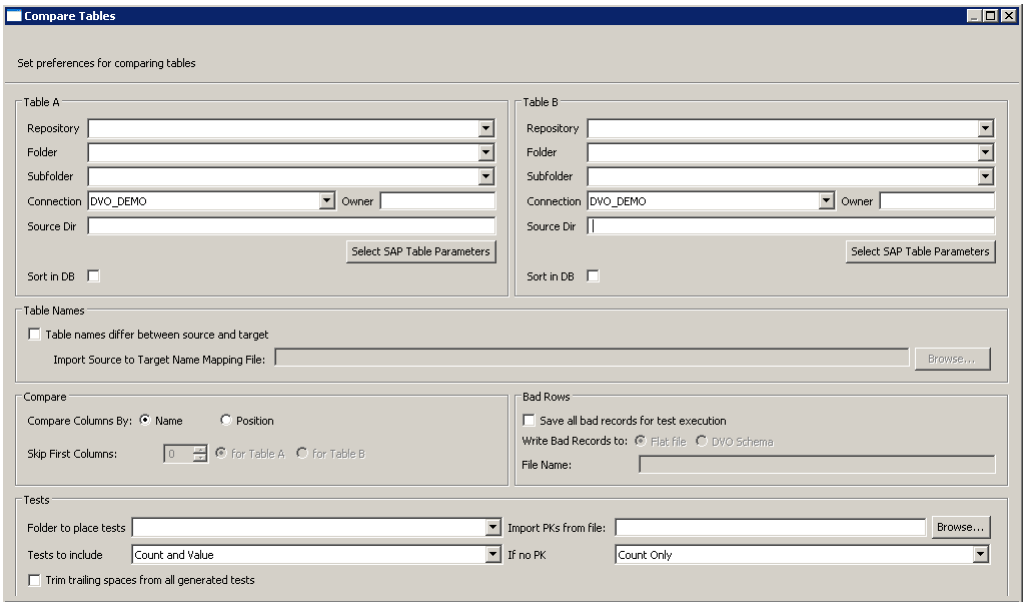
- Define table pairs in the first worksheet. Data Validation Option ignores other worksheets in the Excel workbook.
- The Excel file must have a .xls or .xlsx file extension.
- You must enter values in the LEFT_TABLE and RIGHT_TABLE columns.
- The column names are not case sensitive.
- If you create a joined table pair, you must specify a column name in the Join_A and Join_B columns. To enter an expression for the join field, generate the table pair and then edit the join field in the table pair.
- You cannot specify an XML file as a table in the spreadsheet.
- If you import a table pair from a spreadsheet and the table pair is based on the same tables as an existing table pair, Data Validation Option creates a duplicate version of the table pair with the same name.

Generating Table Pairs and Tests

Run a table comparison to generate table pairs between tables in two specified folders. Data Validation Option also generates count and value tests for each table pair.

1. In Data Validation Option, click **Action** > **Compare Tables**.

   The **Compare Tables** dialog box appears.



2. Select the repositories that contain the tables that you want to compare.

3. Select the folders that contain the tables that you want to compare.
4. Select **Sources** or **Targets** for each subfolder.
5. Select the default connection for each folder.
6. If the connection is for IBM DB2, enter the default database owner name in the **Owner** field.
7. If connection is for flat files, in the **Source Dir** field, enter the default directory that contains the files.
8. If the connection is for SAP, click **Select SAP Table Parameters** and configure the default SAP source parameters.
9. Select **Sort in DB** to push the sorting logic for joins in table pairs to the database.
10. You can select whether to save all the bad records.
11. If you choose to save all the bad records, select whether to save the bad records in a flat file or in the Data Validation Option schema.
12. To generate table pairs for tables that have different names, select **Table names differ between source and target**, and then click **Browse** to find the spreadsheet file that contains the table pair definitions.
13. To generate table pairs for tables that have the same name, select whether to compare columns by name or position.

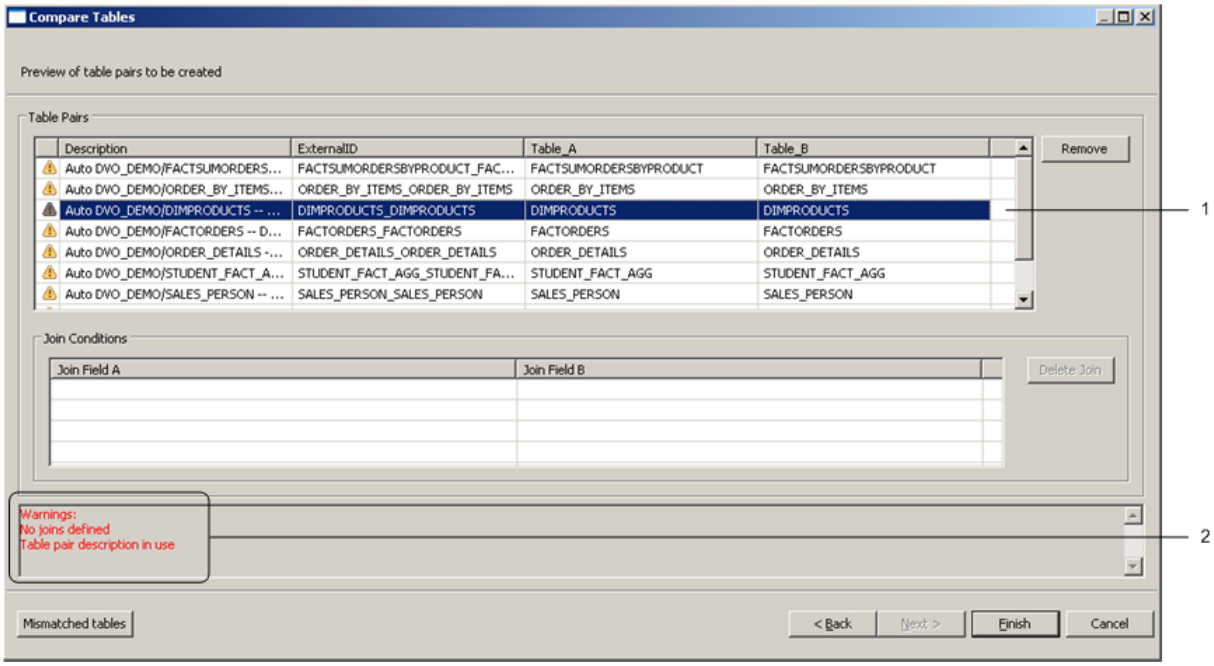
    If you select compare columns by position, if applicable, specify the number of columns to skip in Table A or Table B.
14. In the **Folder to Place Tests** field, select the folder to store the table pairs and tests.
15. In the **Tests to Include** field, specify whether to generate count tests or count and value tests.
16. In the **Trim Trailing Spaces from All Generated Tests** field, select whether to trim the trailing spaces in the values of table columns.
17. To generate value and outer value tests for tables that do not have primary keys, click **Browse** to select the text file that lists the primary keys for the flat files or tables.

    If you select a primary key for the table and specify a primary key file, Data Validation Option ignores the primary key file.
18. If a primary key does not exist for a table or file, in the **If no PK** field, select whether you want to generate count tests only or skip the test generation.

    If you skip the test generation and no primary key is defined, Data Validation Option does not generate the table pair.
19. Click **Next**.

    The **Compare Tables** dialog box previews table pairs that will be generated.

The following figure shows a preview of the table pairs that will be generated and the errors:



1. Selected table pair
2. Errors for the selected table pair

20. To not generate a table pair, select the table pair and click **Remove**.
21. If there is a problem with a table pair, select the table pair to view the error.
22. If an error says that no join is defined, select the table pair, and then select the join fields in the **Join Conditions** area.
23. Click **Mismatched Tables** to view a list of all tables that are not matched in each folder.

    Data Validation Option does not generate a table pair or tests for tables that are not matched.
24. Click **Finish**.

    Data Validation Option generates the table pairs and tests.

If a default setting does not apply to a particular table in a table pair, you can edit the table pair to change the setting after Data Validation Option generates the tables pairs and tests. For example, if a table requires a different connection than the default connection that you specified for all tables in the table comparison, edit the table pair and change the connection.

Editing Table Pairs

To edit a table pair, right-click the table pair in the Navigator and select **Edit Table Pair**. You can also edit a table pair by double-clicking the table pair.

When you edit a table pair, the **Table Pair Editor** dialog box opens.

Deleting Table Pairs

To delete a table pair, right-click a table pair in the Navigator and select **Delete Table Pair**. You can also select a table pair and press the Delete key to delete a table pair.

When you delete a table pair, Data Validation Option deletes all of the associated tests. Data Validation Option does not delete lookup, join, or SQL views used in the table pair.

Viewing Overall Test Results for a Table Pair

When you select a table pair in the navigator, the **Tests** tab of the details area shows all tests for the corresponding table pair. The Tests tab also shows information about the latest test run for each test.

Select a test to view the details about that test in the properties area. The properties area contains multiple views. You can view the test properties in the **Properties** view, test results in the **Results** view, and bad records in the **Bad Records** view.

CHAPTER 8

Tests for Table Pairs

This chapter includes the following topics:

Tests for Table Pairs Overview

You can create data validation tests for a table pair to compare the columns of the two tables. For example, you can create tests to verify that all corresponding columns have the same number of distinct values.

You can add the following types of tests for table pairs:

Aggregate tests

Includes COUNT, COUNT_DISTINCT, COUNT_ROWS, MIN, MAX, AVG, and SUM.

Set tests

Includes SET_AinB, SET_BinA, SET_AeqB, and SET_ANotInB.

Value tests

Includes VALUE and OUTER_VALUE.

You can also automatically generate tests to quickly create tests for table pairs. There are multiple ways to generate tests.

You can run all tests for a table pair. You can also run tests for all table pairs included in a folder in the Navigator.

If a test for a table pair fails to run, right-click the table pair to retrieve the corresponding PowerCenter session log. View the session log to view errors that occurred while running the underlying PowerCenter session.

Table-Pair Tests

The following table describes the table-pair tests:

Test	Description
COUNT	Compares the number of non-null values for each of the selected fields. This test works with any datatype. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.
COUNT_DISTINCT	Compares the distinct number of non-null values for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime.
COUNT_ROWS	Compares the total number of values for each of the selected fields. This test counts nulls, unlike the COUNT and COUNT_DISTINCT tests. This test works with any datatype.
MIN	Compares the minimum value for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime.
MAX	Compares the maximum value for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime.
AVG	Compares the average value for each of the selected fields. This test can only be used with numeric datatypes.
SUM	Compares the sum of the values for each of the selected fields. This test can only be used with numeric datatypes.
SET_AinB	Determines whether the entire set of values for Field A exist in the set of values for Field B. This test works with any datatype except binary/other. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.
SET_BinA	Determines whether the entire set of values for Field B exist in the set of values for Field A. Determines whether the entire set of values for the field selected from Table B exist in the set of values for the field selected from Table A. This test works with any datatype except binary/other. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.

Test	Description
SET_AeqB	Determines whether the set of values for the selected fields are exactly the same when compared. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.
SET_ANotInB	Determines whether there are any common values between the selected fields. If there are common values, the test returns an error. If there are no common values, the test succeeds.
VALUE	For joined table pairs, this test compares the values for the fields in each table, row-by-row, and determines whether they are the same. If there are any rows that exist in one table but not the other, the rows are disregarded which implies an inner join between the tables. If the fields are both null and the **Null=Null** option is disabled, this pair of records fails the test. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime.
OUTER_VALUE	For joined table pairs, this test compares the values for the fields in each table, row-by-row, and determines whether they are the same. If there are any rows that exist in one table but not the other, they are listed as not meeting the test rules which implies an outer join between the tables. For the test to pass, the number of rows for the tables, as well as the values for each set of fields must be equal. If the fields are both null and the **Null=Null** option is disabled, this set of records fails the test. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric- to-numeric or datetime-to-datetime.

Test Properties for a Table Pair

You define the test properties when you create a test for a table pair. Some test properties apply to specific tests.

The following table describes the test properties:

Property	Description
Function	The test you run such as COUNT, COUNT_DISTINCT, AinB, or VALUE.
Field A/B	The field that contains the values you want to compare when you run the test. You must select a field from each table in the table pair.
Operator	The arithmetic operator that defines how to compare each value in Field A with each value in Field B.
Threshold	The allowable margin of error for an aggregate or value test that uses the approximate operator. You can enter an absolute value or a percentage value.
Max Bad Records	The maximum number of records that can fail the comparison for a test to pass. You can enter an absolute value or a percentage value.
Case Insensitive	Ignores case when you run a test that compares string data.
Trim Trailing Spaces	Ignores trailing spaces when you run a test that compares string data. Data Validation Option does not remove the leading spaces in the string data.

Property	Description
Null=Null	Allows null values in two tables to be considered equal.
Condition A/B	Allows you to filter records after Data Validation Option joins the tables in the table pair. Enter a valid PowerCenter Boolean expression.
Field A/B is Expression	Allows you to enter an expression instead of a field name
Datatype	The datatype of the expression if you enter an expression instead of the field name.
Precision	The precision of the expression if you enter an expression instead of the field name.
Scale	The scale for the expression if you enter an expression instead of the field name. Maximum value is 99.
Comments	Information about a test. Data Validation Option displays the comments when you view test properties in the properties area.

Related Topics:

- "Spreadsheet Requirements for Table Pair Definitions" on page 83

Fields A and B

To create a test, you must select the fields that contain the values that you want to compare. Select a field from each table in the table pair. The fields available in each table appear in **Field A** and **Field B**.

Conditions A and B

You can filter the values for each field in a VALUE or OUTER_VALUE test to exclude rows that do not satisfy the test condition. For example, you want to exclude telephone extension numbers that contain fewer than three characters. Apply the following conditions to the VALUE test:

- Table A.EXT = Table B.EXT
- Condition A = `LENGTH(EXT)<3`
- Condition B = `LENGTH(EXT)<3`

The filter condition you enter for a test differs from the WHERE clause you enter for a table pair. Data Validation Option applies the WHERE clause to all tests in the table pair, before it joins the tables. It applies the test filter condition after it joins the tables. You might want to use a test filter condition instead of a filter condition in the WHERE clause when the filter condition does not remove a large percentage of rows. This can improve performance if you run one test on the table pair.

Data Validation Option does not check the condition syntax. Any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the PowerCenter syntax is not valid, a mapping installation error occurs when you run the test.

To enter a filter condition for either field, enter the filter condition in the **Condition A** or **Condition B** field. Because the PowerCenter Integration Service processes the filter condition, it must use valid PowerCenter syntax. Enter the field name in the filter condition, for example, Emp_ID > 0. Do not include the WHERE keyword.

Operator

The operator defines how to compare the test result for Field A with the test result for Field B. Enter an operator for aggregate and value tests.

The following table describes the operators available in the **Operator** field:

Operator	Definition	Description
=	Equals	Implies that the test result for Field A is the same as the test result for Field B. F or example, SUM(Field A) is the same as SUM(Field B).
<>	Does not equal	Implies that the test result for Field A is not the same as the test result for Field B.
<	Is less than	Implies that the test result for Field A is less than the test result for Field B.
<=	Is less than or equal to	Implies that the test result for Field A is less than or equal to the test result for Field B.
>	Is greater than	Implies that the test result for Field A is greater than the test result for Field B.
>=	Is greater than or equal to	Implies that the test result for Field A is greater than or equal to the test result for Field B.
≈	Is approximately the same as	Implies that the test result for Field A is approximately the same as the test result for Field B. An approximate test must have a threshold value. You can use this operator with numeric datatypes only.

Note: Data Validation Option compares string fields using an ASCII table.

Related Topics:

- "BIRT Report Examples" on page 167

Threshold

A threshold is a numeric value that defines an acceptable margin of error for a test. You can enter a threshold for aggregate tests and for value tests with numeric datatypes.

An aggregate test fails if the number of non-matching records exceed the threshold value. For example, you run a COUNT test that uses the "≈" operator and set the threshold to 10. The test passes when the results are within 10 records of each other.

In a value test, the threshold defines the numeric margin of error used when comparing two values. For example, you run a VALUE test that uses the "=" operator. The test compares a REAL field with a value of 100.99 to an INTEGER field with a value of 101. The test passes when the threshold value is at least 0.01.

You can enter an absolute value or a percentage value as the threshold. To enter a percentage value as the threshold, suffix the number with a percentage (%) sign. For example, 0.1%.

You must configure the threshold if the test uses the approximate operator.

Max Bad Records

Data Validation Option lists records that do not compare successfully as bad records. You can configure a tolerance value for the acceptable number of bad records.

By default, for a set or value test to pass, all records must compare successfully. You can configure an acceptable value for the maximum number of bad records. The test passes if the number of bad records does not exceed the max bad records value.

You can enter an absolute value or a percentage value for the max bad records. To enter a percentage value as the max bad records, suffix the number with a percentage (%) sign. For example, 0.1%.

Value and set tests display bad records on the Results tab.

Case Insensitive

String comparison in PowerCenter is case-sensitive. If you want the PowerCenter Integration Service to ignore case when you run a test that compares strings, enable the **Case Insensitive** option. This option is disabled by default.

Trim Trailing Spaces

By default, string comparison fails if two strings are identical except one string contains extra spaces. For example, one field value in a test is 'Data Validation' and the other field value is 'Data Validation ' (with three blank spaces after the last character). If you do not trim trailing spaces, the test produces a bad record. If you trim trailing spaces, the comparison passes because the extra spaces are ignored.

You might want to trim trailing spaces when the CHAR datatype, which pads a value entered with spaces to the right out to the length of the field, is used. A field of CHAR(20) compared to CHAR(30) fails, even if both fields have the same value, unless you trim the trailing spaces.

Enable the **Trim Trailing Spaces** option if there are spaces after the value entered in a field that should be ignored in the comparison. This option is disabled by default.

Null = Null

If null values in Field A and Field B should be considered equal, enable the **Null = Null** option. For example, a current employee in a table that contains employee information has a null termination date. If two records with null termination dates were compared by a database, they would not be considered equal because SQL does not consider a null value in one field to be equal to a null value in another field.

Because business users often consider null values to be equal, the **Null = Null** option is enabled by default.

Comments

You can enter information about a test in the **Comments** field. Data Validation Option displays comments in the Properties window when you select the test in the Navigator or the Tests tab.

Adding a Table-Pair Test

You can add a test to a table pair.

1. Right-click the name of the table pair in the Navigator, and select **Add Test**.

   The **Table Pair Test Editor** dialog box opens.
2. Enter the test properties.
3. Click **Save**.

Automatic Test Generation for Table Pairs

You can automatically generate tests for tables pairs to create tests quickly. You can generate tests between columns that have the same name or same position.

When you generate tests based on column names, Data Validation Option matches columns that have the same name and generates tests for each set of matched columns. Data Validation Option performs a case-sensitive match.

When you generate tests based on column position, Data Validation Option matches columns based on their column position in the table and generates tests for the matched columns. For example, Data Validation can generate tests between the third columns of both tables.

You can use the following methods to generate table-pair tests:

Select the Generate Value Tests option.

Use this method if you want to generate count and value tests for an existing table pair based on matching columns.

Import tests from a spreadsheet.

Use this method to generate all types of tests for an existing table pair. For example, you can generate count, value, MIN, AVG, and SET_AinB tests. You can also use this method to generate tests if you want to work offline or if you are not a Data Validation Option user.

Run a table comparison.

Use this method to generate table pairs between tables and flat files in two PowerCenter folders and generate count and value tests for each table pair.

Column Comparison by Position

You can generate table-pair tests that compare columns by position. Compare columns by position when the order of the columns in the two tables are the same.

If one table contains extra columns, you can skip the first columns of one table when you compare columns by position. For example, the first column in Table A matches the second column in Table B, and the second column in Table A matches the third column in Table B. To match the columns by position, you can skip the first column in Table B.

The following examples describe scenarios when you compare columns by position:

Table pair with tables that have the same number of columns and different names

For example, you have SampleA as Table A and SampleB as Table B in a table pair.

The following table lists the columns in SampleA and SampleB:

SampleA	SampleB
column1	col1
column2	col2
column3	col3

To compare column1, column2, and column3 in SampleA with col1, col2, and col3 in SampleB, respectively, compare the columns by position. Do not skip any columns in Table A or Table B.

Table pair with tables that have a different number of columns

For example, you have SampleA as Table A and SampleB as Table B in a table pair.

The following table lists the columns in SampleA and SampleB:

SampleA	SampleB
column1	col1
column2	col2
column3	col3
-	column1
-	column2
-	column3

To compare column1, column2, and column3 in SampleA with column1, column2, and column3 in SampleB, respectively, compare the columns by position. Skip three columns in Table B.

Table Pair with an SQL View and a Table

For example, you have an SQL view named sample_view that is based on Table1 and Table2. You also have a table named Table3. In the table pair, you define sample_view as Table A and Table3 as Table B.

The following table lists the columns in sample_view and Table3:

sample_view Columns	Table3 Columns
Table1.columnA	column1
Table1.columnB	column2
Table2.columnC	-
Table2.columnD	-

To compare Table2.columnC and Table2.columnD with column1 and column2 of Table3, respectively, compare the columns by position. Skip the first two columns in sample_view.

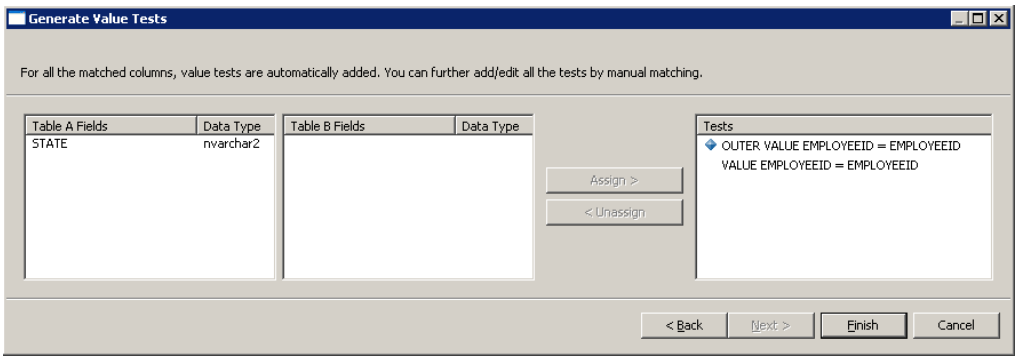
Generating Tests for a Table Pair

You can generate count and value tests for an existing table pair.

1. In the Navigator, right-click the table pair and select **Generate Value Tests**.
2. Select whether to compare columns by name or position.
3. If you compare columns by position and you want to skip columns in one table, specify the number of columns to skip in the Skip First Columns field.
4. Click **Next.**

   The **Generate Value Tests** dialog box shows the tests that are going to automatically be generated based on the matched columns. A blue diamond appears next to each automatically generated test.
5. To manually match additional columns, drag a column in Table A to the matching column in Table B.

   The following figure shows count and outer value tests for automatically matched columns and a value test for manually matched columns:



6. Click **Finish**.

   Data Validation Option generates the count and value tests for the matched columns for the table pair.

Test Generation for a Table Pair from a Spreadsheet

You can generate tests for a table pair from a spreadsheet. Use a spreadsheet to define and generate all types of tests, including MIN, AVG, and SET_AinB.

You might also use a spreadsheet to define tests if you want to work offline. Or, you might use a spreadsheet to define tests if you are not a Data Validation Option user. For example, business analysts might need to create the test definitions because they are more familiar with the data.

To generate tests from a spreadsheet, compete the following steps:

1. Export existing table-pair tests.
2. Add tests to the spreadsheet.
3. Import the tests from the spreadsheet.
4. Resolve tests with errors.

Export table-pair tests to generate a spreadsheet that contains column headers, test metadata, and column metadata. If the table pair already has tests defined, the spreadsheet also contains existing test definitions.

Note: You can find a spreadsheet template in the following Data Validation Option installation directory:

```
<Data Validation Option installation directory>\samples\TablePair-template.xlsx
```

After you generate the spreadsheet, you can add tests to the spreadsheet. For each test, enter the test properties in the spreadsheet. The spreadsheet contains restricted values for particular test properties based on the column and test metadata.

You can then import the tests from the spreadsheet into Data Validation Option. When you import tests from a spreadsheet, Data Validation Option lists and validates the test definitions, and shows any errors. For example, an error appears if you use the approximate operator on a string column in a test definition. You cannot apply the approximate operator on string columns. You must resolve the errors or remove the test before you can import the test definitions. To resolve errors, edit the test definition.

Data Validation Option generates a new test for each test definition specified in the spreadsheet. If you import a test that already exists, Data Validation Option generates a duplicate version of the test. Data Validation Option does not update existing tests or remove tests when you import tests from a spreadsheet.

Spreadsheet Requirements for Table-Pair Tests

You can generate tests for a table pair based on the test definitions specified in a spreadsheet.

Consider the following requirements for the spreadsheet:

- Define tests in the first worksheet of an Excel workbook. Data Validation Option reads the first worksheet and ignores other worksheets in an Excel workbook.
- The Excel file must have a .xls or .xlsx file extension.
- To define tests for multiple table pairs, define tests for each table pair in a separate Excel workbook.
- You can only define one test per row.
- You must specify a field name or a field expression for each table in each row of the spreadsheet. If you specify a field name and field expression for the same table in a row, Data Validation Option uses the expression.
- If you select an operator that is not valid for a particular datatype and you try to import the tests from the spreadsheet, Data Validation Option displays an error.
- The values in the spreadsheet are case sensitive.

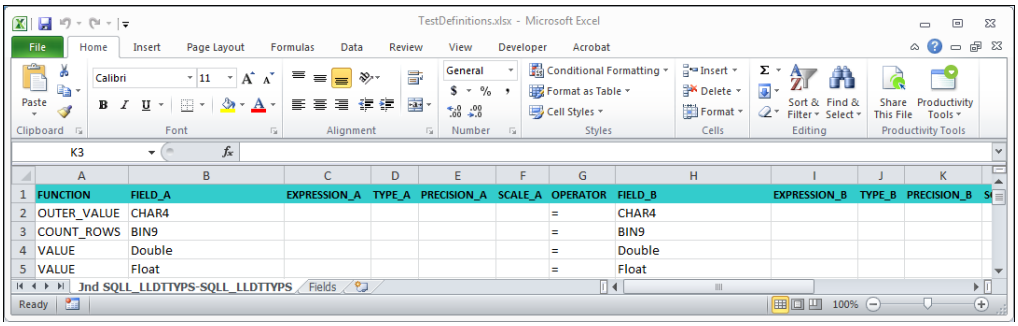
Generating Tests for a Table Pair from a Spreadsheet

Data Validation Option can generate table-pair tests based on test definitions in a spreadsheet. To generate the tests, export the existing tests, add tests to the spreadsheet, and then import the tests from the spreadsheet into Data Validation Option.

1. In the Navigator, right-click the table pair and select **Export Tests to Spreadsheet**.

   The **Save As** dialog box appears.
2. Save the spreadsheet to a location that the Data Validation Client can access.
3. Open the spreadsheet.

The spreadsheet shows the existing table-pair test definitions if tests exist.



	A	B	C	D	E	F	G	H	I	J	K
1	FUNCTION	FIELD_A	EXPRESSION_A	TYPE_A	PRECISION_A	SCALE_A	OPERATOR	FIELD_B	EXPRESSION_B	TYPE_B	PRECISION_B
2	OUTER_VALUE	CHAR4					=	CHAR4			
3	COUNT_ROWS	BIN9					=	BIN9			
4	VALUE	Double					=	Double			
5	VALUE	Float					=	Float			

4. Add tests for the table pair.

   Verify that the content in the spreadsheet meets the spreadsheet requirements.

5. Save and close the spreadsheet file.

6. In the Navigator, right-click the table pair, and click **Generate Tests from Spreadsheet**.

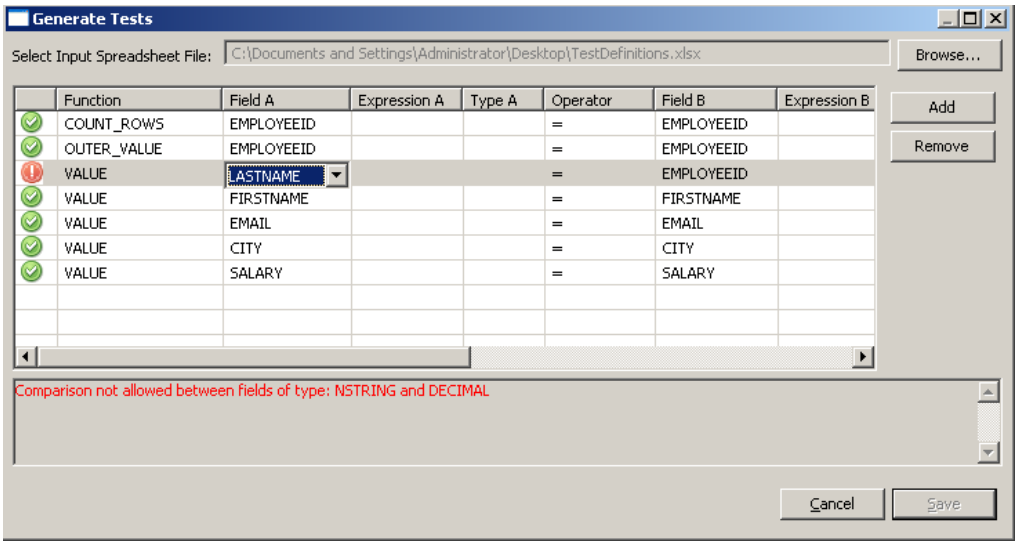
   The **Generate Tests** dialog box appears.

7. Click **Browse** and select the spreadsheet.

   The **Generate Tests** dialog box previews the tests to be generated and shows any errors. A red circle icon indicates that the row has an error. A yellow triangle icon next to a cell indicates that the value of the cell has an error.

8. If a test has an error, select the test row with the error to preview the error.

   The error appears at the bottom of the **Generate Tests** dialog box.



9. Resolve the error or remove the test.

10. To change a test property, select the property in the **Generate Tests** dialog box and then make the change.

11. To remove a test, select the test row and click **Remove**.

12. To add a test, click **Add** and enter the test properties.

13. Click **Save** to import the tests.

Editing a Table-Pair Test

To edit a test, right-click the test in the Navigator, and select **Edit Test**.

Deleting a Table-Pair Test

To delete a test, right-click the test in the Navigator, and select **Delete Test**.

Running Table-Pair Tests

You can run tests from the Data Validation Client or the DVOCmd command line.

Before you run tests, verify that the Data Validation Option target folder is closed in the PowerCenter Designer and the PowerCenter Workflow Manager. If the target folder is open and you run tests, Data Validation Option cannot write to the target folder and the tests fail.

Use one of the following methods to run tests:

- To run all tests defined for a table pair, right-click the table pair object in the Navigator, and click **Run Table Pair Tests**.
- To run tests for all table pairs in a folder, right-click the folder in the Navigator and select **Run Folder Tests.**

After you run a test, you can view the results on the **Results** view in the **Properties** area.

Data Validation Option uses the following logic to determine whether a test passes:

- For an aggregate test, the `A <operator> B` statement must be true. For example, the following statement must be true:

  ```
  A > B
  ```

- For a test that uses the approximate operator, the `ABS(A-B) <= Threshold` statement must be true.
- For a value or set test, the number of records that do not match must be fewer than or equal to the threshold value. If there is no threshold value and there are no records that do not match, the test passes.

Note: To run multiple table pair objects in parallel on DVOCmd command line, you must first install the tests, and then run then the tests.

Bad Records for a Table-Pair Test

Bad records are the records that fail a value or a set test.

When you select a test on the Tests tab, the bad records appear in the **Bad Records** view.

The Bad Records view displays a delimiter before and after each string value to help you identify whitespace in a record of string data type. The Bad Records view displays the difference in value along with the percentage of difference in a record of a numeric data type. Data Validation Option uses the following

formula to calculate the percentage of difference: `(<larger value> - <smaller value>) ÷ <larger value> * 100`

The columns that appear on the **Bad Records** view differ based on the test.

Aggregate Tests

Aggregate tests do not display bad records. The **Results** view displays the test result value from Field A, the test result value from Field B, and the comparison operator.

Value Tests

Value tests display the following columns for each bad record:

- The key for Table A
- The field or expression from Table A being compared
- The key for Table B
- The field or expression from Table B being compared

Set Tests

Set tests display the following columns for each bad record:

- The result from Table A
- The result from Table B

If you choose to view more details about numerical bad data records, the tests display the following additional columns:

- Difference
- % Difference

Note: If you compare two fields where one field has a value and the other is empty, Data Validation option considers the record as a bad record. If the repository is on Oracle, the database stores the empty field as NULL. A bad record with NULL value in an Oracle repository can be either NULL or an empty field.

Viewing the Details of a Bad Record

If you know what caused a record to fail, you can take corrective action on the data. You can view the details of a bad record for value tests, unique tests, and set tests in the Bad Records view.

1. Run a test.
2. In the **Tests** tab of the details area, click the test.

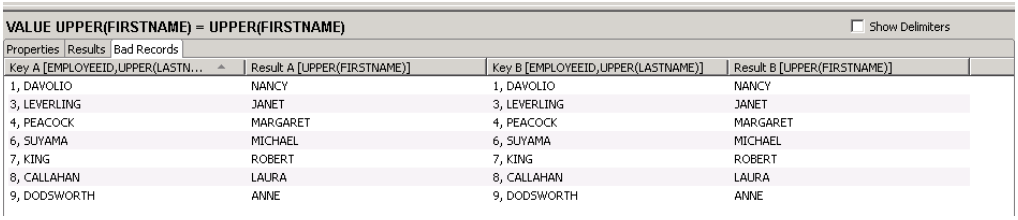
   If the test failed, the properties area includes a **Bad Records** view.
3. Click the **Bad Records** view.

   If the test evaluated strings, a **Show Delimiters** option appears in the upper-right corner of the properties area. If the test evaluated numerical values, a **Show Differences** option appears.

4. To view the details of a bad record, select the **Show Delimiters** or **Show Differences** option.

   Delimiters appear to help you identify whitespace in string data.

   The following image shows records without delimiters:

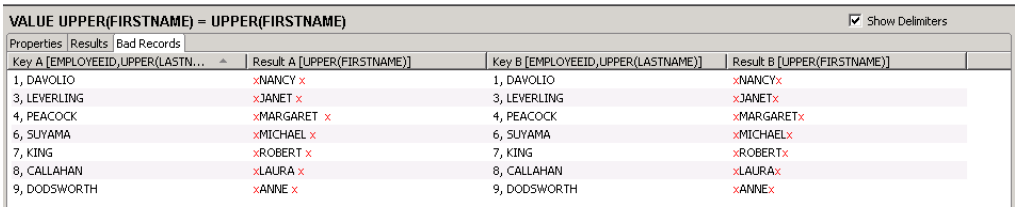

VALUE UPPER(FIRSTNAME) = UPPER(FIRSTNAME)

Show Delimiters

Properties | Results | Bad Records

Key A [EMPLOYEEID,UPPER(LASTN...	Result A [UPPER(FIRSTNAME)]	Key B [EMPLOYEEID,UPPER(LASTNAME)]	Result B [UPPER(FIRSTNAME)]
1, DAVOLIO	NANCY	1, DAVOLIO	NANCY
3, LEVERLING	JANET	3, LEVERLING	JANET
4, PEACOCK	MARGARET	4, PEACOCK	MARGARET
6, SUYAMA	MICHAEL	6, SUYAMA	MICHAEL
7, KING	ROBERT	7, KING	ROBERT
8, CALLAHAN	LAURA	8, CALLAHAN	LAURA
9, DODSWORTH	ANNE	9, DODSWORTH	ANNE

   The following image shows records with delimiters:


VALUE UPPER(FIRSTNAME) = UPPER(FIRSTNAME)

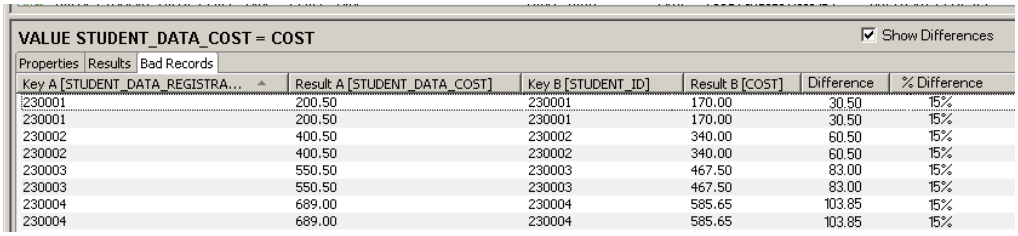
Show Delimiters

Properties | Results | Bad Records

Key A [EMPLOYEEID,UPPER(LASTN...	Result A [UPPER(FIRSTNAME)]	Key B [EMPLOYEEID,UPPER(LASTNAME)]	Result B [UPPER(FIRSTNAME)]
1, DAVOLIO	xNANCY x	1, DAVOLIO	xNANCYx
3, LEVERLING	xJANET x	3, LEVERLING	xJANETx
4, PEACOCK	xMARGARET x	4, PEACOCK	xMARGARETx
6, SUYAMA	xMICHAEL x	6, SUYAMA	xMICHAELx
7, KING	xROBERT x	7, KING	xROBERTx
8, CALLAHAN	xLAURA x	8, CALLAHAN	xLAURAx
9, DODSWORTH	xANNE x	9, DODSWORTH	xANNEx

   The **Difference** and **% Difference** columns appear for numeric data types.

   The Difference and % Difference columns in the following image display information about why the records failed the test:


VALUE STUDENT_DATA_COST = COST

Show Differences

Properties | Results | Bad Records

Key A [STUDENT_DATA_REGISTRA...	Result A [STUDENT_DATA_COST]	Key B [STUDENT_ID]	Result B [COST]	Difference	% Difference
230001	200.50	230001	170.00	30.50	15%
230001	200.50	230001	170.00	30.50	15%
230002	400.50	230002	340.00	60.50	15%
230002	400.50	230002	340.00	60.50	15%
230003	550.50	230003	467.50	83.00	15%
230003	550.50	230003	467.50	83.00	15%
230004	689.00	230004	585.65	103.85	15%
230004	689.00	230004	585.65	103.85	15%

Table Pair Test Example

You want to compare the fields in a flat file and validate the sales against the data in the target Oracle table.

The following table displays the columns and data types in a flat file source:

Column	Data Type
EmployeeID	String(10,0)
LastName	String(20,0)
FirstName	String(20,0)
Designation	String(30,0)
Location	String(15,0)

Column	Data Type
State	String(5,0)
Pin	String(10,0)
Salary	String(20,0)

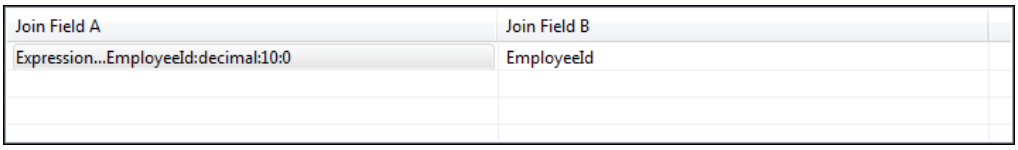
The following table displays the columns and data types in the Oracle target:

Column	Data Type
EmployeeID	number(p,s)(2,0)
LastName	varchar2(9,0)
FirstName	varchar2(10,0)
Designation	varchar2(24,0)
Location	varchar2(8,0)
State	varchar2(2,0)
Pin	number(p,s)(7,0)
Salary	number(p,s)(11,0)

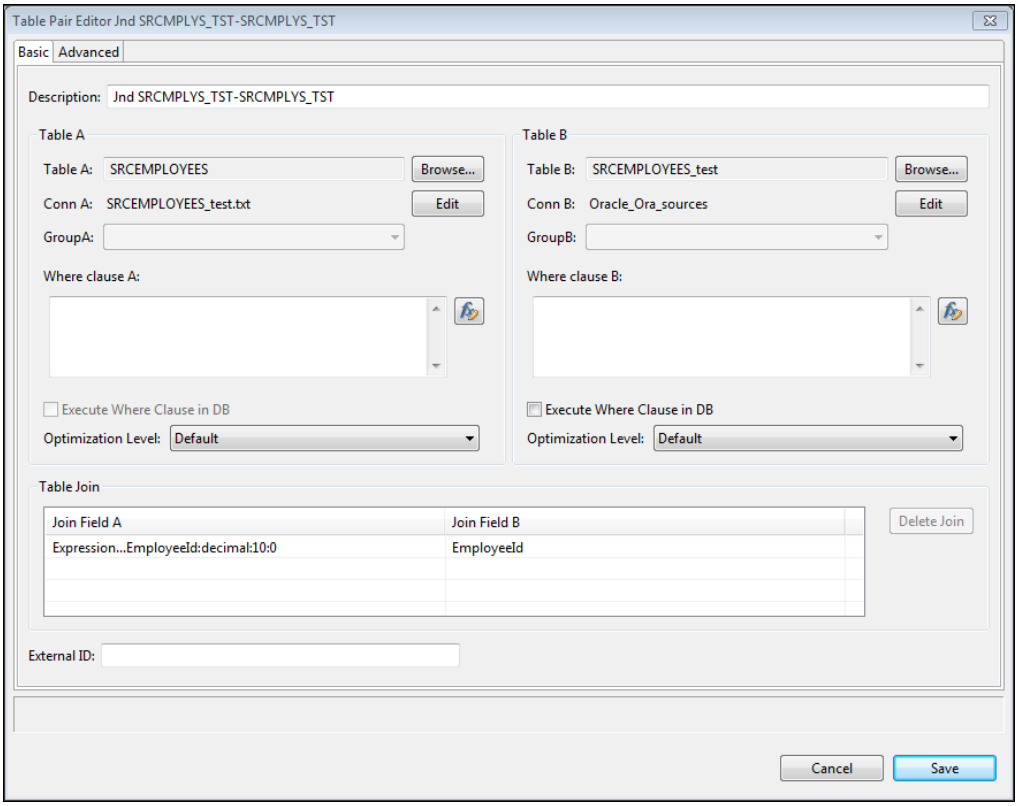
To test the data stored in the source and validate against data stored in the target, perform the following steps:

1. Create the table pair test.
   Select the flat file as Table A and the Oracle table as Table B.
2. Create a table join. Select EmployeeId in Join Field A and Join Field B.
   Because the fields in the source and target are not of the same or compatible type, you cannot do a direct join. To convert the EmployeeId field in the source table from string to decimal data type, add an expression.

   The following image shows the expression:

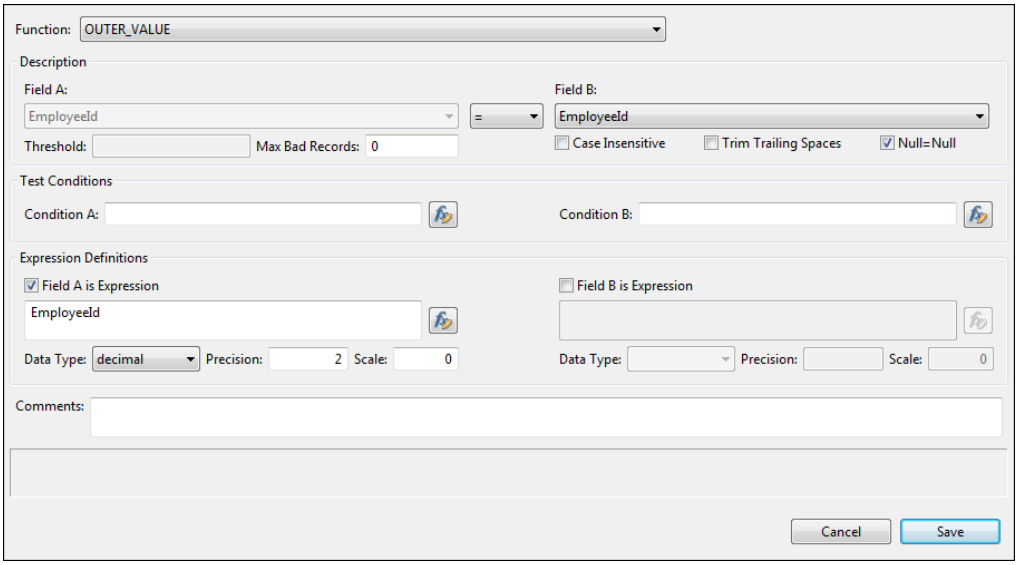


The following image shows the table pair editor:



3. Create an OUTER_VALUE test.
   To compare EmployeeId in the source and the target table, create an OUTER_VALUE test. The test compares the values of the fields, and determines whether they are the same. Because the EmployeeId in source and target tables are of a different data type, add an expression to EmployeeId to convert it to decimal, and compare it with EmployeeId in the target table.

   The following image shows the table pair test editor:



   The test checks the contents of the target table against the contents of the source table.

Troubleshooting Table-Pair Tests

Table-pair tests can fail for different reasons. Review the tips to help you troubleshoot table-pair tests.

Review the following troubleshooting tips for table-pair tests:

The table-pair test fails.

If the Data Validation Option target folder in the PowerCenter Designer or the PowerCenter Workflow Manager is open, Data Validation Option cannot write to the folder and the tests fail. Before you run tests, verify that the target folder is closed in the PowerCenter Designer and the PowerCenter Workflow Manager.

Data Validation Option creates a bad record during a table-pair test when one of the values is empty.

If you compare two fields where one field has a value and the other is empty, Data Validation option considers the record as a bad record. If the repository is on Oracle, the database stores an empty field as NULL. A bad record with NULL value in an Oracle repository can be either NULL or an empty field.

The table join failed for a table pair that has a file list connection.

If you specify a file list connection and the keys in the join condition are not unique among all files in the file list, the join fails. Choose keys that uniquely identify records among all files in the file list.

CHAPTER 9

Single-Table Constraints

This chapter includes the following topics:

Single-Table Constraints Overview

Use a single-table constraint to run tests on a single table. Single-table constraints define valid data within a table. You can enforce valid values, aggregates, formats, and uniqueness. For example, you might want to verify that no annual salary in an employee table is less than $10,000.

Errors in complex logic often manifest themselves in very simple ways, such as NULL values in the target. Therefore, setting aggregate, value, NOT_NULL, UNIQUE, and FORMAT constraints on a target table is a critical part of any testing plan.

To run single-table constraints, you must create a single table. You can select a relational table, flat file, XML file, lookup view, SQL view, or join view as a single table.

If a single-table constraint fails to run, right-click the single table to retrieve the corresponding session log. View the session log to view errors that may have occurred while running the underlying PowerCenter session.

Basic Properties for a Single-Table Constraint

You can view and configure the basic properties for a single-table constraint. The properties vary based on the types of objects you select for the single-table constraint.

Description

Description

Single-table constraint description. By default, Data Validation Option uses the table name.

Table

You can configure the following properties for the table:

Table A

The table name. Select **Browse** to select a table. You can search for a data source by name or path. You can search for a lookup view, SQL view, join view, and aggregate view by name.

Conn A

Connection properties for the selected table. Select **Edit** to edit the connection properties.

XML Group A

The XML group name for an XML data source. Only child elements of the XML group are available to use in the WHERE clause, primary key field, and tests.

Where Clause

The WHERE clause filters the records that are read from the data source. If the PowerCenter Integration Service processes the WHERE clause, enter a valid PowerCenter expression. If a relational, Salesforce, SAP, or SAS data source processes the WHERE clause, enter a WHERE clause that is supported by the data source.

Execute Where Clause in DB

If the data source is relational, Salesforce, SAP, or SAS, enable if you want the data source to process the WHERE clause. You must select this option to select the **Where Clause, Sorting and Aggregation in DB** optimization level.

Optimization Level

Controls the test logic processed by the PowerCenter Integration Service and the data source. Select one of the following options:

- Default. The PowerCenter Integration Service sorts the input data and processes all test logic.
- WHERE clause, Sorting, and Aggregation in DB. If the data source is a relational data source, the database processes the WHERE clause, sorts records for joins, and aggregates data for aggregate tests. The PowerCenter Integration Service processes all remaining test logic that the data source does not process.
- Already Sorted Input. Indicates that the records in the data source are already sorted. If the records in the data source are not sorted, tests might fail.

Table Join

Primary Key

Primary key column or columns for the table.

The primary key columns that appear for an XML data source are based on the selected XML group. Only child elements of the selected XML data source appear as primary key columns.

External ID

External ID

Identifier for the single table that you can use when you run Data Validation Option tests at the command line.

Related Topics:

- "XML Groups" on page 39

Advanced Properties for a Single-Table Constraint

You can configure bad records, a parameter file, sampling, and caching for a single-table constraint. The properties vary based on the types of objects you select for the single-table constraint.

Bad Records

You can configure the following properties for bad records:

Save All Bad Records for Test Execution

Saves all bad records for value and unique tests for single-table constraints.

Write Bad Records to:

The location where Data Validation Option writes bad records. You can select one of the following options:

- Flat File. Data Validation Option writes bad records to the specified flat file.
- DVO Schema. Data Validation Option writes bad records to the Data Validation Option repository.

File Name

The location and name of the flat file to which Data Validation Option writes bad records. Default location is `$INFA_HOME/server/infa_shared/BadFiles/<single table name /table pair name>/<file name>`.

Parameters

Parameter File

Name of the parameter file. Click **Add** to add a parameter file. Click **Delete** to delete the selected parameter file.

Data Sampling

You can configure the following properties for data sampling:

Enable Data Sampling

Indicator that determines if data sampling is enabled.

Percentage of Rows

Percentage of rows that you want to sample. Minimum value is .000001. Maximum value is less than 100.

Seed Value

The starting value used to generate a random number for sampling. You can configure the seed value for the following platforms:

- PowerCenter. Minimum seed value is 0. Maximum seed value is 4294967295.
- IBM DB2. Minimum seed value is 1 - 10^{31}. Maximum seed value is 10^{31} - 1.
- Oracle. Minimum seed value is -2,147,483,648. Maximum seed value is 2,147,483,647.

Use Native Sampling

The database performs the sampling. If you do not enable this option, PowerCenter performs the sampling.

Cache Size

Cache Size

Total cache used to run PowerCenter sessions associated with the table pair. By default, the units are bytes. Append KB, MB, or GB to the value to specify other units. For example, you can specify 1048576, 1024 KB, or 1 MB. Select **Automatic** to enable PowerCenter to compute the cache size.

Bad Records Configuration for a Single-Table Constraint

You can store up to 16,000,000 bad records per test to perform advanced analysis and up to 1000 bad records for reporting.

For reporting, you can set the maximum number of bad records in the mapping properties in the Preferences dialog box. The default number of bad records is 100. You can set up to 1000 bad records for reporting.

For advanced analysis, you can set the maximum number of bad records for detailed analysis and the file delimiter. You set the properties in the Detailed Error Rows Analysis section in the mapping properties. The default number of bad records is 5000. You can set up to 16,000,000 bad records.

You can store all the bad records from the tests for a table pair or single table. Select Save all bad records for test execution on the **Advanced** tab when you create or edit a table pair or single table.

Data Validation Option stores the following constraint tests for single tables:

- Value
- Unique

You must select whether you want to store the bad records in a flat file or a table in the Data Validation Option schema. If you store bad records in a flat file, you can optionally enter the name of the file. Data Validation Option appends test information to the name and retains the file extension.

Note: If you modify the file delimiter in the preferences file, run the InstallTests command with the forceInstall option for the existing table pairs or single tables that you already ran. You can also edit and save the table pair or single table from the Data Validation Client before you run the test. If you modify the bad records value, you need not reinstall the tests.

Bad Records in Database Schema Mode

If you choose to save detail bad records in the Data Validation Option schema, the bad records are written into the detail tables.

The following table describes the tables to which Data Validation Option writes the bad records based on the type of test:

Test Type	Table	Columns
Value, Outer Value, Value Constraint	ALL_VALUE_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, KEY_A, VALUE_A, KEY_B, VALUE_B
Unique Constraint	ALL_UNIQUE_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, VALUE
Set	ALL_SET_RESULT_DETAIL	TEST_RUN_ID, TEST_CASE_INDEX, VALUE_A, VALUE_B

Note: You must ensure that the database table has enough table space to hold all the bad records.

The test details of all the tests that you ran in Data Validation Option is available in the TEST_CASE table. The test installation details of the tests are available in the TEST_INSTALLATION table. You can obtain the TEST_ID of a test from the TEST_INSTALLATION table. You need TEST_ID of a test to query the complete details of a test from the TEST_CASE table.

You can get the Test Case Index from the Properties tab and the Test Run ID from the results summary of the test in the detail area. You can get the Table Pair ID/Table ID from the Table Pair or Single Table properties.

For example, you ran a table pair with a value test and outer value test.

The following SQL query is a sample query to retrieve the information of bad records of a test with Test Case Index as 5.

```
select ALL_VALUE_RESULT_DETAIL.*,TEST_CASE.* from ALL_VALUE_RESULT_DETAIL , TEST_RUN,
TEST_INSTALLATION, TEST_CASE
where
ALL_VALUE_RESULT_DETAIL.TEST_RUN_ID=TEST_RUN.TEST_RUN_ID
and TEST_RUN.TEST_INSTALLATION_ID=TEST_INSTALLATION.TEST_INSTALLATION_ID
and TEST_INSTALLATION.TABLE_PAIR_ID=TEST_CASE.TEST_ID
and ALL_VALUE_RESULT_DETAIL.TEST_CASE_INDEX=TEST_CASE.TEST_CASE_INDEX
and ALL_VALUE_RESULT_DETAIL.TEST_RUN_ID=220
```

Bad Records in Flat File

If you configure to store bad records in flat file, Data Validation Option creates the flat file in the machine that runs Informatica services.

The flat files that Data Validation Option generates after running the tests are stored in the following folder:`<PowerCenter installation directory>\server\infa_shared\TgtFiles`

You can edit the folder name in the Administration tool. Modify the $PMTarget variable in the PowerCenter Integration Service.

Data Validation Option generates a folder for each of the single table. The name of the single table is in the following format: `SingleTableName_TestRunID`

Data Validation Option creates flat files for each test inside the folder. The name of the flat file is in the following format: `<user defined file name>_ TestCaseType_TestCaseColumn A_TestCaseIndex.<user defined file extension>`

Suppose you enter the file name as BAD_ROWS.txt when you configure the single table and you run an value constraint test on field SSID. The test-case index is 1. The bad records file after you run the test is of the format, BAD_ROWS_VALUE_CONSTRAINT_SSID_1.txt.

Data Validation Option supports all the file delimiters that PowerCenter supports. When you enter non-printable characters as delimiters, you should enter the corresponding delimiter code in PowerCenter. When you import these files in PowerCenter, you have to manually create the different data fields since the code appears in the place of delimiter in the bad records file.

Caution: Data Validation Option uses comma as a delimiter if there a multiple primary keys. Do not use comma as a file delimiter.

When you run the tests for the single table, Data Validation Option stores the details of the bad records along with the following format:

```
single table Name
Table A Name
Table A Connection
Test Definition
Test Run Time
Test Run By User
Key A[],Result A[]
```

Data Sampling for a Single-Table Constraint

You can use data sampling to run tests on a subset of a dataset. You might use data sampling when the data set is large. You can perform data sampling on table pairs and single tables.

When you run a test on a sample data set, Data Validation Option runs the test on a percentage of the data set. The sample percentage that you specify represents the chance that each data is included in the sample.

You can use a seed value to repeat the same sample data set in multiple runs of a test. Data Validation uses the seed value as the starting value to generate a random number. If you do not enter a seed value, Data Validation Option generates a random seed value for each test run. You might use a seed value to replicate a Data Validation Option test.

By default, PowerCenter performs the sampling. If you sample data from IBM DB2, Microsoft SQL Server, Oracle, or Teradata, you can perform native sampling in the database. Push sampling to the database to increase performance.

If you add the WHERE clause and enable sampling, the order of operations depend on where you perform sampling and execute the WHERE clause. Generally, PowerCenter and the database performs sampling before executing the WHERE clause. However, when you configure the database to execute the WHERE clause and PowerCenter to perform sampling, the database executes the WHERE clause before PowerCenter performs the sampling.

Note: Because data sampling reduces the number of rows, some tests might have different results based on whether you enable data sampling. For example, a SUM test might fail after you enable data sampling because Data Validation Option processes less rows.

Rules and Guidelines for Data Sampling

Certain rules and guidelines apply when you perform data sampling on a single-table constraint.

Use the following rules and guidelines when you perform data sampling on a single-table constraint:

- The sample percentage that you specify may differ from the actual sample percentage during a test run. The sample percentage that you specify represents the chance that each data is included in the sample.
- You cannot use native sampling if the underlying Oracle tables or views from which you are sampling do not have unique primary keys. Instead, use PowerCenter sampling.
- Some tests might have different results based on whether you enable data sampling.

PowerCenter Cache for a Single-Table Constraint

You can configure the total cache memory for a single-table constraint. If you enable caching, the PowerCenter Integration Service processes the Lookup, Joiner, Aggregator, and Sorter transformations faster. The PowerCenter Integration Service creates in-memory index and data caches to temporarily store data for each transformation while processing the transformation.

If you enable caching and run a test for a single-table constraint, the PowerCenter Integration Service divides the specified cache size among all Aggregator, Joiner, Lookup, and Sorter transformations in the underlying PowerCenter session. The PowerCenter Integration Service allocates cache memory based on the transformation requirements, such as the number of rows processed, number and types of tests, and precision and scale of the table columns.

For optimal session performance, configure the cache size so that the PowerCenter Integration Service processes each transformation in memory without paging to disk. Session performance decreases when the Integration Service pages to disk. To determine the required cache size for each transformation, review the cache size specified in the session log after the session completes.

Adding a Single Table

You can create a single table from the file menu or from the shortcut in the menu bar.

1. Select the folder to which you want to add the single table.
2. Right-click the folder and select **Add Single Table**.

   The **Single Table Editor** window appears.
3. Browse and select the data source that you want to use the single table.

   You can search for a data source by name or path. You can search for lookup views, sql views, and join views only with their names.
4. Click **Edit** and configure the connection properties for the table.
5. Enter the WHERE clause you want to execute on the data source.
6. Enter the description for the single table.
7. Enter the external id for the single table.

   You can use the external id to execute the single table from the command line.

8. If your data source is relational, you can choose whether to execute the WHERE clause within the data source.

   If you choose to execute the WHERE clause within the database, PowerCenter Integration Service passes the WHERE clause to the database for execution before data loading.

9. Select the database optimization level.

   The following options are available:

   - Default. Data Validation Option creates a PowerCenter mapping based on the test logic and applies sorting on the data source.
   - WHERE clause, sorting, and aggregation in DB. Data Validation Option pushes the WHERE clause and sorting to the database. Applicable for relational data sources.
   - Already sorted input. Data Validation Option creates a PowerCenter mapping based on the test logic. If the data source is not sorted, the tests may fail.

10. Select the primary key for the table in the **Key Column** pane.

Editing Single Tables

To edit a single table, right-click it in the Navigator or on the **Single Tables** tab, and select **Edit Single Table**. You can also edit a single table by double-clicking it on the **Single Tables** tab.

When you edit a single table, the **Single Table Editor** dialog box opens

Deleting Single Tables

To delete a single table, right-click it in the Navigator or Single Tables tab, and select **Delete Single Table**. You can also delete a single table by selecting it and pressing the Delete key. When you delete a single table, Data Validation Option deletes all of its tests. Data Validation Option does not delete lookup or SQL views used in the single table.

Viewing Overall Test Results for a Single-Table Constraint

When you select a single-table constraint in the navigator, the **Tests** tab of the details area shows all tests for the selected single-table constraint. The Tests tab also shows information about the latest test run for each test.

Select a test to view the details about that test in the properties area. The properties area contains multiple views. You can view the test properties in the **Properties** view, test results in the **Results** view, and bad records in the **Bad Records** view.

CHAPTER 10

Tests for Single-Table Constraints

This chapter includes the following topics:

Tests for Single-Table Constraints Overview

You can create data validation tests for a single-table constraints to validate the values in columns of a table. For example, you can compare the column values against a constraint value or verify that all values in a column are not null.

You can add the following types of tests for single-table constraints:

Aggregate tests

Includes COUNT, COUNT_DISTINCT, COUNT_ROWS, MIN, MAX, AVG, and SUM.

Value tests

Includes VALUE.

Other tests

Includes FORMAT, UNIQUE, NOT_NULL, and NOT_BLANK.

Most single-table constraints allow you to enter a constraint value for the test. The constraint value defines the value or values to which you want to compare the values in a field. For example, you might want to verify that a SALARY field contains values greater than $10,000. Enter the minimum salary as the constraint value.

Single-Table Constraint Tests

The following table describes the single-table constraint tests:

Test	Description
COUNT	Compares the number of non-null values for the selected field to the constraint value. This test works with any datatype.
COUNT_DISTINCT	Compares the distinct number of non-null values for the selected field to the constraint value. This test works with any datatype except binary.
COUNT_ROWS	Compares the total number of values for the selected field to the constraint value. This test counts nulls, unlike the COUNT and COUNT_DISTINCT tests. This test works with any datatype.
MIN	Compares the minimum value for the selected field to the constraint value. This test works with any datatype except binary.
MAX	Compares the maximum value for the selected field to the constraint value. This test works with any datatype except binary.
AVG	Compares the average value for the selected field to the constraint value. This test can only be used with numeric datatypes.
SUM	Compares the sum of the values for the selected field to the constraint value. This test can only be used with numeric datatypes.
VALUE	Examines the values for the field, row by row, and compares them to the constraint value. This test works with any datatype except binary.
FORMAT	Determines whether the values in the field match the pattern that you use as the constraint value. The PowerCenter Integration Service uses the REG_MATCH function for this test. You cannot use this test with binary datatypes. If you choose to use a pattern, use the perl compatible regular expression syntax. Enclose the pattern within single quotation marks. For example, to test the format of a telephone number of the form xxx-xxx-xxxx, use the following pattern: `\d\d\d-\d\d\d-\d\d\d\d` The test passes if the phone number is 049-157-0156 but fails if the phone number is 0491-570-156. To test a value that consists of a number followed by two alphabets and another number, use the following pattern: `\d[a-z][a-z]\d` The test passes if the alphanumeric string is 1ab2 but fails if the alphanumeric string is a1b2.
UNIQUE	Confirms that the value in the field is unique. This test does not use a constraint value. This test cannot be used with binary datatypes.
NOT_NULL	Confirms that the value in the field is not null. This test does not use a constraint value. This test cannot be used with binary datatypes.
NOT_BLANK	If the value in the field is a string value, this test confirms that the value in the field is not null or an empty string. If the value in the field is a numeric value, this test confirms that the value in the field is not null or zero. This test does not use a constraint value. This test cannot be used with datetime or binary datatypes.

Test Properties for a Single-Table Constraint

You define the test properties when you create a test for a single-table constraint. Some test properties apply to specific tests.

Edit test properties in the **Single Table Test Editor** dialog box when you add or edit a test.

The following table describes the test properties:

Property	Description
Function	The test you run such as COUNT, COUNT_DISTINCT, VALUE, or NOT_NULL.
Field	The field that contains the values you want to test.
Operator	The operator that defines how to compare each value in the field with the constraint value.
Constraint Value	The value or values you want to compare the field values to.
Threshold	The allowable margin of error for an aggregate or value test that uses the approximate operator. You can enter an absolute value or a percentage value.
Max Bad Records	The number of records that can fail comparison for a test to pass. You can enter an absolute value or a percentage value.
Case Insensitive	Ignores case when you run a test on string data.
Trim Trailing Spaces	Ignores trailing spaces when you run a test that on string data. Data Validation Option does not remove the leading spaces in the string data.
Condition	Filter condition for the test. Enter a valid PowerCenter Boolean expression.
Field is Expression	Allows you to enter an expression for the field.
Datatype	The datatype for the expression if the field is an expression.
Precision	The precision for the expression if the field is an expression.
Scale	The scale for the expression if the field is an expression. Maximum value is 99.
Comments	Information about a test. Data Validation Option displays the comments when you view test properties in the Properties area.

Field

To create a single-table constraint, you must select the field that contains the values you want to test. The fields available in the single table appear in the **Field** drop-down list. Select a field from the list.

Condition

You can filter the values for the test field in a VALUE, FORMAT, NOT_NULL, or NOT_BLANK test. Data Validation Option does not test records that do not satisfy the filter condition. For example, you want to test rows in an ORDERS table only if the store ID number is not “1036.” Enter `STORE_ID <> 1036` in the **Condition** field.

Data Validation Option does not check the condition syntax. Any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the PowerCenter syntax is not valid, a mapping installation error occurs when you run the test.

Enter the filter condition in the **Condition** field. Because the PowerCenter Integration Service processes the filter condition, it must use valid PowerCenter syntax. Do not include the WHERE keyword.

Operator

The operator defines how to compare the test result for the field with the constraint value. Enter an operator for aggregate, VALUE, and FORMAT tests.

The following table describes the operators available in the **Operator** field:

Operator	Definition	Description
=	Equals	Implies that the test result for the field is the same as the constraint value. For example, SUM(field) is the same as the constraint value.
<>	Does not equal	Implies that the test result for the field is not the same as the constraint value.
<	Is less than	Implies that the test result for the field is less than the constraint value.
<=	Is less than or equal to	Implies that the test result for the field is less than or equal to the constraint value.
>	Is greater than	Implies that the test result for the field is greater than the constraint value.
>=	Is greater than or equal to	Implies that the test result for the field is greater than or equal to the constraint value.
~	Is approximately the same as	Implies that the test result for the field is approximately the same as the constraint value. The approximate operator requires a threshold value. It only applies to numeric datatypes
Between	Is between two values entered	Implies that the test result for the field is between the two constants entered as the constraint value. This operator is generally used for numeric, string, or datetime datatypes. For example, to check if the EMP_ID is between 1000 and 2000, specify the following information in the single table test editor: - Select **VALUE** as the function. - Select EMP_ID as the field. - Select the between operator. - Specify `1000,2000` as the constraint value.
Not Between	Is not between two values entered	Implies that the test result for the field is not between the two constants entered as the constraint value. This operator is generally used for numeric, string, or datetime datatypes. For example, to check if the EMP_ID is not between 1000 and 2000, specify the following information in the single table test editor: - Select **VALUE** as the function. - Select EMP_ID as the field. - Select the not between operator. - Specify `1000,2000` as the constraint value.

Operator	Definition	Description
In	Is included in a list of values entered	Implies that the test result for the field is in the list of constants entered as the constraint value. For example, to check if the ITEM_NAME is in 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', specify the following information in the single table test editor: - Select **VALUE** as the function. - Select ITEM_NAME as the field. - Select the in operator. - Specify `'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife'` as the constraint value.
Not In	Is not included in a list of values entered	Implies that the test result for the field is not in the list of constants entered as the constraint value. For example, to check if the ITEM_NAME is not in 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', specify the following information in the single table test editor: - Select **VALUE** as the function. - Select ITEM_NAME as the field. - Select the not in operator. - Specify `'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife'` as the constraint value.

Note: Data Validation Option compares string fields using an ASCII table.

Related Topics:

- "BIRT Report Examples" on page 167

Constraint Value

The constraint value represents a constant value to which you want to compare the field values. For example, you might want to verify that all values in the ORDER_DATE field fall between January 1, 2020 and December 31, 2020. Or, you might want to verify that the minimum ORDER_ID number is greater than 1000.

The constraint value must be a string, numeric, or datetime constant. The datatype of the constraint value depends on the test.

The following table lists the constraint value datatype allowed for each test:

Test	Datatype
COUNT, COUNT_DISTINCT, COUNT_ROWS	Integer
MIN, MAX, SUM, VALUE	Same as the **Field** datatype.
FORMAT	String
AVG	Double
UNIQUE, NOT_NULL, NOT_BLANK	These tests do not use a constraint value.

Enter a constraint value in the **Constraint Value** field. Enter a constant or list of constants separated by commas.

The number of constants you enter as the constraint value depends on the operator you use:

Arithmetic operator such as =, <>, or ~

Enter a single constant.

Between or Not Between operator

Enter two constants separated by a comma.

In or Not In operator

Enter multiple constants separated by commas.

Enclose each string value, datetime value, or format pattern within single quotes. Datetime values must match the PowerCenter standard datetime format of MM/DD/YYYY HH24:MI:SS.

Remaining Controls on Test Editor

The remaining controls on the Single Table Test Editor are used in the same manner that they are used for table pairs.

Adding a Single-Table Constraint Test

You can add a test to a single table. Add a test to validate the values in the columns of the table.

1. Right-click the name of the single table in the Navigator, and select **Add Constraint Test**.

   The **Single Table Test Editor** dialog box opens.
2. Enter the test properties.
3. Click **Save**.

Test Generation for a Single-Table Constraint from a Spreadsheet

You can generate tests for a single table from a spreadsheet.

You might use a spreadsheet to define tests if you want to work offline. Or, you might use a spreadsheet to define tests if you are not a Data Validation Option user. For example, business analysts might need to create the test definitions because they are more familiar with the data.

To generate tests from a spreadsheet, compete the following steps:

1. Export existing single-table tests.
2. Add tests to the spreadsheet.
3. Import the tests from the spreadsheet.
4. Resolve tests with errors.

Export single-table tests to generate a spreadsheet that contains column headers, test metadata, and column metadata. If the single table already has tests defined, the spreadsheet also contains existing test definitions.

Note: You can find a spreadsheet template in the following Data Validation Option installation directory:

```
<Data Validation Option installation directory>\samples\SingleTable-template.xlsx
```

After you generate the spreadsheet, you can add tests to the spreadsheet. For each test, enter the test properties in the spreadsheet. The spreadsheet contains restricted values for particular test properties based on the column and test metadata.

You can then import the tests from the spreadsheet into Data Validation Option. When you import tests from a spreadsheet, Data Validation Option lists and validates the test definitions, and shows any errors. For example, an error appears if you use the approximate operator on a string column in a test definition. You cannot apply the approximate operator on string columns. You must resolve the errors or remove the test before you can import the test definitions. To resolve errors, edit the test definition.

Data Validation Option generates a test for each test definition specified in the spreadsheet. If you import a test that already exists, Data Validation Option generates a duplicate version of the test. Data Validation Option does not update existing tests or remove tests when you import tests from a spreadsheet.

Spreadsheet Requirements for Single-Table Tests

You can generate tests for a single table based on the test definitions specified in a spreadsheet.

Consider the following requirements for the spreadsheet:

- Define tests in the first worksheet of an Excel workbook. Data Validation Option reads the first worksheet and ignores other worksheets in an Excel workbook.
- The Excel file must have a .xls or .xlsx file extension.
- To define tests for multiple single-table constraints, define tests for each single-table constraint in a separate Excel workbook.
- You can only define one test per row.
- You must specify a field name or a field expression for each table in each row of the spreadsheet. If you specify a field name and field expression for the same table in a row, Data Validation Option uses the expression.
- If you select an operator that is not valid for a particular datatype and you try to import the tests from the spreadsheet, Data Validation Option displays an error.
- The values in the spreadsheet are case sensitive.

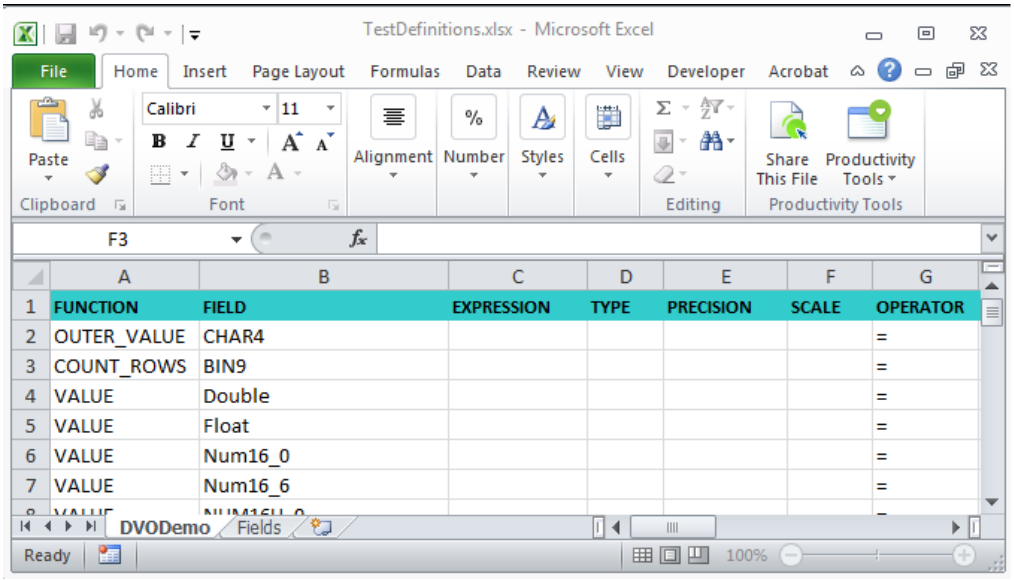
Generating Tests for a Single-Table Constraint from a Spreadsheet

Data Validation Option can generate single-table tests based on test definitions in a spreadsheet. To generate the tests, export the existing tests, add tests to the spreadsheet, and then import the tests from the spreadsheet into Data Validation Option.

1. In the Navigator, right-click the single table and select **Export Tests to Spreadsheet**.

   The **Save As** dialog box appears.
2. Save the spreadsheet to a location that the Data Validation Client can access.
3. Open the spreadsheet.

The spreadsheet shows the existing single-table constraint test definitions if tests exist.



	A	B	C	D	E	F	G
1	FUNCTION	FIELD	EXPRESSION	TYPE	PRECISION	SCALE	OPERATOR
2	OUTER_VALUE	CHAR4					=
3	COUNT_ROWS	BIN9					=
4	VALUE	Double					=
5	VALUE	Float					=
6	VALUE	Num16_0					=
7	VALUE	Num16_6					=

4. Add tests for the single table.

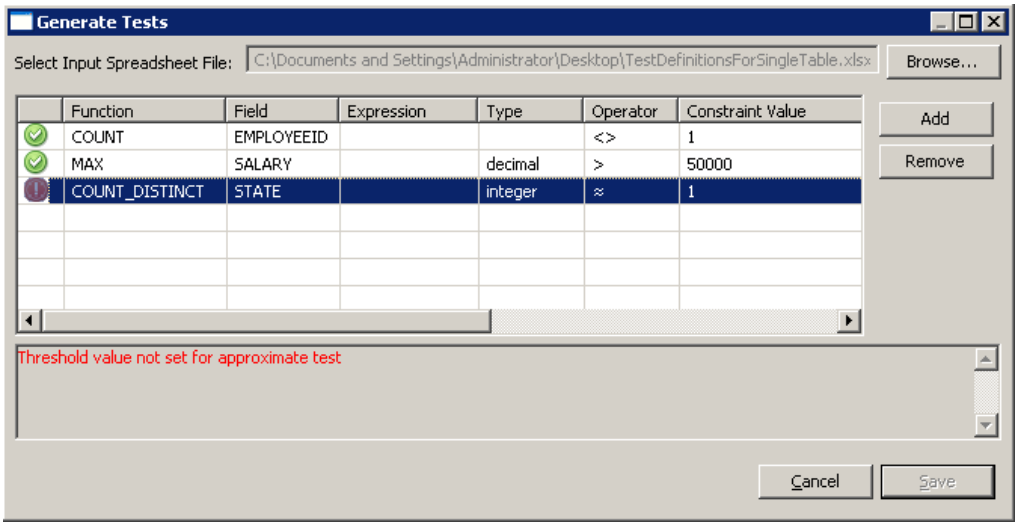
   Verify that the content in the spreadsheet meets the spreadsheet requirements.
5. Save and close the spreadsheet file.
6. In the Navigator, right-click the single table and click **Generate Tests from Spreadsheet**.

   The **Generate Tests** dialog box appears.
7. Click **Browse** and select the spreadsheet.

   A message specifies how many tests are imported.
8. Click **OK**.

   The **Generate Tests** dialog box previews the tests to be generated and shows any errors. A red circle icon indicates that the row has an error. A yellow triangle icon next to a cell indicates that the value of the cell has an error.
9. If a test has an error, select the test row with the error to preview the error.

   The error appears at the bottom of the **Generate Tests** dialog box.



10. Resolve the error or remove the test.
11. To change a test property, select the property in the **Generate Tests** dialog box and then make the change.
12. To remove a test, select the test row and click **Remove**.
13. To add a test, click **Add** and enter the test properties.
14. Click **Save** to import the tests.

Editing a Single-Table Constraint Test

To edit a test, right-click the test in the Navigator, and select **Edit Test**. The **Single Table Test Editor** dialog box opens.

Deleting a Single-Table Constraint Test

To delete a test, right-click the test in the Navigator, and select **Delete Test**.

Running Single-Table Constraint Tests

Before you run tests, verify that the Data Validation Option target folder is closed in the PowerCenter Designer and the PowerCenter Workflow Manager. If the target folder is open and you run tests, Data Validation Option cannot write to the target folder and the tests fail.

Use one of the following methods to run tests:

- Select one or more table pairs and click **Run Tests.**
- Right-click a folder in the Navigator and select **Run Folder Tests.**
- Right-click a test on the **Tests** tab and select **Run Selected Tests.**

Data Validation Option runs all tests for a single table together. You cannot run tests individually unless only one test is set up for the table. If you select an individual test, Data Validation Option runs all tests for the single table.

After you run a test, you can view the results on the **Results** tab.

Data Validation Option uses the following logic to determine whether a test passes or fails:

- An aggregate test is calculated as “value <operator> constraint.” If this relationship is true, the test passes. If the operator chosen is approximate, then the test is calculated as ABS(value-constraint) <= Threshold.
- A VALUE test must produce fewer or an equal number of records that do not match compared to the threshold value. If there is no threshold value and there are no records that do not match, the test passes.
- A FORMAT test is calculated as “value <operator> constraint.” If this relationship is true, the test passes.

- A UNIQUE, NOT_NULL, or NOT_BLANK test passes if the field value is unique, is not null, or is not blank, respectively. For string values, not blank means the string is not null or empty. For numeric values, not blank means the number is not null or 0.

Bad Records for Single-Table Constraint Tests

When you select a test on the Tests tab, the records that fail a test appear on the **Bad Records** view.

The Bad Records view displays a delimiter before and after each string value to help you identify whitespace in a record of string data type. Data Validation Option uses the following formula to calculate the percentage of difference: `(<larger value> - <smaller value>) ÷ <larger value> * 100`

Different columns appear on the**Bad Records** view based on the test.

Aggregate Tests

Aggregate tests do not display bad records. The **Results** view displays the test result value.

VALUE, FORMAT, NOT_NULL, and NOT_BLANK Tests

These tests display the following columns for each bad record:

- The key or expression for the field
- The field value

UNIQUE Tests

UNIQUE tests display the field values that are not unique.

If you upgrade to Data Validation Option release 9.6.0 from release 9.5.2 or earlier, you must reinstall a test if you want to view more details about bad records for the test.

Viewing the Details of a Bad Record

If you know what caused a record to fail, you can take corrective action on the data. You can view the details of a bad record for value tests, unique tests, and set tests in the Bad Records view.

1. Run a test.
2. In the **Tests** tab of the details area, click the test.

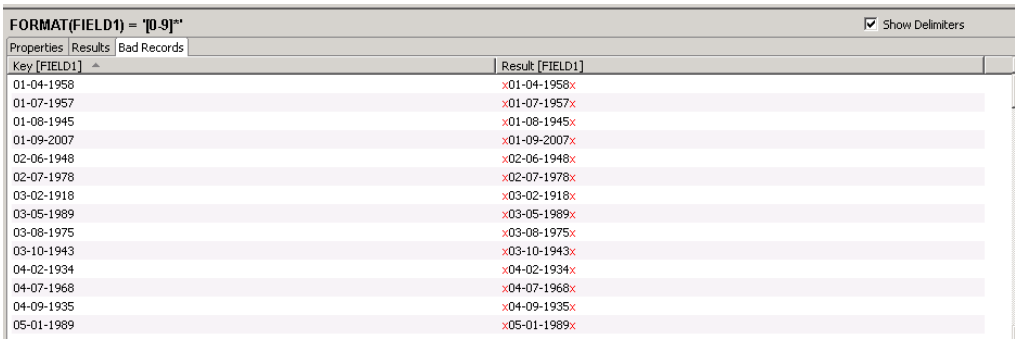
   If the test failed, the properties area includes a **Bad Records** view.
3. Click the **Bad Records** view.

   If the test evaluated strings, a **Show Delimiters** option appears in the upper-right corner of the properties area. If the test evaluated numerical values, a **Show Differences** option appears.

4. To view the details of a bad record, select the **Show Delimiters** or **Show Differences** option.

   Delimiters appear to help you identify whitespace in string data.

   The following image shows records with delimiters:



CHAPTER 11

Examples of Tests from Spreadsheets

This chapter includes the following topics:

Examples of Tests from Spreadsheets Overview

You can generate tests for table pairs and single tables based on test definitions in a spreadsheet. Define the test properties as columns in the spreadsheet and then import the test definition into Data Validation Option.

You might use spreadsheets to define test for the following reasons:

- Data Validation Option users can work offline when creating tests.
- Analysts who are familiar with the data and testing requirements but are not familiar with Data Validation Option can define the tests. A Data Validation Option user can then import the test definitions and create the tests in Data Validation Option.
- Data extraction, transformation, and loading (ETL) processes and testing specification documents are often defined as a source-to-target spreadsheet. Some of the definitions in those spreadsheets can often be repurposed for test creation.
- Large numbers of tests, particularly those that follow certain patterns, can be defined quickly in a spreadsheet with copy and paste and other basic spreadsheet features. Importing multiple tests into Data Validation Option can save a significant amount of time and effort over creating multiple tests in the Data Validation Client.

For example, a flat file contains 200 columns. Each column is defined as a string in the source definition. During the ETL, about 90 of the fields are converted to decimal and written to the final target table.

Testing this scenario would require creating 200 value tests, one for each column that is read and written to the target table. The value tests for the columns that are converted to decimal must contain the expression TO_DECIMAL (<fieldname>).

On the Data Validation Client, it would be tedious to manually create the 90 tests that include an expression. But by using basic cut and paste or search and replace capabilities within a spreadsheet, you can define these 90 tests in about one minute or less. And if, instead of 90 tests, there were 200 or even 900 similar

tests, the time to create them would be minutes at most. This is a big productivity increase over using the Data Validation Client.

Importing from a spreadsheet provides an alternative to the Data Validation Client that is very effective when a large number of similar tests must be generated.

The following examples show the how you can import tests from spreadsheets in different scenarios:

- Import and export of table pair tests
- Import of table pair tests for compare tables

Note: Each of the scenarios uses a table pair as an example, but analogous functionality and processes can be used for single tables as well.

Spreadsheet Import and Export of Table Pair Tests

Scenario 1 - Creating a blank spreadsheet to use for import

Although any spreadsheet with the proper column formats can be used with DVO, samples are provided at the following location:

<DVO client install dir>\samples\TablePair-template.xlsx

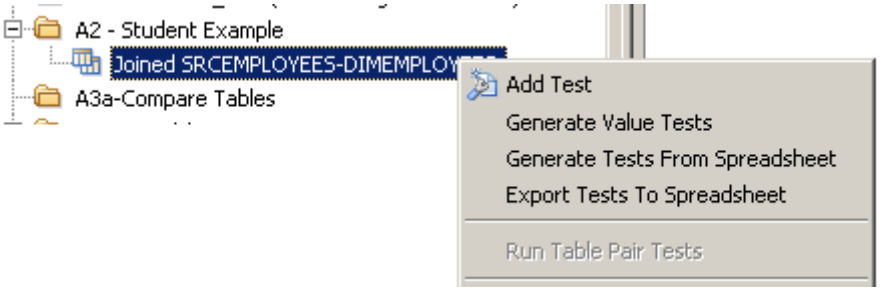
<DVO client install dir>\samples\SingleTable-template.xlsx

But, DVO also provides a simple way to export a pre-formatted spreadsheet with proper column headings AND column metadata specific to a particular table pair or single table. This makes it MUCH easier for a spreadsheet user to quickly define the tests they are interested in.

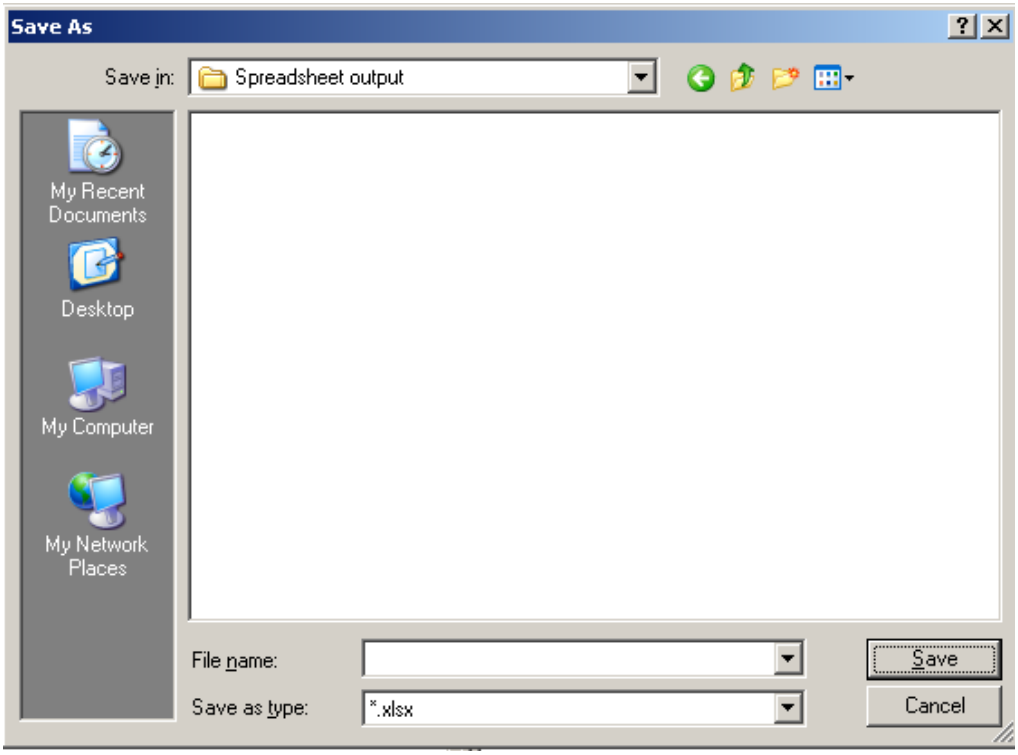
Creating a Blank Spreadsheet to Use for Import

To create a spreadsheet from an existing Table Pair, do the following:

1. Create a Table Pair that you want to use.
2. Ensure there is a Join Condition on the Table Pair.
3. Right click on the Table Pair and select *Export Tests to Spreadsheet* .

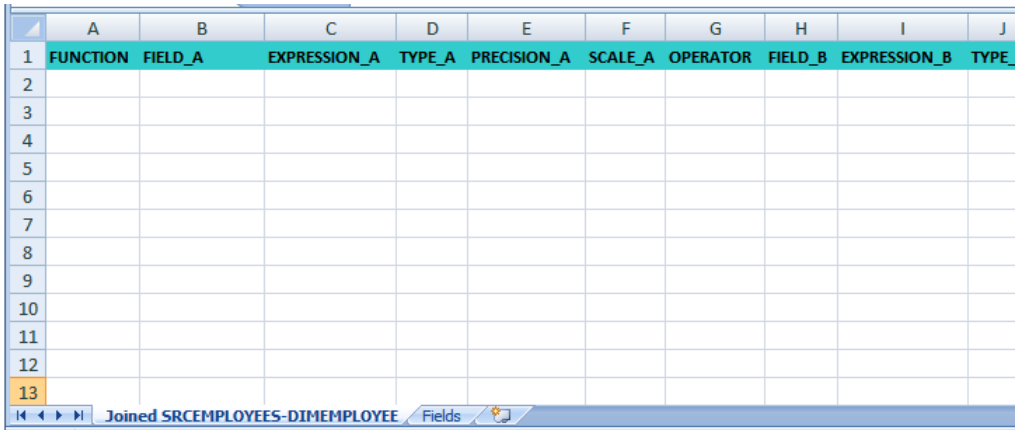


4. Select a location and name for the exported file and click Save.



The spreadsheet has been exported to your computer.

5. Open the spreadsheet you just generated in Excel. You should see the following:

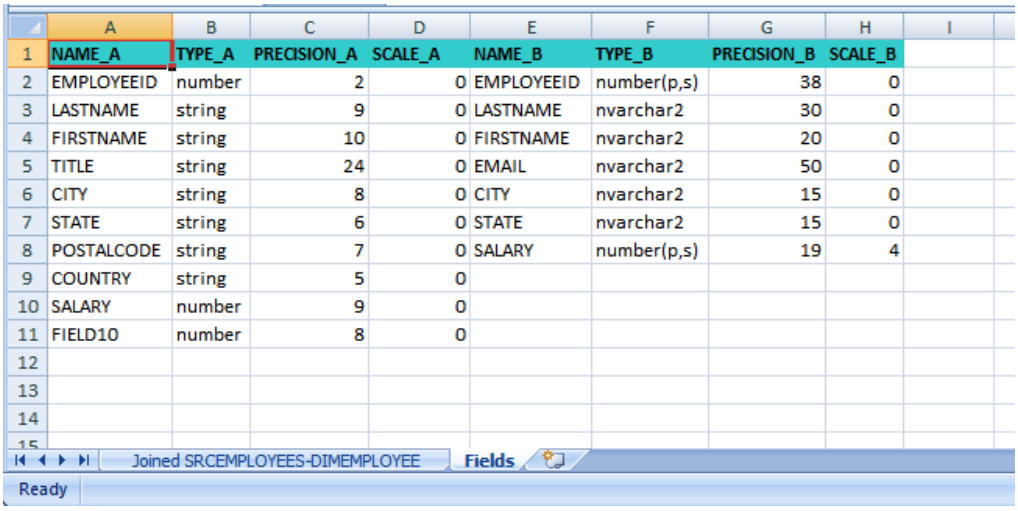


There are two worksheets. The first takes its name from the Table Pair that was used to generate the spreadsheet. This worksheet is empty because there were no tests defined in the Table Pair. If tests had been defined, then those would be visible - one test per row - in the worksheet.

This worksheet should have 18 columns defined. These correspond to fields that are present in the Add Test dialog in DVO.

Column	Title
A	Function
B	Field_A
C	Expression_A
D	Type_A
E	Precision_A
F	Scale_A
G	Operator
H	Field_B
J	Type_B
K	Precision_B
L	Scale_B
M	Case_Insensitive
N	Trim_Trailing_Spaces
O	NullEQNull
P	Threshold
Q	Max_Bad_Records

The second worksheet, Fields, contains the Field/Column metadata based on the Table A/B definitions in the Table Pair. This information is provided for reference purposes, and can be used when creating expressions or in other situations where the column metadata is required.



	A	B	C	D	E	F	G	H	I
1	NAME_A	TYPE_A	PRECISION_A	SCALE_A	NAME_B	TYPE_B	PRECISION_B	SCALE_B	
2	EMPLOYEEID	number	2	0	EMPLOYEEID	number(p,s)	38	0	
3	LASTNAME	string	9	0	LASTNAME	nvarchar2	30	0	
4	FIRSTNAME	string	10	0	FIRSTNAME	nvarchar2	20	0	
5	TITLE	string	24	0	EMAIL	nvarchar2	50	0	
6	CITY	string	8	0	CITY	nvarchar2	15	0	
7	STATE	string	6	0	STATE	nvarchar2	15	0	
8	POSTALCODE	string	7	0	SALARY	number(p,s)	19	4	
9	COUNTRY	string	5	0					
10	SALARY	number	9	0					
11	FIELD10	number	8	0					
12									
13									
14									
15									

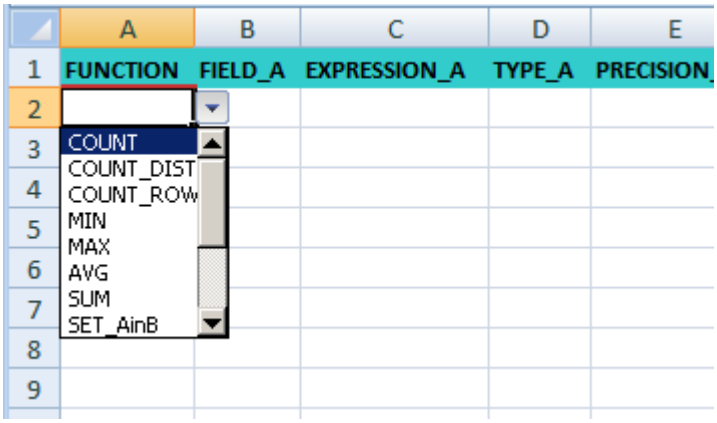
Joined SRCEMPLOYEES-DIMEMPLOYEE | Fields

Ready

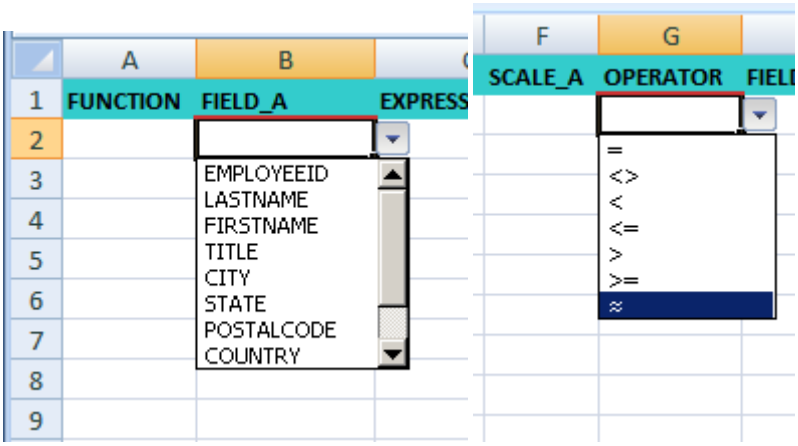
Scenario 2 - Create Tests in the Spreadsheet

The first worksheet contains all the column and test metadata (column names, test types, operators etc.) that is normally present in the GUI. This metadata is visible as dropdown menus when you click on a column.

For example, the *Function* column will show a drop down of all the Test types (e.g. Count, Value, Outer Value etc.) which are available. The Field_A (and Field_B) columns will show drop downs of the corresponding columns for Table A and Table B in the Table Pair etc.



	A	B	C	D	E
1	FUNCTION	FIELD_A	EXPRESSION_A	TYPE_A	PRECISION_
2					
3	COUNT				
4	COUNT_DIST				
5	COUNT_ROW				
6	MIN				
7	MAX				
8	AVG				
9	SUM				
	SET_AinB				

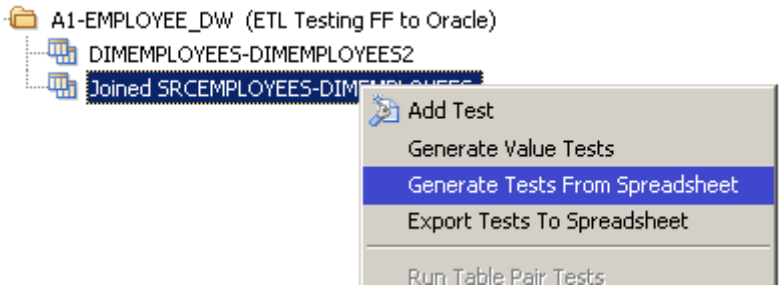


	A	B	F	G
1	FUNCTION	FIELD_A	SCALE_A	OPERATOR
2				
3		EMPLOYEEID		=
4		LASTNAME		<>
5		FIRSTNAME		<
6		TITLE		<=
7		CITY		>
8		STATE		>=
9		POSTALCODE		≈
		COUNTRY		

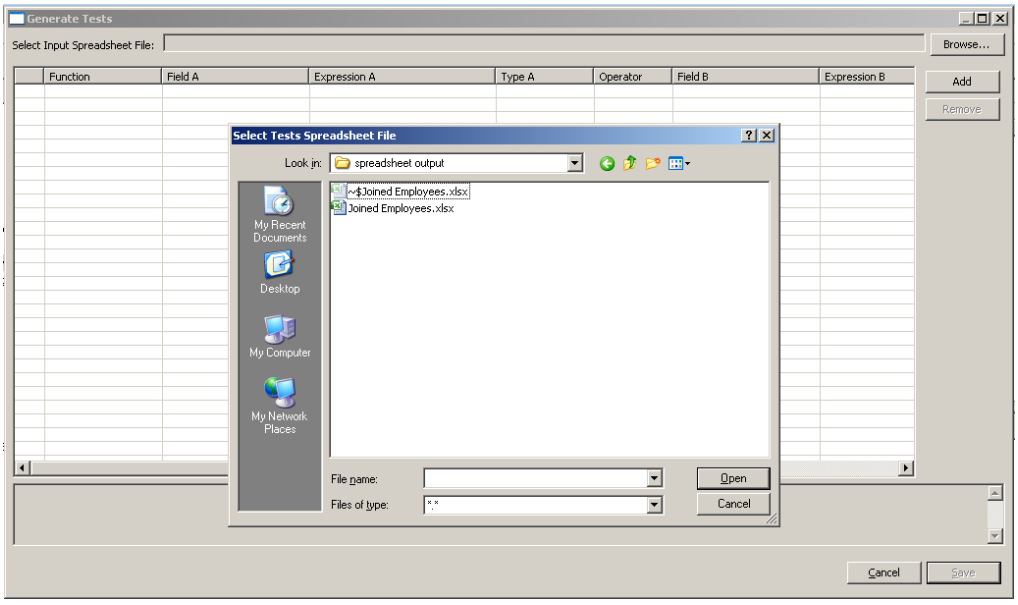
This information speeds test creation in the spreadsheet, helps reduce input errors and eliminates the need to memorize items (like Test types) or specific column names/types etc.

Scenario 3 - Importing the Spreadsheet into DVO

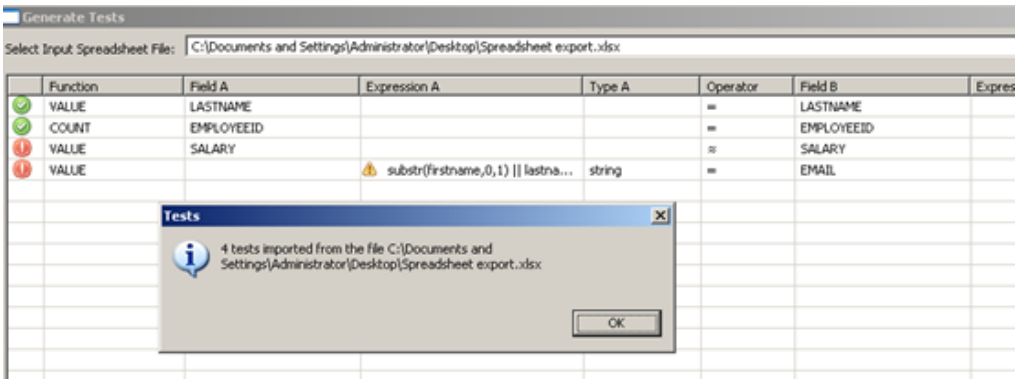
1. Open the DVO Navigator.
2. Right click on the same Table Pair used to create the empty spreadsheet and select *Generate Tests from Spreadsheet* .



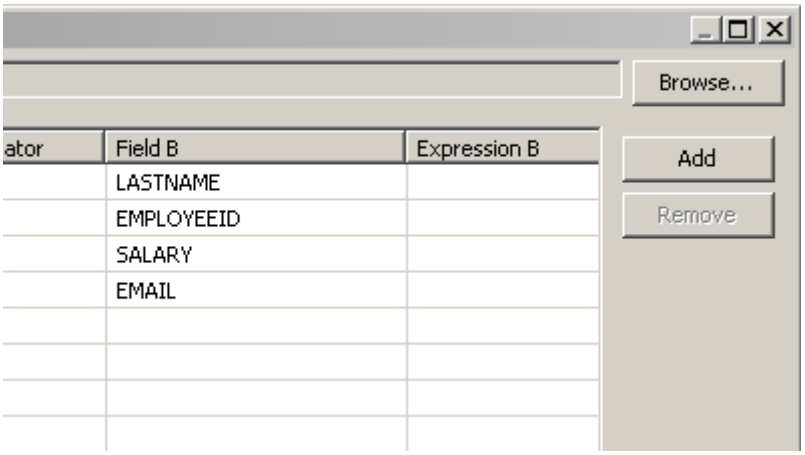
3. Using the dialog, click the *Browse* button and open the spreadsheet that you just saved.



4. The tests are imported and displayed in the dialog.
5. If you import a LOT of tests (e.g. many dozens or hundreds), the import process can take some time (e.g. 30-60 seconds or even longer) as the import process validates each row of the spreadsheet and identifies any errors.
6. These errors are highlighted in the import dialog with red/yellow icons.



7. Red icons on the left indicate the test in that row has errors.
8. Yellow icons on individual cells point out specific cells that require attention.
9. Correct the errors found during the import. In the image above, the last 2 tests have errors in them that need to be addressed.
10. You can add and remove tests using this dialog by clicking on the *Add* and *Remove* buttons on the right.



11. This provides you with additional controls when correcting imported test definitions.
12. Click *Save* to generate the tests. Inspect the generated tests to verify that tests imported correctly.
13. Note that all tests imported will be added to the Table Pair, even IF, the Table Pair already contains tests with the same definitions. i.e. duplicate checking is not performed when creating the new tests from the spreadsheet metadata.

    Note: Although, this is a very simple example to illustrate how to use the spreadsheet import functionality, the real power of this functionality is when many dozens or hundreds of tests are required, and basic spreadsheet capabilities such as copy/paste or simple string expressions can be used in the spreadsheet to quickly create large numbers of tests.

Spreadsheet Import for Compare Tables

The Compare Tables dialog has always provided an efficient way to mass create Table Pairs and their tests based on table metadata. This is very useful when testing scenarios such as product upgrades or migrations

where the 2 data sets being tested are virtually the same. That is, same table/data structures, and in theory, same data.

One limitation of Compare Tables was that the Table Pair generation process assumed that Table names must be identical across the Table A/B metadata, and there was no way for a user to over-ride that.

While that works for certain scenarios, there are many cases where table structures are the same, but the names are not. For example in a migration, table names may change or in a replication, tables may be prefixed (or suffixed) across source/target to distinguish them.

The Spreadsheet Import for Compare Tables allows you to use the Compare Tables functionality but provide simple metadata for Tables when the names don't exactly match.

There are two test scenarios for the Spreadsheet import for Compare Tables:

1. Adding Tables to the Compare Tables Import spreadsheet

2. Importing the Compare Tables spreadsheet into DVO

Each scenario is explained in detail in the following sections.

Scenario 1 - Add tables to the Compare Tables Import Spreadsheet

The Compare Tables import spreadsheet is much simpler than the Table Pair import spreadsheet. It has the following 7 columns.

Column	Title
A	Description
B	External_ID
C	Table_A
D	Table_B
E	Join_A
F	Join_B
G	Comments

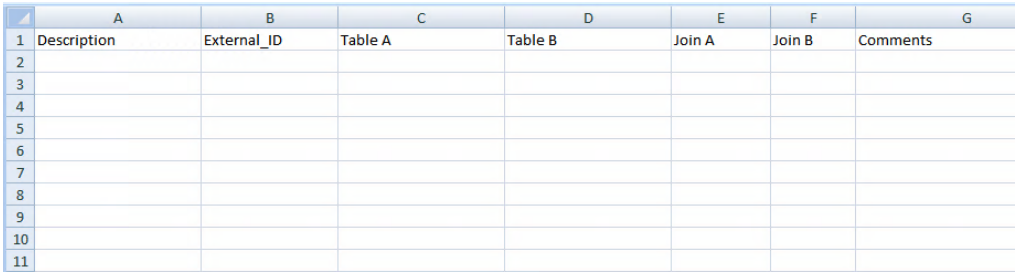
Note: Unlike the Table Pair example earlier in this document, you cannot create an export spreadsheet for Compare Tables from the GUI. Given that it only has 7 columns, it can be easily recreated manually. Informatica also provides a sample spreadsheet in the following directory on the client machine.

<DVO client install dir>/samples/CompareTables-template.xlsx

Of these 7 columns, only Columns C and D (Table_A, Table_B) are mandatory. The others are optional.

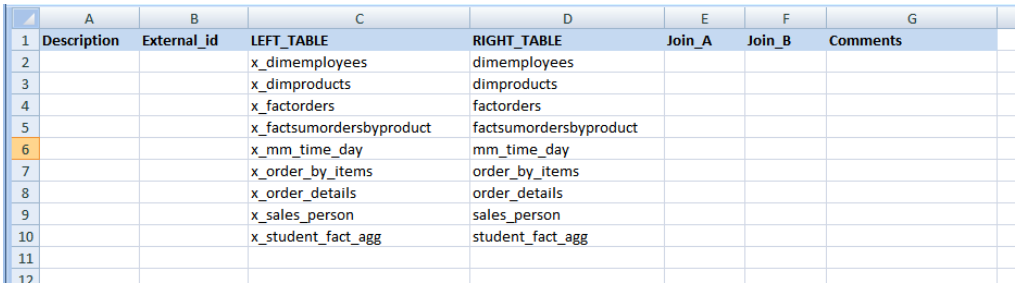
Adding Tables to the Compare Tables Import Spreadsheet

1. Open the Compare Tables spreadsheet.



	A	B	C	D	E	F	G
1	Description	External_ID	Table A	Table B	Join A	Join B	Comments
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

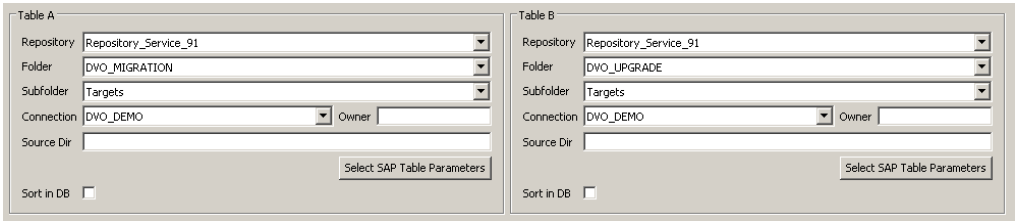
2. Fill in the necessary spreadsheet columns to define the Table Pairs you need to create.



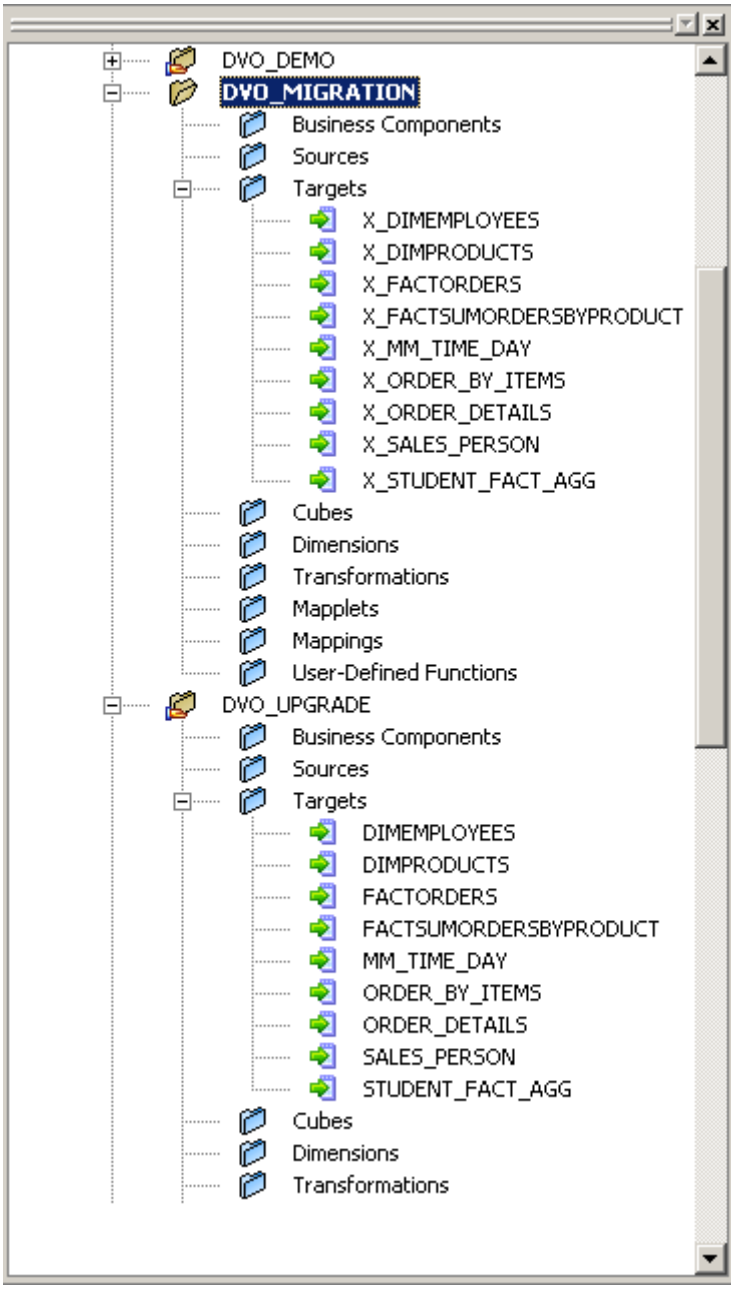
	A	B	C	D	E	F	G
1	**Description**	**External_id**	**LEFT_TABLE**	**RIGHT_TABLE**	**Join_A**	**Join_B**	**Comments**
2			x_dimemployees	dimemployees			
3			x_dimproducts	dimproducts			
4			x_factorders	factorders			
5			x_factsumordersbyproduct	factsumordersbyproduct			
6			x_mm_time_day	mm_time_day			
7			x_order_by_items	order_by_items			
8			x_order_details	order_details			
9			x_sales_person	sales_person			
10			x_student_fact_agg	student_fact_agg			
11							
12							

Note: In this spreadsheet only the table names have been entered (i.e. the minimum), but you can enter as much metadata as needed including Description, External_ID, Join_A/B fieldsand Comments.

3. Once the spreadsheet is complete, save it.
4. Remember that tables with these names need to be defined in the folders that will be selected in the *Compare Tables* dialog:
5. e.g. in the folders specified here

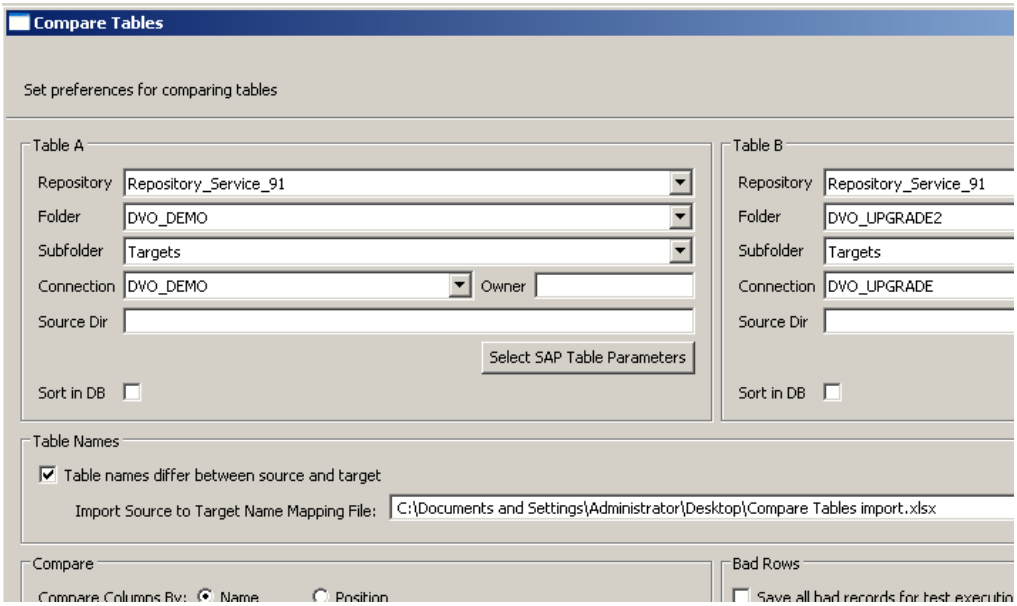


6. This image shows the tables (Targets) as they are defined in the PowerCenter repository. The folders DVO_MIGRATION and DVO_UPGRADE are the folders specified in the Compare Tables dialog.



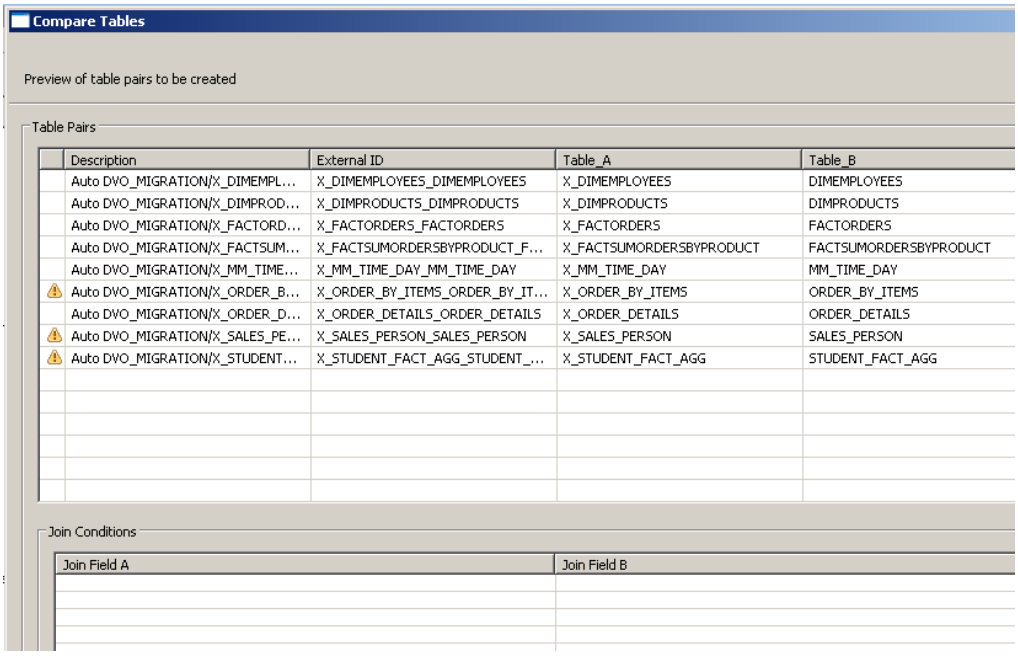
Scenario 2 - Importing the Compare Tables Spreadsheet into DVO

1. In DVO, open the Compare Tables dialog and fill in the appropriate values at the top for Table A and Table B.



2. Select the *Table names differ between source and target* check box.
3. Select the spreadsheet that has the table names you want to use in the Table Pairs.
4. Complete the rest of the Compare Tables dialog as needed and click Next.

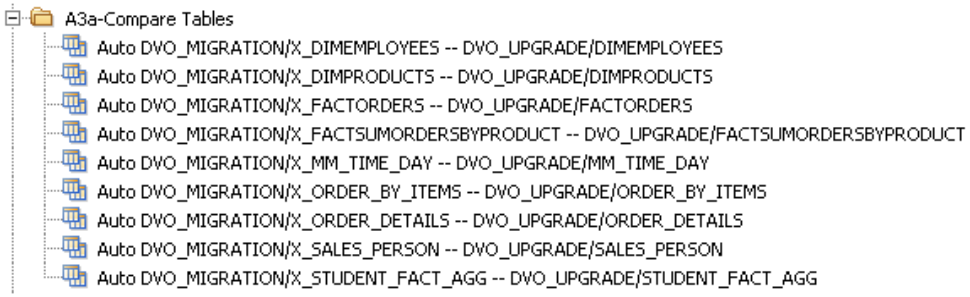
   DVO imports the metadata from the spreadsheet, defines the Table Pairs and displays them in the dialog box.



The Table Pairs have been defined (but not yet created), and error checking has been performed.

For those table pairs with a warning, click on them and fix the problems identified at the bottom of the dialog.

5. When all issues are addressed, click Finish to generate the Table Pairs.



CHAPTER 12

SQL Views

This chapter includes the following topics:

SQL Views Overview

Create an SQL view to produce a set of fields that you can use as a table in a single table or table pair. In an SQL view definition, you specify an SQL statement that can combine multiple sources, calculate field values, and filter columns and rows.

You can create an SQL view based on one or more relational tables that exist in the same database.

When you specify the SQL statement for an SQL view, you can add database functions, any join type, subqueries, and stored procedures in the SQL statement.

When you run a single-table test or table-pair test based on an SQL view, the database processes the SQL statement defined for the SQL view.

SQL View Properties

You specify the SQL view properties when you create an SQL view.

The following table describes the SQL view properties:

Property	Description
Description	SQL view description.
Table Name	Name of the table included in the SQL view. All tables you use in the SQL view must exist in the same database.
Alias	Alias name for the table.
Connection	PowerCenter connection for the tables.
Column Definition	The columns that make up the SQL view. Data Validation Option imports all columns from the tables you select. You can create, delete, and rearrange columns.
SQL Statement	SQL statement you run against the database to retrieve data for the SQL view.
Comment	Information about an SQL view. Data Validation Option displays the comment when you view the SQL view in the Properties area.

Description

Enter a description so you can identify the SQL view. Data Validation Option displays the description in the Navigator and on the SQL Views tab. The description can include spaces and symbols.

Table Definitions and Connection

To provide Data Validation Option with the information it needs to create an SQL view, you must specify the tables that the SQL statement is based on and the corresponding database connection. When you provide the tables and connection information, Data Validation Option can access the metadata that is necessary for the view to function correctly.

To add a table, click **Add Table**. The Choose Data Source dialog box opens. This dialog box displays all of the relational tables available in the repositories. You can sort information in this dialog box by clicking the column headers. You can reduce the number of items to select by typing one or more letters of the table, file, or view name in the **Search** field. Select a table and click **Select**. All of the tables you use in an SQL view must exist in the same database.

If you identify the table with an alias in the SQL statement you use to create the view, enter the alias name next to the table name.

When you finish adding tables, select the PowerCenter connection for the tables from the **Connection** list.

Column Definition

After you specify the tables on which the SQL view is based, you must specify the columns that make up the view.

Add the columns from each table to the SQL view. Delete the columns that you do not want to use. You can rearrange the columns in the view.

Note: The number and order of columns you define for the view must match the SQL statement.

You can also create a column. To do this, select **Insert Column**. Enter the column name and specify the datatype, precision, and scale for the column. The datatype, precision, and scale information must match the transformation datatype that the PowerCenter Integration Service uses for the column. The scale must be zero for datetime, string, and integer datatypes.

SQL Statement

Enter an SQL statement to retrieve data for the SQL view.

Because the database runs the SQL statement, you must use valid database syntax when you enter the SQL statement. The number and order of the columns in the SQL statement and column definition list must match. The datatypes of the columns must be compatible.

To avoid errors when you run tests, test the SQL statement in the database before you paste it into the **SQL Statement** field. Data Validation Option does not validate the SQL statement syntax.

Comment

You can associate a comment with the view. Data Validation Option displays the comment when you view the SQL view in the Properties area.

Rules and Guidelines for SQL Views

Certain rules and guidelines apply when you create an SQL view.

Consider the following rules and guidelines when you create an SQL view:

- All tables in the SQL view must exist in the same database.
- The SQL statement must use valid SQL syntax based on the database of the tables included in the SQL view. Data Validation Option does not validate the SQL statement syntax.
- The number and order of the columns must match in the column definitions and the SQL statement of the SQL view.
- The scale must be zero for datetime, integer, and string datatypes.
- If you create a column in the SQL view, verify that the datatype, precision, and scale match the PowerCenter datatype, precision, and scale shown in the **Column Definitions** area.
- You can add database functions, any join type, subqueries, and stored procedures in the SQL statement of an SQL view.
- If the SQL view calls a stored procedure, the connection assigned to SQL view must have Execute permission on the stored procedure.
- If you include the SQL view in a table pair, you can create a WHERE clause for the table pair based on a field in the SQL view.
- When you use SQL view as the data source, you can edit the connection in the table pairs, single tables, aggregate views, and join views.

Adding an SQL View

You can add a SQL view to create a view on multiple tables in the same database. You can add a WHERE clause to filter the records from each table and configure which columns appear in the SQL view.

1. Click **File** > **New** > **SQL View**.

   The **SQL View Editor** dialog box appears.
2. Enter a description for the SQL view.
3. Click **Add Table** in the **Table Definitions** area.

   The **Choose Data Source** dialog box appears.
4. Select the table and click **Select**.

   To filter the available tables, enter a search string in the **Search** field.
5. Select at least one more table to include in the SQL view.
6. To change a table alias name, double click the table alias name in the **Alias** column.

   By default, the table alias name is the same as the table name.
7. To delete a table from the SQL view, select the table and click **Delete Table**.
8. Select a connection to the database that contains the tables included in the SQL view.
9. To add columns from the tables to the SQL view, click **Select Columns** in the **Column Definition** area.

   The **Output Fields** dialog box appears.
10. To add columns to the SQL view, add the columns in the **Available Fields** pane to the **Selected Fields** pane.
11. Click **OK**.

    The columns appear in the **Column Definition** area of the **SQL View Editor** dialog box.
12. To create a custom column, click **Insert Column** in the **Column Definition** area.

    The **SQL View Column Definition Editor** dialog box appears.
13. Enter the name, datatype, precision, and scale for the custom column, and then click **Save**.

    The custom column appears in the **Column Definition** area of the **SQL View Editor** dialog box.
14. To change the order of the columns, select a column and click **Move Up** or **Move Down**.
15. Enter the SQL statement that creates the SQL view based on the columns in the **Column Definition** area of the **SQL View Editor** dialog box.
16. Enter a comment for the SQL view.
17. Click **Save**.

Editing SQL Views

To edit an SQL view, right-click the SQL view in the Navigator or on the **SQL Views** tab, and select **Edit SQL View**. The **SQL View Editor** dialog box opens.

Deleting SQL Views

To delete an SQL view, right-click the SQL view in the Navigator or on the **SQL Views** tab, and select **Delete SQL View**. You can also select the SQL view and press the Delete key.

When you delete an SQL view, Data Validation Option deletes all table pairs, single tables, and tests that use the SQL view.

SQL View Example

You want to verify that customers have larger transactions at your store than last quarter. To verify this, you create and run a single-table test based on an SQL view.

To determine if customers have larger transactions, you must verify that the average transaction amount is greater than $100, which was the average transaction amount last quarter. You get this information from transaction tables in an Oracle database.

You create an SQL view based on the transaction tables. Next, you create a single table based on the SQL view. Finally, you create a single-table test to compare the average transaction amount against the $100 constraint value.

Tables and Columns

The following table lists the transaction tables and columns that are included in the SQL view:

Table	Columns
ORDERS	- ORDER_ID - CUSTOMER_ID - PRODUCT_ID - PRICE_PER_UNIT - UNITS_SOLD
CUSTOMERS	- CUSTOMER_ID - LAST_ORDER_DATE
PRODUCTS	- PRODUCT_ID

Creating the SQL View

Complete the following tasks when you create the SQL view:

1. Enter Sales_SQLView as the description.
2. Add ORDERS, CUSTOMERS, and PRODUCTS as the tables in the SQL view.
3. Select the connection for the Oracle database that contains the transaction tables.
4. Add the required columns from each table in the SQL view definition.
5. Enter the following SQL statement in the SQL view definition:

```
SELECT AVG(O.PRICE_PER_UNIT), C.CUSTOMER_ID, C.LAST_ORDER_DATE, P.PRODUCT_ID,
O.UNITS_SOLD, O.PRICE_PER_UNIT, (O.UNITS_SOLD*O.PRICE_PER_UNIT) AS
TRANSACTION_AMOUNT
FROM CUSTOMERS C, ORDERS O, PRODUCTS P
WHERE (O.CUSTOMER_ID=C.CUSTOMER_ID AND P.PRODUCT_ID=O.PRODUCT_ID AND C.customer_ID
```

```
IN (SELECT UNIQUE(C.CUSTOMER_ID) FROM CUSTOMERS C))
GROUP BY  C.CUSTOMER_ID, C.LAST_ORDER_DATE, P.PRODUCT_ID
```

Creating the Single Table and the Test

Create a single table based on the SQL view. Next, add an AVG test to the single table based on the TRANSACTION_AMOUNT column in the SQL view. In the test, create a test condition where the average transaction amount exceeds 100.

Running the Test

Run the test. If the test passes, the average transaction amount exceeds $100. Customers made larger transactions than last quarter.

CHAPTER 13

Lookup Views

This chapter includes the following topics:

Lookup Views Overview

Data Validation Option lookup views allow you to test the validity of the lookup logic in your transformation layer.

Lookup views allow you to validate the process of looking up a primary key value in a lookup or reference table based on a text value from a source, and then storing the lookup table primary key in the target fact table. For example, a product name in the source system might be in a dimension that serves as the lookup table. The data transformation process involves looking up the product name and placing the primary key from the lookup table in the target fact table as a foreign key. You must validate the product name in the source table against the foreign key in the target table.

The following table lists the keys used in the example:

Source Table	Lookup Table	Target Table
source_id product_name	lookup_id product_name	target_id source_id lookup_id

The source table product name field is found in the lookup table. After the product name is found, the primary key from the lookup table is stored in the target table as a foreign key.

To test the validity of the lookup table foreign key in the target table, complete the following tasks:

1. Create the lookup view. Add the source table and the lookup table to the lookup view. Then create a relationship between the product name in the source and lookup tables.

2. Create a table pair with the lookup view and the table that is the target of the data transformation process. Join the tables on the source table primary key, which is stored in the target table as a foreign key.
3. Create an OUTER_VALUE test that compares the primary key of the lookup table to the lookup ID that is stored as a foreign key in the target table.

The OUTER_VALUE test checks the validity of the lookup table primary key stored in the target table against the contents of the source table. The test also finds any orphans, which are records in the target table that do not match any records in the lookup table.

Lookup View Properties

Lookup view properties describe the properties of source and lookup tables.

You can view lookup view properties by selecting a lookup view in either the Navigator or on the **Lookup Views** tab and viewing the properties. Most properties come from the values entered in the **Lookup View Editor** dialog box. Other properties come from the tests set up for and run on the lookup view.

Edit lookup view properties in the **Lookup View Editor** dialog box when you add or edit a lookup view.

The following table describes the lookup view properties:

Property	Description
Table	Name of source table or lookup table. For a source table, you can use a table from any type of data source, except Salesforce, SAP, and SAS. For a lookup table, you can use a table from any type of data source, except Salesforce, SAP, SAS, and XML.
Conn	PowerCenter connection for the source table or lookup table.
Override Owner Name	Override for the owner or name of the source table.
XML Group	XML group name for an XML data source. An XML group is the parent element of one or more child elements in an XML file. Only child elements of the XML group are available to use as lookup view columns.
Description	Lookup view description.
Source Fields	Source field on which to join the source table and the lookup table.
Lookup Fields	Lookup field on which to join the source table and the lookup table.

Selecting Source and Lookup Tables

A lookup view consists of a source table and a lookup table. Use the **Browse** button and the **Select Data Sources** dialog box to select the source and lookup table in the same way that you select tables for table pairs.

Selecting Connections

Select the correct connections for the source and lookup tables in the same way that you select connections for table pairs.

Overriding the Owner or Table Name

You can override the owner name or table name for the source table in a lookup view. You may need to override the owner name if the table exists in a different schema. You may need to override the table name if the table name changes or you want to use a different table that has the same table metadata.
To perform the override, enter the new owner name, table name, or both in the Override Owner Name connection property for the source table. When you enter the new owner name or table name, you must use a valid syntax.

Note: You must have Execute permission on the owner name to override the owner name.

The following table specifies the required syntax for each type of override:

Type of Override	Syntax
Change owner name.	<Owner Name>
Change table name.	/<Table Name>
Change owner name and table name.	<Owner Name>/<Table Name>

Description

Data Validation Option automatically generates the description for a lookup view based on the tables you select. You can change the description.

Source-to-Lookup Relationship

When you create the source-to-lookup relationship, you select the fields on which you want to join the source and lookup tables.

The fields included in a join condition must have compatible datatypes, precisions, and scales. For example, you can join an INT field to a DECIMAL field, but not to a DATETIME or VARCHAR field. Data Validation Option supports joins on fields with the following datatypes: datetime, numeric, and string. Joins are not allowed on binary/other datatypes.

Adding Lookup Views

You can use a lookup view to validate data in a target table. The lookup view includes fields from the source table and lookup table. You join the tables based on fields in the tables or expressions. Data Validation Option precedes the source table field names with "S_." The lookup view stores the primary key of the lookup table as a foreign key.

1. Right-click **Lookup Views** in the Navigator and select **Add Lookup View**.

   The **Lookup View Editor** dialog box opens.
2. Select the source table, lookup table, and their connections.

   Select the lookup field to look up in the lookup table. You can use an expression for join fields in a lookup view.
3. Enter the description for the lookup view.

4. Create the source-to-lookup relationship.

   Select **Expression** to create an expression for the source or lookup field. If you enter an expression, you must specify the datatype, precision, and scale of the result. The datatype, precision, and scale of the source and lookup fields must be compatible. Use the PowerCenter expression syntax in the source field. Use database-specific SQL in the lookup field. You can validate the PowerCenter expression syntax.

5. Click **Delete Join** to delete the selected source-to-lookup relationship.
6. Click **Save**.

Editing Lookup Views

To edit a lookup view, right-click the lookup view in the Navigator or on the **SQL Views** tab, and select **Edit Lookup View**. The **Lookup View Editor** dialog box opens. You cannot modify the sources in a lookup view. You can modify the lookup relationship.

Deleting Lookup Views

To delete a lookup view, right-click the lookup view in the Navigator or on the **Lookup Views** tab, and select **Delete Lookup View**. You can also select the lookup view and press the Delete key.

When you delete a lookup view, Data Validation Option deletes all table pairs and tests that use the lookup view.

Lookup Views Example

Use a lookup view to test the validity of the foreign key stored in the target or fact table, and to confirm that there are no orphans.

The following table displays sample data from a source table:

ORDER_ID	PRODUCT_NAME	AMOUNT
101	iPod	100
102	Laptop	500
103	iPod	120

The following table displays sample data from a lookup table:

LKP_PRODUCT_ID	LKP_PRODUCT_NAME
21	iPod
22	Laptop

The following table displays sample data from a target table:

TARGET_ID	ORDER_ID	LKP_PRODUCT_ID	AMOUNT
1	101	21	100
2	102	22	500
3	103	21	120

To test the validity of the lookup table foreign key in the target table, perform the following steps:

Create the lookup view.

Create a lookup view with the source and lookup tables. The lookup relationship uses the product name fields in both the source and the lookup tables. The lookup view includes the following fields:

- `S_ORDER_ID`
- `S_PRODUCT_NAME`
- `S_AMOUNT`
- `LKP_PRODUCT_ID`
- `LKP_PRODUCT_NAME`

The columns that originate in the source have the prefix "S_."

Create the table pair.

Create a table pair using the lookup view and the target table. Create a join relationship between the primary key of the source table and the corresponding foreign key in the target table as follows:

```
S_ORDER_ID and ORDER_ID
```

Create an OUTER_VALUE test.

Create an OUTER_VALUE test. Compare LKP_PRODUCT_ID in both the lookup table and the target table as follows:

```
LKP_PRODUCT_ID and LKP_PRODUCT_ID
```

CHAPTER 14

Join Views

This chapter includes the following topics:

Join Views Overview

A join view is a virtual table that contains columns from related heterogeneous data sources joined by key columns.

Use a join view to run tests on several related columns across different tables. You can create a join view instead of multiple SQL views with joins. For example, the Employee table has employee details, Inventory table has sales details, and the customer table has customer details. If you create a join view with the tables, you can obtain a consolidated view of the inventory sold by the partner and the revenue generated by the employees associated with the partners. You can run tests with the join view to validate data across the tables.

You can create a join view with different types of data sources. For example, you can create a join view with a flat file, an SAP table, an Oracle table, and an XML file.

You can add a join view in a single table or a table pair. You can then create tests with the table pair or single table to validate the data in the join view. Add multiple data sources to the join view and add join conditions to define the relationship between data sources.

Join View Data Sources

You can create a join view with tables from multiple data sources.

You can use the following data sources when you create a join view:

- Flat files
- IBM DB2

- Microsoft SQL Server
- Netezza
- Oracle
- PowerExchange for DB2/zOS
- Salesforce.com
- SAP R/3
- SAS
- SQL view
- XML files
- Sybase
- Teradata
- Salesforce
- DB2 AS/400
- DB2 z/OS
- IMS
- ADABAS
- Mainframe flat files
- VSAM

Join View Properties

Join view properties include table definitions and join conditions for each of the data source.

The following table describes the join view properties:

Property	Description
Description	Description of the join view.
Order	Sequence of data sources in the join view. You can join a data source with any of the preceding data sources.
Table Name	Data source that you add in the join view.
Alias	Alias name for the table.
Join type	Type of join used to join the data sources.
Where Clause	The WHERE clause to filter rows in the data source.
Left field	Left field of the join condition. Left field is the key column of the data source to which you create the join.
Right field	Right field of the join condition. Right field is the key column of the data source for which you create the join.

Alias in Join View

Alias of the data source in a join view helps you identify data sources that share the same name.

By default, Data Validation Option assigns the data source name as the alias name. If you select a data source with the same name as any other data source in the join view, Data Validation Option appends the alias name with a number.

Note: If you edit alias name after you a create a join view, Data Validation Option deletes the join conditions. Create the join conditions again to save the join view.

Join Types

You can choose join types to link fields in different data sources.

You can create the following types of joins between two tables in a join view:

Inner join

Inner join creates a result table by combining column values in two tables A and B based on the join condition. Data Validation Option compares each row in A with each row in B to find all pairs of rows that satisfy the join condition.

Left outer join

Left outer join between tables A and B contains all records of the left table A, even if the join condition does not find any matching record in the right table B. The resulting join contains all rows from table A and the rows from table B that match the join condition.

Right outer join

Right outer join between tables A and B contains all records of the right table B, even if the join condition does not find any matching record in the left table A. The resulting join contains all rows from table B and the rows from table A.

Full outer join

A full outer join contains all the rows from tables A and B. The resulting join has null values for every column in the tables that does not have a matching row.

Join Conditions

You must configure join conditions for all the data sources in the join view.

When you configure a join condition, select the data source in the table definition list and specify the left and right fields of the join condition. Data Validation Option displays the alias names of the data sources. The left field of the join condition consists of the output fields from the alias name you choose. The right field consists of the output fields of the data source for which you configure the join condition.

You can create a join condition with any of the data sources in the previous rows in a join view. You can create multiple join conditions for the same data source. You cannot save a join view unless you create at least one valid join condition for all the data sources.

Rules and Guidelines for Joins

Certain rules and guidelines apply when you create a join in a join view.

Use the following rules and guidelines when you create a join:

- The fields included in a join condition must have compatible datatypes, precisions, and scales. For example, you can join an INT field to a DECIMAL field, but not to a DATETIME or VARCHAR field.
- Data Validation Option supports joins on fields with the following datatypes: datetime, numeric, and string. Joins are not allowed on binary/other datatypes.
- For an XML data source, only child elements of the selected XML group appear as possible join fields.
- You can create multiple join conditions.
- If you specify a file list connection and the fields in the join condition are not unique among all files in the file list, the join fails.

Database Optimization in a Join View

You can optimize the data sources in a join view for better performance. You can choose to select a subset of data in the data source and aggregate the rows for a relational data source.

To improve read performance of the join view, you can provide a WHERE clause. The WHERE clause ensures that the data source uses a subset of data that satisfies the condition specified in the WHERE clause.

Data Validation Option does not check the WHERE clause syntax. If the PowerCenter Integration Service executes the WHERE clause, any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the PowerCenter syntax is not valid, a mapping installation error occurs.

Use PowerCenter expression in cases where you do not push down the WHERE clause in to the data source.

Use the following guidelines if the data source executes the WHERE clause:

- Relational data source. The WHERE clause must be a valid SQL statement. If the SQL statement is not valid, a runtime error occurs.
- SAP data source. The WHERE clause must be a valid SAP filter condition in the ERP source qualifier.
- Salesforce data source. The WHERE clause must be a valid SOQL filter condition.
- SAS data source. The WHERE clause must be a valid Where clause Overrule condition in the SAS source qualifier.

You can choose one of the following optimization levels when you configure a data source in a join view:

- Default. Data Validation Option converts all test logic to a PowerCenter mapping and applies sorting to the data source.
- WHERE clause, Sorting, and Aggregation in DB. Data Validation Option pushes the WHERE clause, sorting logic for joins, and all aggregate tests to the database. Data Validation Option converts all other test logic to a PowerCenter mapping. You can choose this option with relational data sources.
- Already Sorted Input. PowerCenter mapping does not sort the input. Ensure that you sort the data so that the tests run successfully.

Output Field Properties

When you create a join view, you can specify the fields to expose in single-table constraints and table pairs that use the join view.

The following table describes the output field properties:

Field

Name of the view column. By default, the view column name is a concatenation of the table name, an underscore character, and the table column name.

Alias

Alias name for the column. Default is no alias name.

Include as Output Field

Indicates whether the view column is exposed in single-table constraints and table pairs that use the join view.

Column Aliases

You can create an alias for the columns of a join view. You might create a column alias to shorten or change the column name in the join view.

You might need to shorten the view column name if the length causes a database error when PowerCenter processes a test based on the join view. By default, Data Validation Option appends the table name to the table column name to generate the view column name.

You might change the name of the view column to make the column name more descriptive. Or, to automatically generate table-pair tests based on matching column names, you can use a column alias to ensure that the column names in both tables match.

If applicable, you must override the column alias before you create an expression for the join condition. Data Validation Option processes the join condition using the column aliases. If you create the join condition expression based on the original column names and then override the column aliases, an error occurs.

Adding a Join View

You can add a join view to join multiple data sources. You can add a WHERE clause to filter the records from each data source and configure which columns appear in the join view.

1. Click **File** > **New** > **Join View**.

   The **Join View Editor** dialog box appears.
2. Enter a description for the join view.
3. Click **Add** in the **Table Definitions** pane.

   The **Select Data Source** dialog box appears.
4. Select the data source and click **Select**.
5. To edit the table definition, select the table definition and click **Edit**.

   You can edit the table definition to specify or change the table alias, connection properties, WHERE clause, and the optimization level. You must edit the table definition to resolve any errors that appear at the bottom of the **Join View Editor** dialog box.
6. To change the order of the data sources, select a data source and click **Up** or **Down**.
7. To remove a data source, select the data source and click **Remove**.

   When you make any change to the data sources in the join view, you must create the join conditions again.

8. Click **Select Output Fields**.

   The **Output Fields** dialog box appears.
9. Click in the **Alias** field to enter a column alias.
10. In the **Include as Output Field** column, select the checkbox for fields that you want to include as output fields of the join view.
11. To change the order of the fields, select a field and click **Move Up** or **Move Down**.
12. Configure join conditions for all the data sources.

    You can join a data source with any of the preceding data sources in the **Table Definitions** pane. You need not specify the join condition for the first data source. If applicable, you must create the join condition after you create the column aliases.
13. Click **OK**.

Configuring a Table Definition

Configure a table definition after you add a data source to the join view in the **Join View Editor** dialog box.

1. Select the data source in the **Join View Editor** dialog box.
2. Click **Edit** in the **Table Definitions** pane.

   The **Edit Table** dialog box appears.
3. Enter an alias name for the data source.

   By default, the data source name appears as the alias name.
4. Select the join type for the data source.

   Select join types for all the data sources except the first data source in the join view.
5. Configure the connection details for the table. The connection details vary depending on the data source type.
6. Optionally, enter the WHERE clause.

   Data Validation Option runs the WHERE clause when it fetches data from the table.
7. If the table is relational, you can choose to push down the WHERE clause in the database.
8. Select the database optimization level.
9. Click **OK**.

Configuring a Join Condition

Add join conditions to specify the relationship between data sources. You can create join conditions for a data source with any other data source in the previous rows.

1. Select the data source in the **Join View Editor** dialog box.
2. Click **Add** in the **Join Conditions** pane.

   The **Join Condition** dialog box appears.
3. Select the alias name of any of the data source in the previous rows.
4. Select the left field of the join condition.

   The left field of the join condition consists of the output fields from the alias name you choose. You can also configure and validate a PowerCenter expression as the field. When you enter a field name in the expression, append the alias name followed by an underscore. For example, if the alias name of the table

is customer1 and you want to use the CustIDfield in an expression, enter the expression as customer1_CustID > 100.

5. Select the right field of the join condition.

   The right field consists of the output fields of the data source for which you configure the join condition. You can also configure and validate a PowerCenter expression as the field.

6. Click **OK**.

Managing Join Views

You can edit and delete join views that you create in Data Validation Option

1. Click **Join Views** in the Navigator.

   Join views appear on the right pane.

2. Edit or delete the join view.

   If you modify the join view, re-create the join conditions in the join view. If you delete a join view, you must re-create the table pairs or single tables that contain the join view.

Join View Example

You need to validate the inventory sales done by the employees and partners to cross-check with the annual sales report.

Account table in an SAP system holds the information of an employee account. Partner table is a Salesforce table that contains the information of the inventory sold to a partner associated with an employee. Inventory is a flat file that contains the details of the inventory sold. Account_History is an Oracle table that contains the history of activities done by the account.

Current requirement is to validate data across the tables based on the inventory sales of an account. You also need to validate the account details with the historic account details to check for discrepancies. You can create a join view with the tables so that you can run single table tests to validate data.

Tables and Fields

The following table lists the join view tables and their columns:

Table	Columns
Account (SAP)	Account table contains the following columns: - Account ID - Account Name - Collection - Inventory
Partner (Salesforce)	Partner contains the following columns: - Partner ID - Partner Name - Inventory - Cost - Associated Account ID

Table	Columns
Inventory (Flat file)	Inventory table contains the following columns: - Inventory ID - Quantity - Associated Partner ID - Associated Account ID
Account_History (Oracle)	Account_History contains the following columns: - Historic Account ID - Account Name - Total Inventory - Total Collection

Creating the Join View

1. Enter Account_Cumulative as the description.
2. Add Account as the first table in the join view.
3. Add Partner, Inventory, and Account_History tables in that order.
4. Configure the table definitions with the required join types.
5. Create join conditions for Partner, Inventory, and Account_History.

Table Definition Configuration

The following list describes the tables and their join types when you configure the table definitions:

Partner

You want to capture the details of partners associated with each account. Configure an inner join for the Partner table so that Data Validation Option adds the details of the partners for which there are corresponding accounts to the join view.

Inventory

You want to capture the details of the inventory sold by the partners. Configure an inner join for the Inventory table so that Data Validation Option adds the details of the inventory for which there are corresponding partners to the join view.

Account_History

You want to capture the historic details of an account. Configure a left outer join for the Account_History table so that Data Validation Option adds all the historic account details to the join view.

Adding Join Conditions

Configure the following join conditions for the tables:

Partner

Select Account as the join table. Select Account ID output field from the Account table as the left field and Associated Account ID output field from the Partner table as the right field of the join.

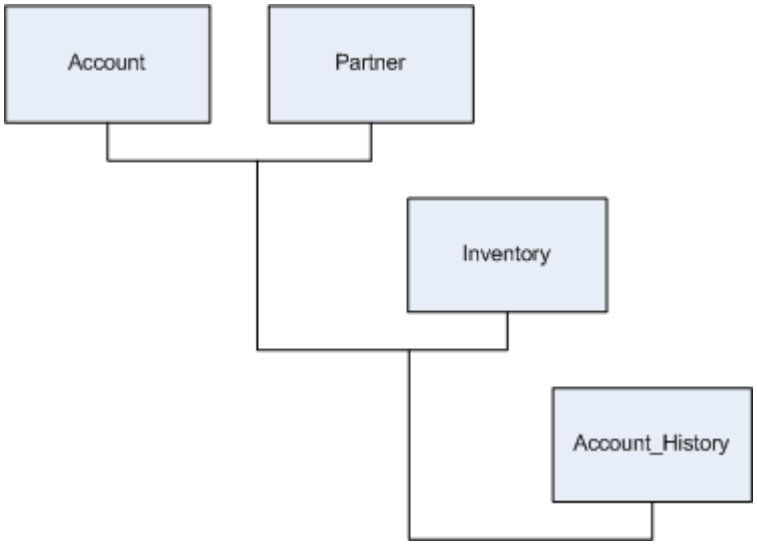
Inventory

Select Partner as the join table. Select Partner ID output field from the Partner table as the left field and Associated Partner ID output field from the Inventory table as the right field of the join.

Account_History

Select Account as the join table. Select Account ID output field from the Account table as the left field and Historic Account ID output field from the Account_History table as the right field of the join.

The following figure shows the formation of the join view with the table relationship:

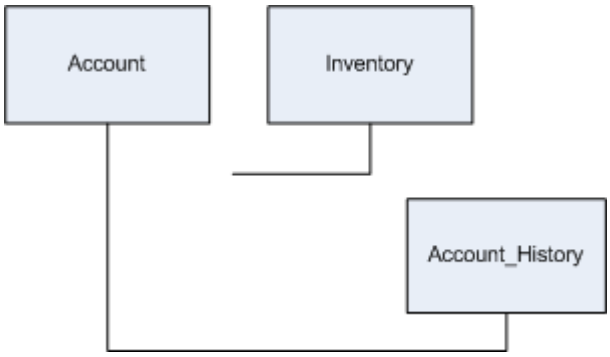


After you create the join view, create a single table with the join view. Generate and run tests on the single table to validate the data in the join view.

Removing Table from the Join View

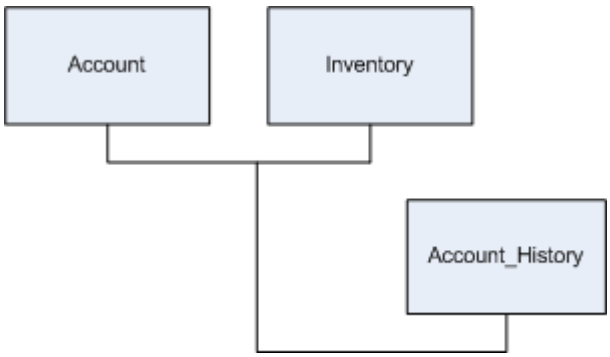
After you run the required tests for the join view with all the tables, you may want to remove the partner information and run tests solely for the account. If you remove the table Partner from the join view, the join condition for the table Inventory is no longer valid. You need to create another join condition for the table Inventory to save the join view.

The following figure shows the broken join view:



Add a join condition to the table Inventory with Account as the join table. Join Account ID field in the Account table with Associated Account ID field in the Inventory table.

The following figure shows the join view without the Partner table:


Account
Inventory
Account_History

CHAPTER 15

Aggregate Views

This chapter includes the following topics:

Aggregate Views Overview

Aggregate views allow you to view and validate data aggregations, such as averages, sums, or counts on groups of data. The aggregations are calculated and grouped in the target tables, but exist as single transactions or rows in sources. To validate the data aggregations in the sources against the data in targets, you can use aggregate views.

You can use Data Validation Option sources, lookup views, SQL views, and join views as sources in an aggregate view.

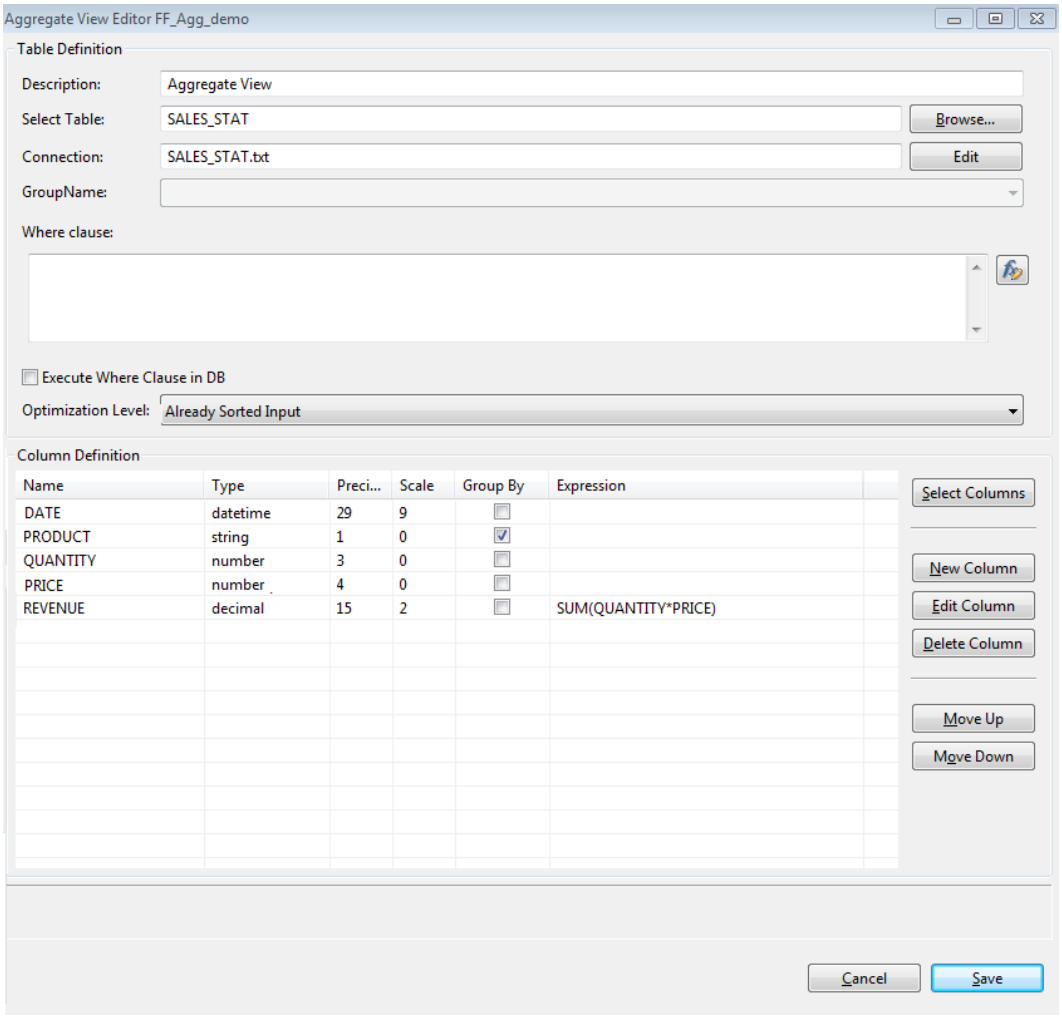
The following are some examples where you might want to use aggregate views:

- you want to pre-calculate aggregated sales data for each sales region.
- you want to calculate aggregated salaries for each department in an organization.

Aggregate View Editor

To create aggregate views, use the aggregate view editor. You specify the aggregate view properties when you create aggregate views.

The following image shows the **Aggregate View Editor**:



The **Aggregate View Editor** has the following views:

- **Table Definition**. You specify the aggregate view description, table, connection, group name, and optimization level in this view.
- **Column Definition**. You select columns that you want in the aggregate view. You can also create a column and specify columns you want to group for aggregations.

Aggregate View Properties

When you create an aggregate view, you specify the aggregate view properties.

The following table describes the aggregate view properties:

Property	Description
Description	Description of aggregate view.
Select Table	Name of the table included in the aggregate view. Provide Data Validation Option with information to create an aggregate view. Specify the tables that you want to include in the aggregate view.
Connection	PowerCenter connection to the source tables, target tables, or views.
Group Name	Name of the group if you use an XML, a PowerExchange for IMS, COBOL, or VSAM data source.
WHERE Clause	The WHERE clause filters the records that are read from the data source. Enter a valid PowerCenter expression.
Execute Where Clause in DB	Indicator that specifies that the data source must use the WHERE clause. If the data source is relational, Salesforce, SAP, or SAS, select **Execute Where Clause in DB** if you want the data source to process the WHERE clause. If you do not select this option, the aggregate view filters the data before performing aggregations.
Optimization Level	The optimization level controls the test logic processed the data source. Select one of the following options: - Default. Sorts the source data before performing aggregations. - Already Sorted Input. Indicates that the records in the data source are already sorted. If the records in the data source are not sorted, tests might fail.
Columns	The columns that you want to include in the aggregate view. Data Validation Option imports all tables that you select. You can create, edit, delete, and rearrange columns.

Group By Columns

You can define groups for aggregations in the aggregate views, instead of performing aggregations across all input data. For example, you can find the total sales grouped by region, instead of finding the total company sales.

To define a group for the aggregate view, select the appropriate column in the aggregate view editor. You can select multiple group by columns to create a new group for each unique combination.

When you group values, Data Validation Option produces one row for each group. If you do not group values, the Data Validation Option returns one row for all input rows. Data Validation Option returns the last row of each group or the last row received with the result of the aggregation.

When selecting multiple group by columns in the aggregate view, Data Validation Option uses column order to determine the order by which it groups. Order group by columns to ensure the appropriate grouping because group order can affect the results. For example, the results of grouping by DATE then PRODUCT_NAME can vary from grouping by PRODUCT_NAME then DATE.

Using Sorted Input

You can improve aggregate view performance by using the sorted input option in the aggregate view editor. When you use sorted input, Data Validation Option assumes that data is sorted by group. Data Validation Option performs aggregate calculations as it reads rows for a group. To use the **Already Sorted Input** option, you must pass sorted data to the aggregate view. If the records in the data source are not sorted, tests might fail.

Rules and Guidelines for Aggregate Views

Certain rules and guidelines apply when you create an aggregate view.

Consider the following rules and guidelines when you create an aggregate view:

- Use sorted input to decrease the use of aggregate caches. Sorted input reduces the amount of data cached during the session and improves session performance.
- Limit the number of input/output columns to reduce the amount of data the aggregate view stores in the data cache.
- Filter the data before aggregating it.
- If you create a column in the aggregate view, verify that the datatype, precision, and scale match the PowerCenter datatype, precision, and scale shown in the **Column Definitions** area. By default, the scale is 0.
- You can either include single level functions, for example, `SUM(QUANTITY)`, or nested functions, for example, `MAX(COUNT(ITEM))`, in the output columns of an aggregate view. However, you cannot include both single level and nested functions in an aggregate view.

Adding an Aggregate View

You can add an aggregate view to create a view on multiple tables in the same database.

1. Click **File** > **New** > **Aggregate View**.

   The **Aggregate View Editor** dialog box appears.
2. Enter a description for the aggregate view.
3. Click **Browse** in the **Select Table** field.

   The **Choose Data Source** dialog box appears. This dialog box displays the sources available in the repositories. You can sort information in this dialog box by clicking the column headers. You can reduce the number of items to select by typing one or more letters of the table, file, or view name in the Search field.
4. Select a table and click **Select**.

   To filter the available tables, enter a search string in the **Search** field.
5. Select a connection to the source that contains the tables included in the aggregate view.

   If you select file source, select the source directory, file name, and file type in the **Select Connection for Table** area.

6. Select a group name if you use an XML, a PowerExchange for IMS, COBOL, or VSAM data source.

   Only child elements of the XML group are available for use in the column definition of the aggregate view clause, table join field, and tests.

7. Specify an expression in the WHERE clause to filter the records. Filter the records to increase test performance.

   If the data source is relational, Salesforce, SAP, or SAS, enable **Execute Where Clause in DB** if you want the data source to process the WHERE clause.

8. Specify the optimization level.

   By default, Data Validation Option uses **Default** optimization, which sorts the source data.

9. To add columns from the tables to the aggregate view, click **Select Columns** in the **Column Definition** area.

   The **Output Fields** dialog box appears.

10. Add the columns in the **Available Fields** pane to the **Selected Fields** pane.
11. Click **OK**.

    The columns appear in the **Column Definition** area of the **Aggregate View Editor** dialog box.

12. To create a column, select a column **Column Definition** area, and click **New Column**.

    The **Aggregate View Column Definition Editor** dialog box appears.

13. Enter the name, datatype, precision, scale, and expression for the column and click **Save**.

    The column appears in the **Column Definition** area of the **Aggregate View Editor** dialog box.

14. Select **Group By** to specify columns for groupings.
15. To change the order of the columns, select a column and click **Move Up** or **Move Down**.
16. Click **Save**.

Editing Aggregate Views

1. Right-click the aggregate view in the Navigator or on the **Aggregate Views** tab.
2. Select **Edit Aggregate View**.

   The **Aggregate View Editor** dialog box opens.

3. Make the changes you require and click **Save**.

Deleting Aggregate Views

1. Right-click the aggregate view on the **Aggregate Views** tab.
2. Select **Delete Aggregate View**.

   When you delete an aggregate view, Data Validation Option deletes all table pairs, single tables, and tests that use the aggregate view.

Aggregate View Example

You want to aggregate the sales of a product by a particular date and validate the sales against the data in the target table. Use an aggregate view to aggregate data stored in the source and validate the sales against data stored in the target.

The following table displays sample data from a flat file source:

DATE	PRODUCT	QUANTITY	PRICE
01-29-2014	A	100	200
01-29-2014	B	800	10
01-29-2014	C	300	30
01-29-2014	D	200	20
01-30-2014	A	200	200
01-30-2014	B	200	700
01-30-2014	E	50	30
01-30-2014	D	100	800
01-31-2014	B	900	100
01-31-2014	C	800	900
01-31-2014	D	300	1100
02-01-2014	E	700	300
02-02-2014	A	700	300

The following table displays sample data from a target table:

PRODUCT	REVENUE
A	270000.00
B	320000.00
C	730000.00
D	530000.00
E	225000.00

To aggregate data stored in the source and validate the sales against data stored in the target, perform the following steps:

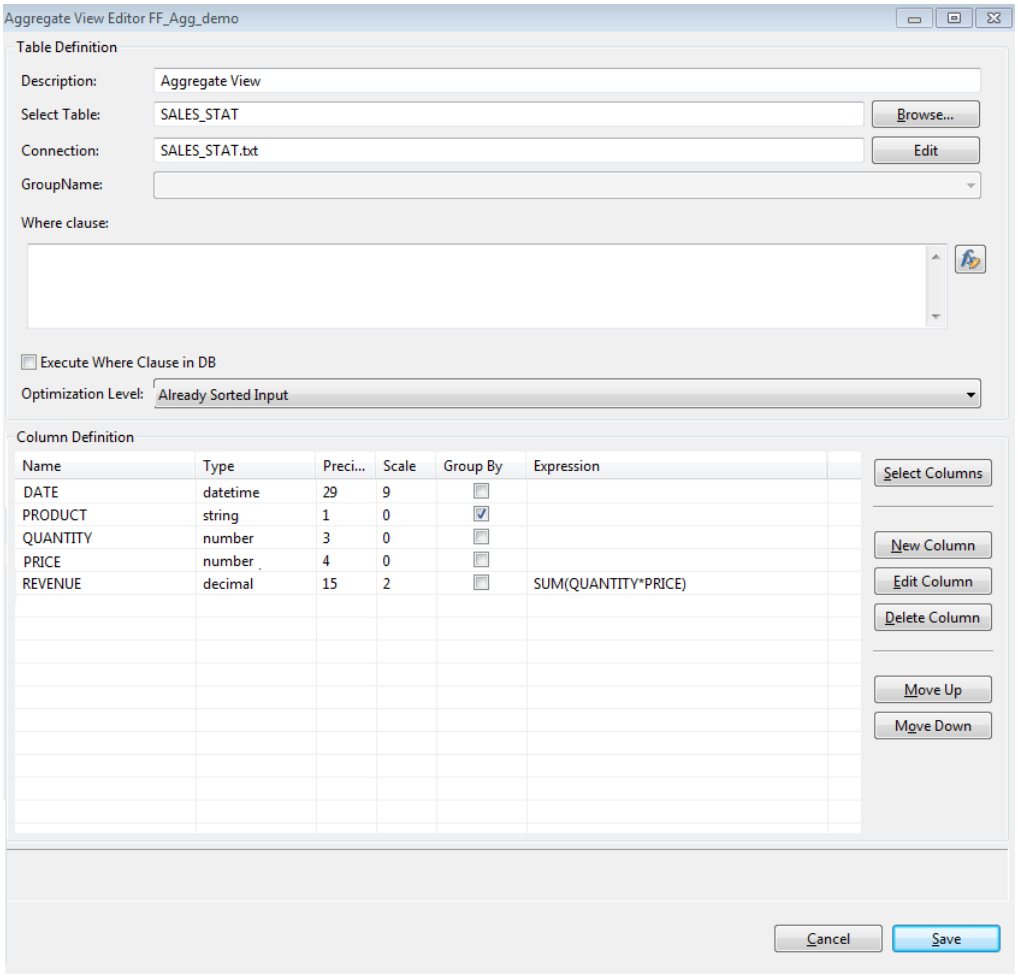
1. Create the aggregate view.
   In the aggregate view editor, specify the aggregate view description, source table, and connection. Select the fields that you require. Group by the product and create an expression to calculate the revenue. Include the following fields in the aggregate view:

   ```
   DATE

   PRODUCT

   QUANTITY

   PRICE

   REVENUE
   ```

   Use the following expression to calculate revenue:

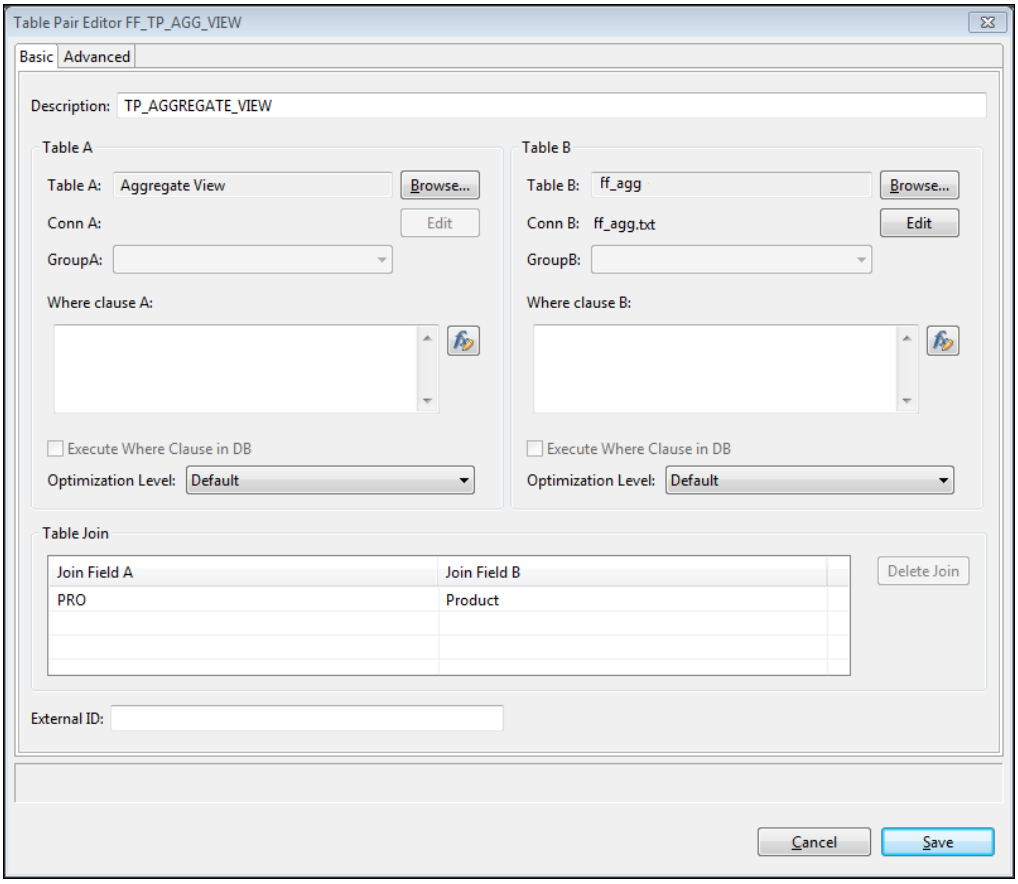
   ```
   SUM(QUANTITY*PRICE)
   ```

   The following image shows the aggregate view editor:



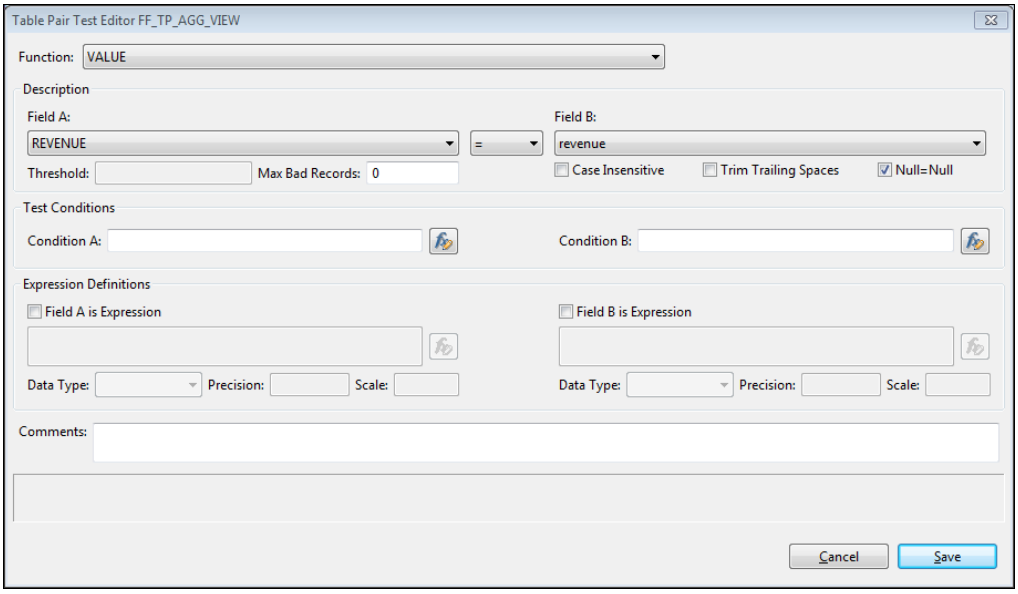
2. Create the table pair.
   Create a table pair using the aggregate view and the target table. Create a join relationship between the product in the aggregate view and the product in the target table.

The following image shows the table pair editor:



3. Create a VALUE test.
   To compare revenue in the aggregate view and the target table, create a VALUE test. The test compares the values of the fields in the aggregate view and the target table, and determines whether they are the same.

   The following image shows the table pair test editor:



   The test fails if the calculated revenue does not match the revenue in the target table.

CHAPTER 16

Business Intelligence and Reporting Tools Reports

This chapter includes the following topics:

Business Intelligence and Reporting Tools Reports Overview

You can use the Business Intelligence and Reporting Tools (BIRT) reporting engine available in Data Validation Option to generate reports.

You can generate a report for one or more table pairs or single tables.

You can generate the following BIRT reports in Data Validation Option:

Summary of Testing Activities

The Summary of Testing Activities report displays the number of passed tests, number of failed tests, number of tests with errors, and total number of tests for all table pairs and single tables. It also shows the number of table pairs and single tables with passed tests, failed tests, and errors. If a table contains some failed tests and some passed tests, the table count is included in the Fail row only.

Table Pair Summary

The Table Pair Summary report lists each table pair or single table with the associated tests. Data Validation Option displays each table pair or single table on a separate page. The report includes a brief description of each test and result.

Detailed Test Results

The Detailed Test Results report displays each test on a separate page with a detailed description of the test definition and results. The report displays the following details:

- Properties of the single-table constraint or table pair.
- View definition, if one of the test sources is an SQL or lookup view.
- Bad records. For each bad record, the report shows the corresponding key and the bad value. You can choose to view details of bad records.
- Corresponding PowerCenter mapping, session, and workflow.

Note: Report generation can take several minutes, especially when the report you generate contains hundreds of tests or test runs.

BIRT Report Generation

You can generate a report for one or more table pairs or single tables. You can also generate a report for all table pairs and single tables in a folder or a test.

Right-click the objects for which you want to generate a report, and select **Generate Report**. The **Report Parameters** dialog box appears.
The following table describes the report options:

Option	Description
User	User that created the tests. By default, Data Validation Option generates a report for the tests that the current user creates and runs. Select **All** to display tests created and run by all users. Select a user name to display tests created and run by that user.
Table Pair	Table pairs or single tables for which you want to run a report. You can generate a report on all table pairs and single tables, on the table pairs and single tables in a folder, or on a test. Data Validation Option gives you different options depending on what you select in the Navigator or details area. For example, you can generate a report on all table pairs and single tables or on the table pairs and single tables in the folder.
Recency	Test runs for which you want to run a report. You can select the latest test runs or all test runs.
Result Type	Test results for which you want to run a report. You can select all results, tests that pass, or tests that do not pass.
Run Dates	The test run dates. Enter the from date, to date, or both in the format MM/DD/YYYY.
Report Subtitle	Data Validation Option displays the subtitle on each page of the report.
Show Bad Records Detail	Show details of bad records. The report displays a delimiter before and after string values to help you identify whitespace. The report displays the difference in value along with the percentage of difference for numeric data types. This option is available only for the Detailed Test Results report.

Note: If you change the Data Validation Option repository after you configure BIRT reports, you must restart the Data Validation Client to generate accurate reports.

SQL and Lookup View Definitions

If a test contains either an SQL view or a lookup view as a source, Data Validation Option prints the view definition as part of the report.

You cannot print a report showing the definition of the view by itself because each view is tied to a specific test definition. For example, you create an SQL view, use it in a table pair, and run a test. You then update the view by changing the SQL statement, and re-run the test. Each test result is based on a different view definition, which is why the view definition must be tied to a specific result.

Custom Reports

You can write custom reports against database views in the Data Validation Option schema.

All Data Validation Option reports run against database views that are set up as part of the installation process. You can write custom reports based on the database views. Do not write reports against the underlying database tables because the Data Validation Option repository metadata can change between versions.

Related Topics:

- "Reporting Views Overview" on page 204

Viewing Reports

Data Validation Option displays reports in a browser window.

Use the arrows in the upper right corner to scroll through the pages of the report. To display a specific page, enter the page number in the **Go To Page** field.

You can display or hide the table of contents. To do this, click the table of contents icon in the upper left corner of the report. When you display the table of contents, you can click a heading to display that section of the report.

You can also print a report or export it to a PDF file. Click the print icon in the upper left corner of the report.

BIRT Report Examples

You can view the following reports:

- Summary of testing activities
- Table pair summary
- Detailed test results
- Bad records

Summary of Testing Activities

The following figure shows an example of a Summary of Testing Activities report:

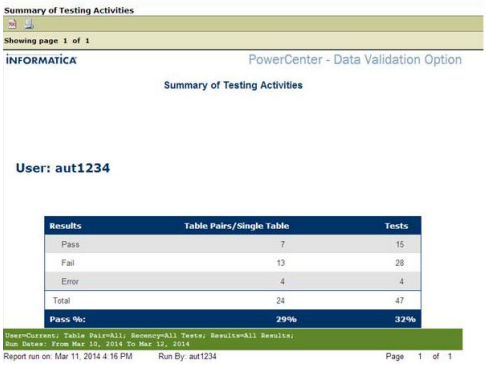


Table Pair Summary

The following figure shows an example of a Table Pair Summary report:

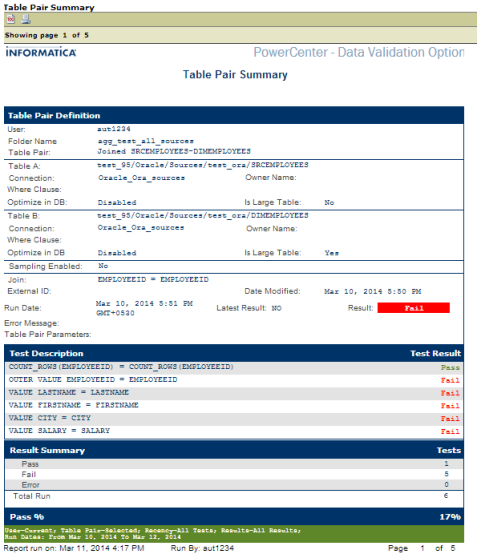


Table Pair Summary

Showing page 1 of 5

INFORMATICA PowerCenter - Data Validation Option

Table Pair Summary

Table Pair Definition

User:	aut1234		
Folder Name	agg_test_all_sources		
Table Pair:	Joined SRCEMPLOYEES-DIMEMPLOYEES		
Table A:	test_95/Oracle/Sources/test_ora/SRCEMPLOYEES		
Connection:	Oracle_Ora_sources	Owner Name:	
Where Clause:			
Optimize in DB:	Disabled	Is Large Table:	No
Table B:	test_95/Oracle/Sources/test_ora/DIMEMPLOYEES		
Connection:	Oracle_Ora_sources	Owner Name:	
Where Clause:			
Optimize in DB	Disabled	Is Large Table:	Yes
Sampling Enabled:	No		
Join:	EMPLOYEEID = EMPLOYEEID		
External ID:		Date Modified:	Mar 10, 2014 5:50 PM
Run Date:	Mar 10, 2014 5:51 PM GMT+0530	Latest Result: NO	Result: Fail
Error Message:			
Table Pair Parameters:			

Test Description	Test Result
COUNT_ROWS(EMPLOYEEID) = COUNT_ROWS(EMPLOYEEID)	Pass
OUTER VALUE EMPLOYEEID = EMPLOYEEID	Fail
VALUE LASTNAME = LASTNAME	Fail
VALUE FIRSTNAME = FIRSTNAME	Fail
VALUE CITY = CITY	Fail
VALUE SALARY = SALARY	Fail

Result Summary	Tests
Pass	1
Fail	5
Error	0
Total Run	6
Pass %	17%

Tags-Current; Table Pair-Selected; Recency-All Tests; Results-All Results;
Run Dates: From Mar 10, 2014 To Mar 12, 2014

Report run on: Mar 11, 2014 4:17 PM Run By: aut1234 Page 1 of 5

Detailed Test Results – Test Page

The following figure shows an example of a test page in the Detailed Test Results report:

Detailed Test Results

Showing page 1 of 77

INFORMATICA PowerCenter - Data Validation Option

Detailed Test Results

Test

Folder Name:	agg_test_all_sources		
Table Pair	Joined SRCEMPLOYEES-DIMEMPLOYEES		
Test:	COUNT_ROWS(EMPLOYEEID) = COUNT_ROWS(EMPLOYEEID)		
Run Date:	Mar 10, 2014 5:51	Test Result:	PASS

Test Result Detail

Result A:	500
Result B:	500
Threshold:	

Test Definition

Test:	COUNT_ROWS(EMPLOYEEID) = COUNT_ROWS(EMPLOYEEID)				
Condition A:					
Condition B:					
Case Insensitive:	No	Trim Trailing	No	Null = Null:	No

Table Pair Definition

Table A:	test_95/Oracle/Sources/test_ora/SRCEMPLOYEES		
Connection:	Oracle_Ora_sources	Owner Name:	
Connection Type:	Oracle	Connection User:	pc_auto1
Connection Details:	Connect String = qa11gr2		
Where Clause:			
Optimize in DB:	Disabled	Is Large Table:	No
Table B:	test_95/Oracle/Sources/test_ora/DIMEMPLOYEES		
Connection:	Oracle_Ora_sources	Owner Name:	
Connection Type:	Oracle	Connection User:	pc_auto1
Connection Details:	Connect String = qa11gr2		
Where Clause:			
Optimize in DB:	Disabled	Is Large Table:	Yes
Sampling Enabled:	No		
Join:	EMPLOYEEID = EMPLOYEEID		
External ID:			
Last Modified:	Mar 10, 2014 5:50 PM GMT+0530		

Run-time Information

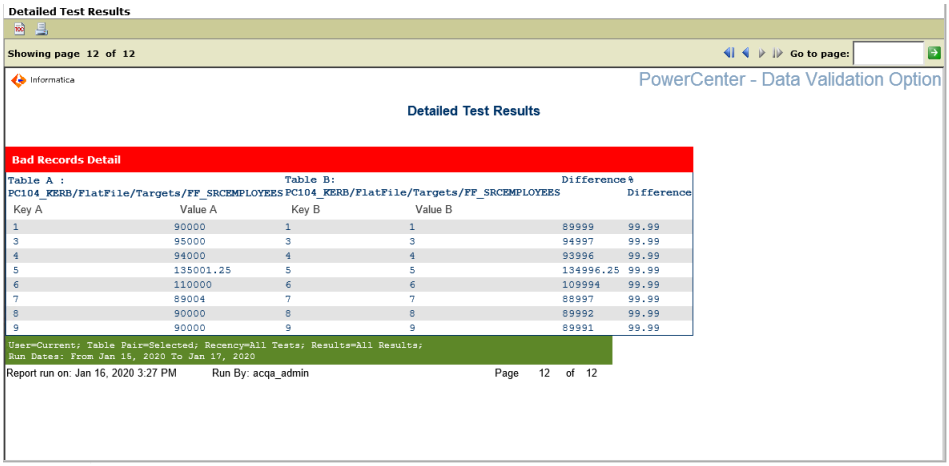
Session Name:	DV_S_Joined_SRCEMPLOYEES_DIMEMPLOYEES_2507
Mapping Name:	DV_M_Joined_SRCEMPLOYEES_DIMEMPLOYEES_2507

Tags-Current; Table Pair-Selected; Recency-All Tests; Results-All Results;
Run Dates: From Mar 10, 2014 To Mar 12, 2014

Report run on: Mar 11, 2014 4:17 PM Run By: aut1234 Page 1 of 77

Detailed Test Results – Bad Records Page

The following figure shows an example of a bad records page in the Detailed Test Results report:



Detailed Test Results

Showing page 12 of 12 Go to page:

Informatica PowerCenter - Data Validation Option

Detailed Test Results

Bad Records Detail

Table A : PC104_KERB/FlatFile/Targets/FF_SRCEMPLOYEES		Table B: PC104_KERB/FlatFile/Targets/FF_SRCEMPLOYEES		Difference	% Difference
Key A	Value A	Key B	Value B		
1	90000	1	1	89999	99.99
3	95000	3	3	94997	99.99
4	94000	4	4	93996	99.99
5	135001.25	5	5	134996.25	99.99
6	110000	6	6	109994	99.99
7	89004	7	7	88997	99.99
8	90000	8	8	89992	99.99
9	90000	9	9	89991	99.99

User=Current; Table Pair=Selected; Recency=All Tests; Results=All Results;
Run Dates: From Jan 16, 2020 To Jan 17, 2020

Report run on: Jan 16, 2020 3:27 PM Run By: acqa_admin Page 12 of 12

CHAPTER 17

Dashboards

This chapter includes the following topics:

Dashboards Overview

You can generate Data Validation Option dashboards to get an overview of the testing activities and test results.

Dashboards display multiple reports in a single page. You need not generate individual reports to get an overview of test results across a fixed time period.

To view a dashboard, click **Dashboard** > **<dashboard name>**.

You can view the following dashboards in Data Validation Option:

DVO Home Dashboard

Displays the following reports for the past 30 days:

- Tests Run Vs Tests Passed
- Total Rows vs Percentage of Bad Rows
- Percentage of Tests Passed
- Percentage of Bad Rows

You can click through the Total Rows vs Percentage of Bad Rows report and Percentage of Bad Rows reports to view the Bad Rows report. You can click through Tests Run Vs Tests Passed to view the Validation Failures report.

Repository Dashboard

Displays the Validation Failures for Folders report for the past 24 hours and 30 days in both graphical and tabular formats. The dashboard also displays the Most Recent Failed Runs report for the repository.

You can click through the Validation Failure by Folder report to view the Failed Tests report. You can click through the Most Recent Failed Runs report to view the Most Recent Failed Runs report for a folder and the Detailed Test Result report.

Folder Dashboard

Displays the Bad Rows report and Failed Tests report for the past 24 hours and 30 days in both graphical and tabular formats.

You can click through the Failed Tests report to view the Table Pair/Table report and the Bad Rows report to view the Detailed Test Result report.

Table Dashboard

The Table dashboard displays the following reports for the past 30 days:

- Tests Passed Vs Test Failed
- Bad Rows
- Table Pair/Table Run Summary

You can click through the Bad Rows report to view the contents to view the Detailed Test Result report.

Dashboard Types

You can view the following dashboards in Data Validation Option:

DVO Home Dashboard

Displays the following reports for the past 30 days:

- Tests Run Vs Tests Passed
- Total Rows vs Percentage of Bad Rows
- Percentage of Tests Passed
- Percentage of Bad Rows

You can click through the Total Rows vs Percentage of Bad Rows report and Percentage of Bad Rows reports to view the Bad Rows report. You can click through Tests Run Vs Tests Passed to view the Validation Failures report.

Repository Dashboard

Displays the Validation Failures for Folders report for the past 24 hours and 30 days in both graphical and tabular formats. The dashboard also displays the Most Recent Failed Runs report for the repository.

You can click through the Validation Failure by Folder report to view the Failed Tests report. You can click through the Most Recent Failed Runs report to view the Most Recent Failed Runs report for a folder and the Detailed Test Result report.

Folder Dashboard

Displays the Bad Rows report and Failed Tests report for the past 24 hours and 30 days in both graphical and tabular formats.

You can click through the Failed Tests report to view the Table Pair/Table report and the Bad Rows report to view the Detailed Test Result report.

Table Dashboard

The Table dashboard displays the following reports for the past 30 days:

- Tests Passed Vs Test Failed
- Bad Rows
- Table Pair/Table Run Summary

You can click through the Bad Rows report to view the contents to view the Detailed Test Result report.

DVO Home Dashboard

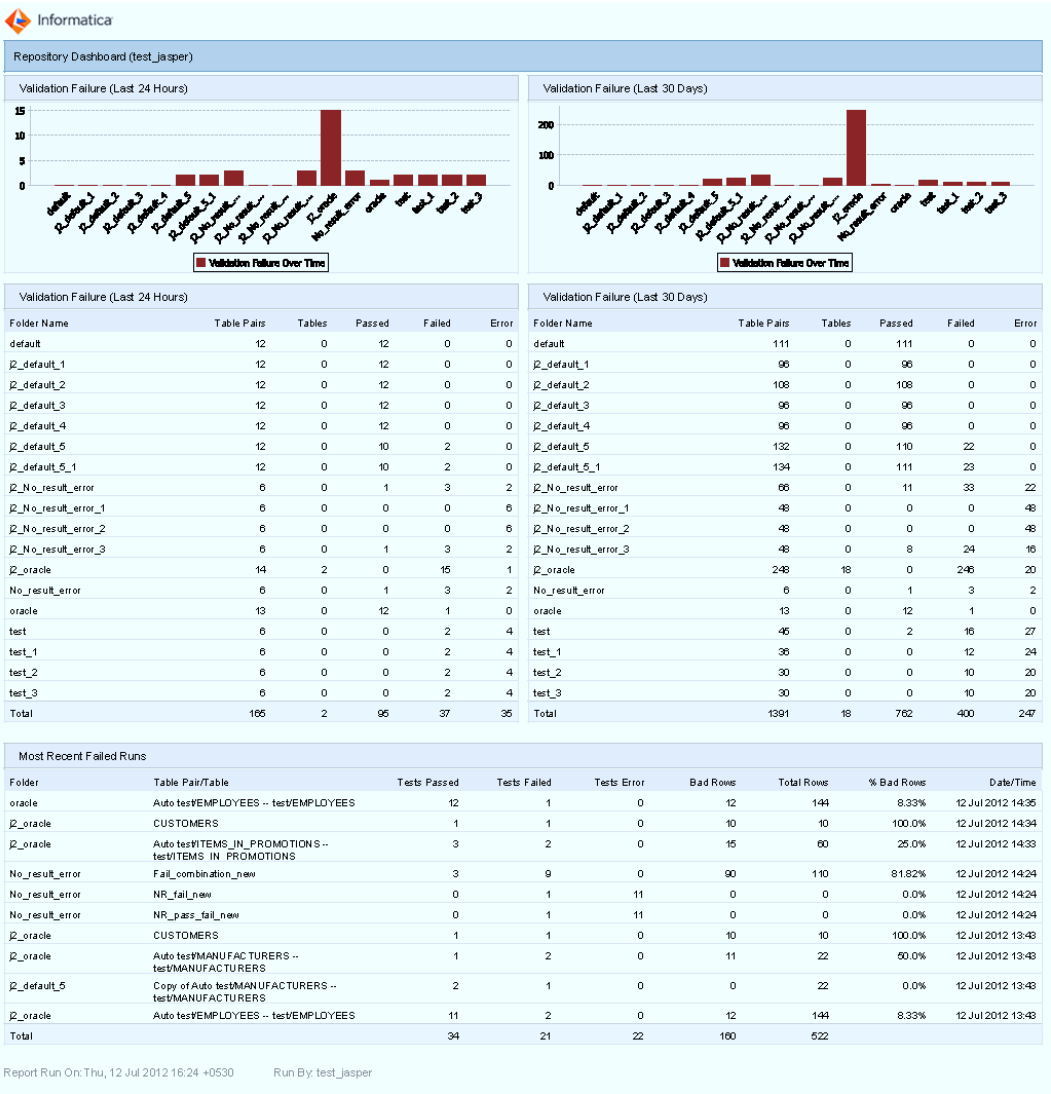
The DVO Home dashboard displays the test details in the Data Validation Schema over the past 30 days.

The following figure shows an example of the DVO Home Dashboard:

Repository Dashboard

The Repository Dashboard displays the details of tests run in the Data Validation Option repository over the past 30 days and the past 24 hours.

The following figure shows an example of the Repository Dashboard:



Informatica

Repository Dashboard (test_jasper)

Validation Failure (Last 24 Hours)

Validation Failure (Last 30 Days)

Validation Failure (Last 24 Hours)

Folder Name	Table Pairs	Tables	Passed	Failed	Error
default	12	0	12	0	0
j2_default_1	12	0	12	0	0
j2_default_2	12	0	12	0	0
j2_default_3	12	0	12	0	0
j2_default_4	12	0	12	0	0
j2_default_5	12	0	10	2	0
j2_default_5_1	12	0	10	2	0
j2_No_result_error	6	0	1	3	2
j2_No_result_error_1	6	0	0	0	6
j2_No_result_error_2	6	0	0	0	6
j2_No_result_error_3	6	0	1	3	2
j2_oracle	14	2	0	15	1
No_result_error	6	0	1	3	2
oracle	13	0	12	1	0
test	6	0	0	2	4
test_1	6	0	0	2	4
test_2	6	0	0	2	4
test_3	6	0	0	2	4
Total	165	2	95	37	35

Validation Failure (Last 30 Days)

Folder Name	Table Pairs	Tables	Passed	Failed	Error
default	111	0	111	0	0
j2_default_1	96	0	96	0	0
j2_default_2	108	0	108	0	0
j2_default_3	96	0	96	0	0
j2_default_4	96	0	96	0	0
j2_default_5	132	0	110	22	0
j2_default_5_1	134	0	111	23	0
j2_No_result_error	66	0	11	33	22
j2_No_result_error_1	48	0	0	0	48
j2_No_result_error_2	48	0	0	0	48
j2_No_result_error_3	48	0	8	24	16
j2_oracle	248	18	0	246	20
No_result_error	6	0	1	3	2
oracle	13	0	12	1	0
test	45	0	2	16	27
test_1	36	0	0	12	24
test_2	30	0	0	10	20
test_3	30	0	0	10	20
Total	1391	18	762	400	247

Most Recent Failed Runs

Folder	Table Pair/Table	Tests Passed	Tests Failed	Tests Error	Bad Rows	Total Rows	% Bad Rows	Date/Time
oracle	Auto test/EMPLOYEES -- test/EMPLOYEES	12	1	0	12	144	8.33%	12 Jul 2012 14:35
j2_oracle	CUSTOMERS	1	1	0	10	10	100.0%	12 Jul 2012 14:34
j2_oracle	Auto test/ITEMS_IN_PROMOTIONS -- test/ITEMS_IN_PROMOTIONS	3	2	0	15	60	25.0%	12 Jul 2012 14:33
No_result_error	Fail_combination_new	3	9	0	90	110	81.82%	12 Jul 2012 14:24
No_result_error	NR_fail_new	0	1	11	0	0	0.0%	12 Jul 2012 14:24
No_result_error	NR_pass_fail_new	0	1	11	0	0	0.0%	12 Jul 2012 14:24
j2_oracle	CUSTOMERS	1	1	0	10	10	100.0%	12 Jul 2012 13:43
j2_oracle	Auto test/MANUFACTURERS -- test/MANUFACTURERS	1	2	0	11	22	50.0%	12 Jul 2012 13:43
j2_default_5	Copy of Auto test/MANUFACTURERS -- test/MANUFACTURERS	2	1	0	0	22	0.0%	12 Jul 2012 13:43
j2_oracle	Auto test/EMPLOYEES -- test/EMPLOYEES	11	2	0	12	144	8.33%	12 Jul 2012 13:43
Total		34	21	22	160	522		

Report Run On: Thu, 12 Jul 2012 16:24 +0530 Run By: test_jasper

Folder Dashboard

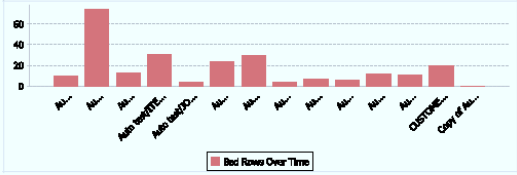
The Folder Dashboard displays the details of tests run in the repository over the past 30 days and the past 24 hours.

The following figure shows an example of the Folder Dashboard:

Informatica

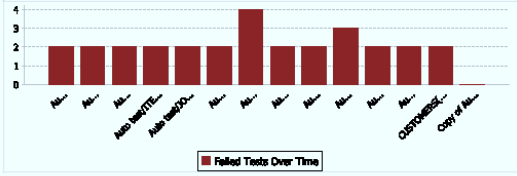
Folder Dashboard: j2_oracle (test_jasper)

Bad Rows (Last 24 Hours)



Bad Rows (Last 30 Days)

Failed Tests (Last 24 Hours)



Failed Tests (Last 30 Days)

Failed Tests (Last 24 Hours)

Table Pairs/Tables	Tests	Passed	Failed	Error	Bad Rows
Auto test\CUSTOMERS -- test/CUSTOMERS(ID:1490)	12	10	2	0	10
Auto test\ORDER_ITEMS -- test/ORDER_ITEMS(ID:1503)	5	3	2	0	74
Auto test\DISTRIBUTORS -- test/DISTRIBUTORS(ID:1522)	6	4	2	0	13
Auto test\ITEMS -- test/ITEMS(ID:1542)	9	7	2	0	31
Auto test\JOBS -- test/JOBS(ID:1546)	3	1	2	0	4
Auto test\ORDERS -- test/ORDERS(ID:1562)	9	7	2	0	24
Auto test\ITEMS_IN_PROMOTIONS -- test/ITEMS_IN_PROMOTIONS(ID:1567	10	6	4	0	30
Auto test\PROMOTIONS -- test/PROMOTIONS(ID:1570)	6	4	2	0	4
Auto test\DEPARTMENT -- test/DEPARTMENT(ID:1582)	3	1	2	0	7
Auto test\STORES -- test/STORES(ID:1585)	10	7	3	0	6
Auto test\EMPLOYEES -- test/EMPLOYEES(ID:1595)	13	11	2	0	12
Auto test\MANUFACTURERS -- test/MANUFACTURERS(ID:1604)	3	1	2	0	11
CUSTOMERS(ID:1605)	4	2	2	0	20
Copy of Auto test/STORES -- test/STORES(ID:1606)	10	0	0	10	0
Total	103	64	29	10	246

Failed Tests (Last 30 Days)

Table Pairs/Tables	Tests	Passed	Failed	Error	Bad Rows
Auto test\CUSTOMERS -- test/CUSTOMERS(ID:1490)	216	180	36	0	180
Auto test\ORDER_ITEMS -- test/ORDER_ITEMS(ID:1503)	90	54	36	0	1332
Auto test\DISTRIBUTORS -- test/DISTRIBUTORS(ID:1522)	126	84	42	0	273
Auto test\ITEMS -- test/ITEMS(ID:1542)	171	133	38	0	589
Auto test\JOBS -- test/JOBS(ID:1546)	54	18	36	0	72
Auto test\ORDERS -- test/ORDERS(ID:1562)	162	126	36	0	432
Auto test\ITEMS_IN_PROMOTIONS -- test/ITEMS_IN_PROMOTIONS(ID:1567	100	60	40	0	300
Auto test\PROMOTIONS -- test/PROMOTIONS(ID:1570)	108	72	36	0	72
Auto test\DEPARTMENT -- test/DEPARTMENT(ID:1582)	66	22	44	0	154
Auto test\STORES -- test/STORES(ID:1585)	180	126	54	0	108
Auto test\EMPLOYEES -- test/EMPLOYEES(ID:1595)	260	220	40	0	240
Auto test\MANUFACTURERS -- test/MANUFACTURERS(ID:1604)	54	18	36	0	198
CUSTOMERS(ID:1605)	36	18	18	0	180
Copy of Auto test/STORES -- test/STORES(ID:1606)	200	0	0	200	0
Total	1823	1131	492	200	4130

Report Run On: Thu, 12 Jul 2012 16:21 +0530 Run By: test_jasper

CHAPTER 18

DVOCmd Command Line Program

This chapter includes the following topics:

DVOCmd Command Line Program Overview

You can use the DVOCmd command line program to invoke Data Validation Option tasks from the command line. For example, you can create and run tests without using the Data Validation Client.

You can use DVOCmd command line program to schedule tests to run. You can also embed a specific test as part of the ETL workflow or as part of another process. For example, you can create an ETL workflow that moves data from source to staging, runs validation, and then moves data into the target or an error table based on the validation results.

The command line program writes regular messages to the STDOUT output stream. It writes error messages to the STDERR output stream. You can use the redirection operator to write messages to a file.

On Windows, the DVOCmd command line program is `DVOCmd.exe`. `DVOCmd.exe` exists in the following directory:

```
C:\Program Files (x86)\Informatica<version>\DVO\
```

Rules and Guidelines for Running DVOCmd Commands

Certain rules and guidelines apply when you run DVOCmd commands.

Use the following rules and guidelines when you run a DVOCmd command:

- Run DVOCmd from the Data Validation Option installation directory.
- Use the following syntax when you run a DVOCmd command: `DVOCmd Command [Argument] [--Options] [Arguments]`.
- Use the proper case for all option names. Option names are case sensitive.
- Optionally, enclose command options and arguments in square brackets.
- Enter the command without any argument to get help about the command. For example, enter `$HOME/DVO/DVOCmd RunTests`.
- If you invoke DVOCmd from a Command task in a PowerCenter workflow, ensure that the environment variables are set for the user account that runs PowerCenter Services. Also, ensure that DVOCmd is installed under the same user account.

CopyFolder

Copies the contents of a folder in a user workspace to a different folder in the same workspace or to another user workspace. The target folder must be a new folder. The target folder cannot exist in the target workspace. You can run this command on Windows.

The CopyFolder command copies the table pairs, single tables, and test cases that exist within the source folder. It does not copy test runs or the external IDs associated with table pairs or single tables.

If the table pairs or single tables in the source folder use an SQL or lookup view, the CopyFolder command copies the SQL or lookup view to the target user workspace unless the workspace contains a view with the same name.

Before Data Validation Option copies a folder, it verifies that all data sources associated with the objects being copied exist in the target workspace. If any data source is missing, Data Validation Option does not copy the folder.

The CopyFolder command uses the following syntax:

```
DVOCmd CopyFolder [--confdir conf_dir] [--username Username] [--password Password] [--
ePassword Password Environment Variable] [--namespace Namespace] --fromUser source_user --
fromFolder source_folder --toUser target_user [--toFolder target_folder] [--reuseViews Yes]
```

The following table describes CopyFolder options and arguments:

Option	Argument	Description
--confdir	CONFDIR	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--username	USERNAME	Informatica domain user name for the domain to which you configured Informatica authentication. Required if you configure Informatica Authentication.
--password	PASSWORD	Password for the Informatica user name. If you set the **password** option, then do not set the **ePassword** option. Required if you configure Informatica Authentication.
--ePassword	PASSWORD ENVIRONMENT VARIABLE	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	NAMESPACE	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.
--fromUser	FROMUSER	Name of the source user. Data Validation Option copies the folder in this user workspace.
--fromFolder	FROMFOLDER	Name of the source folder. Data Validation Option copies this folder.
--toUser	TOUSER	Name of the target user. Data Validation Option copies the folder to this user workspace. The source user and the target user can be the same user.
--toFolder	TOFOLDER	Name of the target folder. The target folder must be unique in the target workspace. If you do not specify a target folder, Data Validation Option creates a folder in the target workspace with the same name as the source folder.
--reuseViews	REUSEVIEWS	Reuses an SQL or lookup view in the target workspace when the workspace contains a view with the same name as a source SQL or lookup view. If you specify this option, Data Validation Option does not copy the source view to the target workspace. If you do not specify this option, Data Validation Option prompts you for the action to take when views with the same name are found.

CreateUserConfig

Creates a preferences file with the specified user names and user configuration directory. You can run this command on Windows.

The CreateUserConfig command uses the following syntax:

```
DVOCmd CreateUserConfig user_name [user_name2 …] [--confdir conf_dir] --outputdir output_dir
[--overwrite]
```

Creates a preferences file called <username>-preferences.xml for each user in the output directory. The preferences file contains connection information for the Data Validation Option repository. Copy each preferences file from the output directory to the user configuration directory and rename it to preferences.xml. This allows each user to access the Data Validation Option repository.

The following table describes CreateUserConfig options and arguments:

Option	Argument	Description
-	user_name	The name of the Data Validation Option user. To create multiple users, enter multiple user names separated with spaces.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Enclose the file path in quotes.
--outputdir	output_dir	Directory in which to store user preferences files. Enclose the file path in quotes. Use double-quotes if the path has space or special characters.
--overwrite	-	Overwrites the configuration files.

DisableInformaticaAuthentication

Disables Informatica authentication in a Data Validation Option schema. You can run this command on Windows.

The DisableInformaticaAuthentication command uses the following syntax:

```
DVOCmd DisableInformaticaAuthentication [--confdir conf_dir] [--username User name] [--
password Password] [--ePassword Password Environment Variable] [--namespace Namespace]
```

The following table describes DisableInformaticaAuthentication options and arguments:

Option	Argument	Description
--confdir	CONFDIR	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Enclose the file path in quotes.
--username	USERNAME	Informatica domain administrator user name for the domain to which you configured Informatica authentication.
--password	PASSWORD	Password of the Informatica administrator account. If you set the **password** option, then do not set the **ePassword** option.

Option	Argument	Description
--ePassword	PASSWORD ENVIRONMENT VARIABLE	Encrypted form of the password of the Informatica administrator account. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	NAMESPACE	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.

EncryptPassword

Generates and displays an encrypted form of a password. If you configure Data Validation Option to use Informatica authentication, use the EncryptPassword command to encrypt the password for an Informatica user or administrator. Set an environment variable to the encrypted password. Use the environment variable in DVOCmd commands, such as the CopyFolder, RefreshRepository, or RunTests commands.

Use a name of your choice for the variable. In the DVOCmd command, use the ePassword option instead of the password option and specify the environment variable. For example, set an environment variable named INFAPWD to the encrypted password. Use INFAPWD as the value of the ePassword option in DVOCmd commands that require the password. The DVOCmd commands replace the environment variable with the encrypted password, and then decrypt the password.

Note: If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**

The EncryptPassword command uses the following syntax:

```
DVOCmd EncryptPassword <password>
```

The following table describes EncryptPassword options and arguments:

Option	Argument	Description
-	password	Password of the Informatica user.

Along with the encrypted password, the command displays the string that results when DVOCmd commands decrypt the password. The decrypted and unencrypted forms of the password must match.

You can also use the command line program pmpasswd to encrypt passwords. The pmpasswd utility installs in one of the following directories:

- `<InformaticaInstallationDir>/server/bin`
- `<InformaticaClientInstallationDir>/PowerCenterClient/client/bin`

The pmpasswd utility uses the following syntax:

```
pmpasswd <password> [-e CRYPT_SYSTEM)]
```

The following table describes pmpasswd options and arguments:

Option	Argument	Description
-	password	Required. The password to encrypt.
-e	CRYPT_SYSTEM	Optional. The encryption type. The value is CRYPT_SYSTEM.

ExportMetadata

Exports all Data Validation Option metadata to a DVML file. You can export all objects in a folder, or you can export individual objects by their external ID. You can run this command on Windows.

The ExportMetadata command uses the following syntax:

```
DVOCmd ExportMetadata file_name [--confdir conf_dir] [--username User name] [--password
Password] [--ePassword Password Environment Variable] [--namespace Namespace][--folderNames
folderA,folderB] [--externalIDs TP1,TP2,TP3] [--skipInvalidObjects]
```

The following table describes ExportMetadata options and arguments:

Option	Argument	Description
-	file_name	The file to which you want to export metadata.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--username	User name	Informatica domain user name for the domain to which you configured Informatica authentication. Required if you configure Informatica Authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica Authentication. If you set the password option, then do not set the ePassword option.
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the ePassword option, and then do not set the password option. If the command contains both the password and ePassword options, then the Data Validation Option uses the ePassword and ignores the password.
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.

Option	Argument	Description
--folderNames	FolderA	Exports one or more folders. Use commas to separate multiple folder names. **Note:** If the folder name or external ID contains commas, spaces, or special characters, you will need to use different syntax.
--externalIDs	tp1	Exports individual objects. Use commas to separate multiple external IDs. **Note:** If the folder name or external ID contains commas, spaces, or special characters, you will need to use different syntax.
--skipInvalidObjects	-	Skips all invalid objects when you export a folder.

Related Topics:

- "Rules and Guidelines for Special Characters" on page 191
- "RefreshRepository" on page 186
- "RunTests" on page 187
- "InstallTests" on page 182

ImportMetadata

Imports metadata into Data Validation Option from an export DVML file. You can run this command on Windows.

The ImportMetadata command uses the following syntax:

```
DVOCmd ImportMetadata file_name [--confdir conf_dir] [--username Username] [--password
Password] [--ePassword Password Environment Variable] [--namespace Namespace] [--overwrite]
```

The following table describes ImportMetadata options and arguments:

Option	Argument	Description
-	file_name	The file that contains metadata to be imported.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--username	User name	Informatica domain user name for the domain to which you configured Informatica authentication. Required if you configure Informatica Authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica Authentication. If you set the **password** option, then do not set the **ePassword** option.

Option	Argument	Description
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.
--overwrite	-	Overwrites existing objects.

InstallTests

Prepares all tests for single table objects or table pair objects. For each test, this command generates the PowerCenter mapping in the Data Validation Option target folder in the PowerCenter repository. You can run this command on Windows or UNIX.

The InstallTests command uses the following syntax:

```
DVOCmd InstallTests external_ID [external_ID …] [--confdir conf_dir] [--username User name] [--password Password] [--ePassword Password Environment Variable] [--namespace Namespace] [--cacheSize CACHESIZE] [--forceInstall] [--folderNames]
```

The following table describes InstallTests options and arguments:

Option	Argument	Description
-	external_ID	Required if you do not use folderNames. The external ID of a single-table object or a table-pair object. You can separate multiple external IDs with spaces. **Note:** If the folder name or external ID contains commas, spaces, or special characters, you will need to use different syntax.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Enclose the file path in quotes.
--username	User name	Informatica domain administrator user name for the domain to which you configured Informatica authentication. Required if you configure Informatica authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica authentication. If you set the password option, then do not set the ePassword option.

Option	Argument	Description
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the ePassword option, then do not set the password option. If the command contains both the password and ePassword options, then the Data Validation Option uses the ePassword and ignores the password.
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.
--cacheSize	CACHESIZE	Memory allocation to generate transformations in PowerCenter mappings. Increase the cache size for tests that contain multiple joins and lookups. Default is 20 MB for the data cache and 10 MB for the index cache. Specify "Auto" to enable PowerCenter to compute the cache size.
--forceInstall	-	Creates mappings for table pair objects in the repository. DVOCmd uses the DTM buffer size value configured in preferences.xml. If you modify the DTM buffer size value in the preferences file, run the InstallTests command with the forceInstall option for existing table-pair objects before you run the RunTests command.
--folderNames	folderA	Required if you do not use external_ID. Prepares all objects in the specified folders. You can separate multiple folder names with commas. **Note:** If your folder name or external ID contains commas, spaces, or special characters, you must use different syntax.

Related Topics:

- "Rules and Guidelines for Special Characters" on page 191
- "ExportMetadata" on page 180
- "RefreshRepository" on page 186
- "RunTests" on page 187

Cache Settings

You can configure the cache settings for the PowerCenter transformations that Data Validation Option generates for the tests.

Data Validation Option generates a PowerCenter mapping with Joiner transformations and Lookup transformations for a test that contains joins and lookups. Joiner transformations and Lookup transformations require a large amount of memory. Configure the cache settings for a complex test that contains joins and lookups.

The value that you specify as cache settings for a test persists until you modify the test. For information about optimal cache setting for the tests, see the *PowerCenter Advanced Workflow Guide*.

LinkDVOUsersToInformatica

Links the existing Data Validation Option users with Informatica domain users. You can run this command on Windows.

Create a text file that contains a list of the Data Validation Option users and Informatica domain users in the following format:

```
<dvo_user_name1>,<Informatica_user_name1>
<dvo_user_name2>,<Informatica_user_name2>
.
.
<dvo_user_nameN>,<Informatica_user_nameN>
```

The LinkDVOUsersToInformatica command uses the following syntax:

```
DVOCmd LinkDVOUsersToInformatica [file_name] [--confdir conf_dir] [--username User name] [--password Password] [--ePassword Password Environment Variable] [--namespace Namespace]
```

The following table describes LinkDVOUsersToInformatica argument:

Option	Argument	Description
-	file_name	Name of the file that contains the mapping between Data Validation Option users and Informatica users. Enclose the file path in quotes.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--username	User name	Informatica domain administrator user name for the domain to which you configured Informatica authentication. Required if you configure Informatica authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica authentication. If you set the **password** option, then do not set the **ePassword** option.
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.

PurgeRuns

Purges test runs from the Data Validation Option repository. You can purge deleted test runs or purge test runs by date. When you purge test runs by date, you can purge all test runs that occur on or after a specified date, before a specified date, or between two dates. You can run this command on Windows.

The PurgeRuns command uses the following syntax:

```
DVOCmd PurgeRuns [--confdir conf_dir] [--username User name] [--password Password] [--
ePassword Password Environment Variable] [--namespace Namespace] [--deleted] [--fromDate
from_date] [--toDate to_date]
```

If you configure Informatica authentication for the Data Validation Option schema, enter --username and --password.

The following table describes PurgeRuns options and arguments:

Option	Argument	Description
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--username	User name	Informatica domain user name for the domain to which you configured Informatica authentication. Required if you configure Informatica authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica authentication. If you set the **password** option, then do not set the **ePassword** option.
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.
--deleted	-	Purges deleted test runs.
--fromDate	from_date	Purges test runs that occur on or after this date.
--toDate	to_date	Purges test runs that occur before this date.

RefreshRepository

Refreshes a source or target repository. You can run this command on Windows or UNIX.

The RefreshRepositorycommand uses the following syntax:

```
DVOCmd RefreshRepository repo_name [--confdir conf_dir] [--username User name] [--password Password] [--ePassword Password Environment Variable] [--namespace Namespace] [--all] [--connections] [--folderList] [--allFolders] [--folder folder_name] [--dryrun]
```

The following table describes RefreshRepository options and arguments:

Option	Argument	Description
-	repo_name	Name of the repository you want to refresh. **Note:** This can take several minutes to several hours depending on the size of the repository.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Use double-quotes if the path has space or special characters.
--username	User name	Informatica domain user name for the domain to which you configured Informatica authentication. Required if you configure Informatica authentication.
--password	Password	Password for the Informatica user name. Required if you configure Informatica authentication. If you set the **password** option, then do not set the **ePassword** option.
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.
--all	-	Refreshes all source, target, folder, and connection metadata for the repository. **Note:** This option can take several minutes to several hours based on the sources and targets in the PowerCenter repository. You can use the option to check whether the import from PowerCenter repository works.
--connections	-	Refreshes connection metadata for the target repository.
--folderList	-	Refreshes the folder list for the repository.
--allFolders	-	Refreshes source and target metadata in all folders in the repository.

Option	Argument	Description
--folder	FolderA	Refreshes source and target metadata for the named folders. Use commas to separate multiple folder named. **Note:** If the folder name or external ID contains commas, spaces, or special characters, you will need to use different syntax.
--dryrun	-	Verifies if the import from the PowerCenter Repository works. The dryrun command reads the sources and targets in the PowerCenter Repository Service, but does not write the objects into the Data Validation Option repository.

Related Topics:

- "Repositories Overview" on page 28
- "Rules and Guidelines for Special Characters" on page 191
- "ExportMetadata" on page 180
- "RunTests" on page 187
- "InstallTests" on page 182

RunTests

Runs all tests for one or more single-table objects or table-pair objects. You can run this command on Windows or UNIX.

For example, to run tests for a table pair with the external ID "abc123," enter the following command:

```
DVOCmd RunTests abc123
```

The RunTests command uses the following syntax:

```
DVOCmd RunTests external_ID [external_ID …] [--confdir conf_dir] [--username User name] [--password Password] [--ePassword Password Environment Variable] [--namespace Namespace] [--email email_ID,...] [--sendEmail NotPass] [--cacheSize CACHESIZE] [--folderNames]
```

The following table describes RunTests options and arguments:

Option	Argument	Description
-	external_ID	Required if you do not use folderNames. External ID of a single-table object or a table-pair object. You can separate multiple external IDs with spaces. **Note:** If the folder name or external ID contains commas, spaces, or special characters, you will need to use different syntax.
--confdir	conf_dir	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Enclose the file path in quotes.

Option	Argument	Description
--username	User name	User name in the Informatica domain for which you configured Informatica authentication. Required if you configure Informatica authentication.
--password	Password	Password for the user name. Required if you configure Informatica authentication. If you set the password option, then do not set the ePassword option.
--ePassword	Password Environment Variable	Encrypted form of the password of the Informatica user name. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the ePassword option, then do not set the password option. If the command contains both the password and ePassword options, then the Data Validation Option uses the ePassword and ignores the password.
--namespace	Namespace	Namespace of the LDAP domain. Required if you configure Informatica Authentication and the domain uses LDAP authentication.
--email	email_ID	The email address to which Data Validation Option sends an email when the tests complete. You can provide multiple email addresses separated by commas. The email provides a link to the test results and specifies whether the tests pass or fail. **Note:** If the PowerCenter Integration Service is running on Windows, configure the outgoing mail server with the following custom properties: SMTPServerAddress, SMTPPortNumber, SMTPFromAddress, and SMTPServerTimeout. To send mail with Microsoft Outlook, enter the Microsoft Exchange profile in the MSExchangeProfile configuration property in the PowerCenter Integration Service.
--sendEmail	NotPass	Data Validation Option sends an email if a test fails.
--cacheSize	CACHESIZE	Memory allocation to generate transformations in PowerCenter mappings. Increase the cache size for tests that contain multiple joins and lookups. Default is 20 MB for the data cache and 10 MB for the index cache. Specify "Auto" to enable PowerCenter to compute the cache size. When you run the RunTests command and set the cache size, Data Validation Option installs the tests in the repository before it runs the test.
--folderNames	folderA	Required if you do not use external_ID. Runs all objects in the specified folders. Separate multiple folder names with commas **Note:** If your folder name or external ID contains commas, spaces, or special characters, you must use different syntax. If the folders have a large number of objects, the PowerCenter Integration Service places them in a queue to run. The PowerCenter Integration Service limits the number of sessions that can run concurrently based on the Max Concurrent Runs property configured in the Data Validation preferences.

The DVOCmd RunTests command returns the following output:

```
ERROR: TablePairs: <number>; Pass: <number>; Fail: <number>; No Result: <number>;
Install/Run error: <number>.
```

The following table describes the output:

Output	Description
TablePairs	Number of table-pair objects or single-table objects that you run in the RunTests command.
Pass	Number of table-pair objects or single-table objects for which all tests completed successfully.
Fail	Number of table-pair objects or single-table objects for which at least one test failed.
No Result	Number of table-pair objects that returned no matching records based on the table join.
Install/Run Error	Number of table-pair objects or single-table objects for which Data Validation Option could not generate or install mappings successfully.

If you run the DVOCmd RunTests command from a shell script or batch file, the command returns a return code. For example, the command returns the return code to the %errorlevel% environment variable on Windows. The return code represents the status of all tests that the DVOCmd RunTests command runs. If you run tests for multiple single-table objects or table-pair objects, the return code applies to all tests for all single-table objects or all table-pair objects.

The following table describes the possible return codes for the DVOCmd RunTests command:

Return Code	Description
0	All tests are successful.
1	At least one test fails. A test fails if it fails to install, fails to run, does not return any result, or does not meet the test condition.

Related Topics:

- "Rules and Guidelines for Special Characters" on page 191
- "ExportMetadata" on page 180
- "RefreshRepository" on page 186
- "InstallTests" on page 182

Running Multiple Table Pair Objects in Parallel

To run multiple table pair objects in parallel, you must first install the tests, and then run then the tests.

The following code is a sample batch file for PowerCenter version 10.2.0, 10.2 HotFix 2, or 10.4.0:

```
SET DVO_PC_VERSION=<10.2.0_10.2.0.2_or_10.4.0>
set INFA_TRUSTSTORE=E:\Informatica\10.0.0\clients\shared\security
set INFA_DOMAINS_FILE=E:\Informatica\10.0.0\clients\powercenterclient\domains.infa
set DV_CONFIG_DIR=C:\Users\Administrator\DataValidator
cd\
E:

cd E:\Informatica10.4.0\DVO\

DVOCmd InstallTests  t2 t3 t4
START /B DVOCmd RunTests t2
START /B DVOCmd RunTests t3
START /B DVOCmd RunTests t4
```

Cache Settings

You can configure the cache settings for the PowerCenter transformations that Data Validation Option generates for the tests.

Data Validation Option generates a PowerCenter mapping with Joiner transformations and Lookup transformations for a test that contains joins and lookups. Joiner transformations and Lookup transformations require a large amount of memory. Configure the cache settings for a complex test that contains joins and lookups.

The value that you specify as cache settings for a test persists until you modify the test. For information about optimal cache setting for the tests, see the *PowerCenter Advanced Workflow Guide*.

UpdateInformaticaAuthenticationConfiguration

Updates the Informatica authentication properties in the Data Validation Option schema. You can run this command on Windows. After you upgrade Data Validation Option, if you use Informatica authentication, you can run the command to connect to the domain.

The UpdateInformaticaAuthenticationConfiguration command uses the following syntax:

```
DVOCmd UpdateInformaticaAuthenticationConfiguration
[--confdir CONFDIR]
[--infaHostName INFAHOSTNAME]
[--infaHttpPort INFAHTTPPORT]
[--isSecureConnection ISSECURECONNECTION]
[--username USERNAME]
[--password PASSWORD]
[--ePassword PASSWORD ENVIRONMENT VARIABLE]
[--namespace NAMESPACE]
```

You can get these parameters from the nodemeta.xml file available in the following location:

```
<Informatica installation directory>/server/isp
```

The following table describes UpdateInformaticaAuthenticationConfiguration options and arguments:

Option	Argument	Description
--confdir	CONFDIR	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.
--infaHostName	INFAHOSTNAME	Host name of the Informatica gateway.
--infaHttpPort	INFAHTTPPORT	Port number that you use to connect to the Informatica gateway.
--isSecureConnection	ISSECURECONNECTION	Set the argument as true if TLS is enabled in the Informatica gateway.
--username	USERNAME	Informatica administrator user for the Informatica gateway.

Option	Argument	Description
--password	PASSWORD	Password of the Informatica administrator account. If you set the **password** option, then do not set the **ePassword** option.
--ePassword	PASSWORD ENVIRONMENT VARIABLE	Encrypted form of the password of the Informatica administrator account. Use the DVOCmd EncryptPassword command to encrypt the password. Set an environment variable to the encrypted password and use the variable in the ePassword option. The command replaces the environment variable with the encrypted password and then decrypts the password. If you set the **ePassword** option, then do not set the **password** option. If the command contains both the **password** and **ePassword** options, then the Data Validation Option uses the **ePassword** and ignores the **password.**
--namespace	NAMESPACE	Namespace of the LDAP domain. Required if you configure Informatica Authentication with LDAP domain.

UpgradeRepository

Upgrades the Data Validation Option repository. Use this command when you upgrade from a previous version of Data Validation Option. You can run this command on Windows or UNIX.

The UpgradeRepository command uses the following syntax:

```
DVOCmd UpgradeRepository [--confdir CONFDIR]
```

The following table describes the UpgradeRepository option and argument:

Option	Argument	Description
--confdir	CONFDIR	The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. If Windows directory contains spaces, enclose the file path in quotes.

Rules and Guidelines for Special Characters

You must use different syntax for commands that contain folder names or external IDs with commas, special characters, or spaces.

Folder Names and External IDs Containing Commas

Follow these rules and guidelines when you run commands on Windows:

- Place single quotes around folder names that contain commas.
- Place double quotes around the entire argument.

- Place a back-slash before each of the two double quotes.

For example, if the folder names are test1 and test,2 and test3, you will use the following syntax:

```
RunTests --folderNames \"test1,'test,2',test3\"
```

Follow these rules and guidelines when you run commands on Linux:

- Place single quotes around folder names that contain commas.
- Place double quotes around the entire argument.

For example, if the folder names are test1 and test,2 and test3, you will use the following syntax:

```
RunTests --folderNames "test1,'test,2',test3"
```

Folder Names and External IDs Containing Spaces or Special Characters

Follow these rules and guidelines when you run commands on Windows:

- Place double quotes around the entire argument.
- Place a back-slash before each of the two double quotes

For example, if the folder names are test1 and test@2 and test<space>3, you will use the following syntax:

```
RunTests --folderNames \"test1,'test@2',test<space>3\"
```

Follow these rules and guidelines when you run commands on Linux:

- Place double quotes around the entire argument.

For example, if the folder names are test1 and test@2 and test<space>3, you will use the following syntax:

```
RunTests --folderNames "test1,'test,2',test3"
```

Related Topics:

- "ExportMetadata" on page 180
- "RefreshRepository" on page 186
- "RunTests" on page 187
- "InstallTests" on page 182

CHAPTER 19

Troubleshooting

This chapter includes the following topics:

Troubleshooting Overview

When you run a test, Data Validation Option performs the following tasks:

1. Creates a mapping in the specified PowerCenter folder.
2. Creates the PowerCenter workflow.
3. Runs the PowerCenter session.

Besides initial installation problems, Data Validation Option errors can occur in one of the following steps:

Installation Error

Data Validation Option cannot create a PowerCenter mapping.

Run Error

Data Validation Option cannot create or run a PowerCenter workflow.

No Results

The PowerCenter session runs but fails, or there are no results in the results database.

Troubleshooting Initial Errors

This section assumes that you have installed Data Validation Option.

I get the following out-of-memory error when I invoke Data Validation Option: The JVM could not be started. The maximum heap size (-Xmx) might be too large or an antivirus or a firewall tool could block the execution.

This error occurs in the following situations:

- The JVM that Data Validation Option uses is unable to reserve the maximum Java heap memory.

- The system has memory but is unable to allocate it.

By default, Data Validation Option allocates a maximum heap memory of 1024 MB. This error occurs if the system on which you install Data Validation Option, is unable to reserve the specified maximum heap memory.

To resolve this error, override the default value to value lower than 1024 MB. Perform one of the following steps to lower the value:

- Start Data Validation Option from the command line using following command:

  ```
  "<Data Validation Option installation directory>\DataValidator.exe" -J-Xmx786m -J-Xms256m
  ```

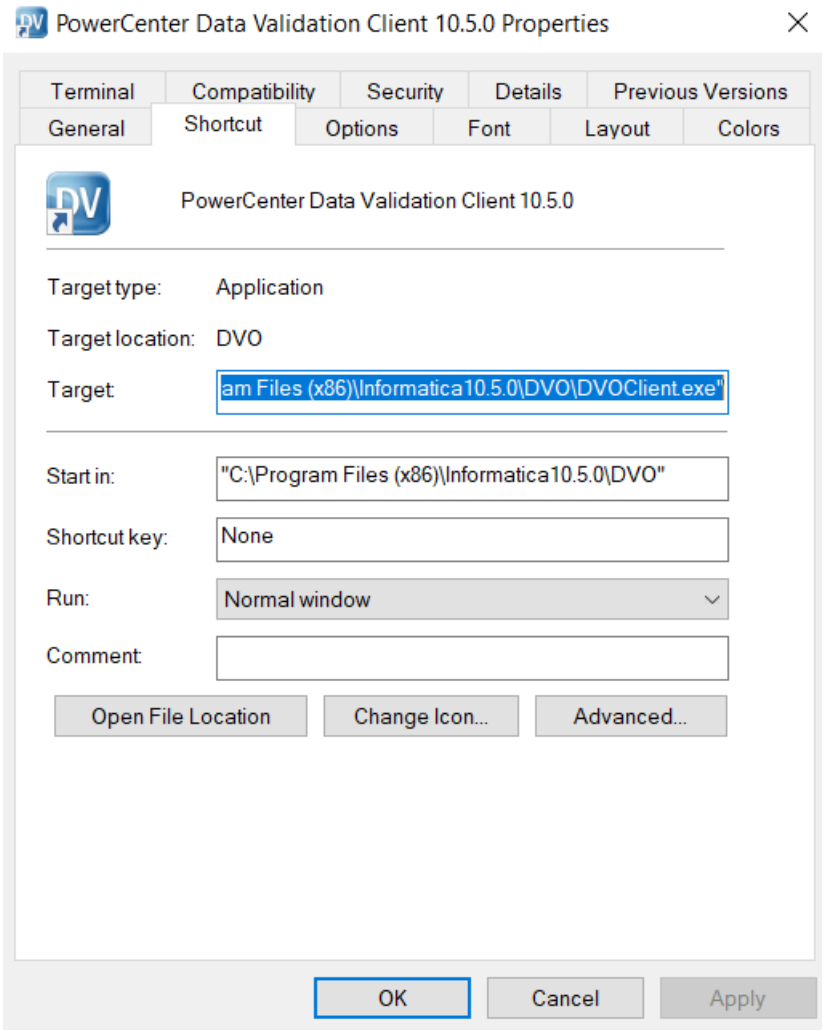
- Create the `DVOClient.vmoptions` or `DVOClientCore.vmoptions` file in the Data Validation Option root directory and add the following line:

  ```
  -Xmx<values>
  ```

- Edit the shortcut to the Datavalidator.exe and add the following parameter in the Target box:

  ```
  "<Data Validation Option installation directory>\DataValidator.exe" -J-Xmx1024m -J-Xms256m
  ```

The following image shows the parameter in the Target box:



This section assumes that you have installed Data Validation Option and not executed any successful tests. This section also assumes that the first test is a simple test that does not contain expressions or SQL views.

The following table describes common initial errors:

Error	Possible Cause and Solution
Cannot connect to the Data Validation Option repository	Database credentials are incorrect. Check the server, port, and database name specified in the URL line. If the problem persists, contact your database administrator. If the problem is database username and password, the error message explicitly states this.
Cannot read the PowerCenter repository	Check the repository settings up to **Security Domain**. (Informatica Domain names are not used until later.) If you cannot resolve the error, contact the Informatica administrator. Another way to troubleshoot this error is by trying to log into the repository through the pmrep command line utility. If you get an out-of-memory error when you access large repositories, increase the Java heap size of the Data Validation Client from the command line. You can increase the Java heap size from the command line with the following command: `DVOClient.exe -J-Xmx<heapsize value>` Default is 1024 MB.
Installation Error	Verify that the Data Validation Option folder is closed in the Designer, Workflow Manager and the Repository Manager. The Workflow Monitor can be open. Verify that the INFA_HOME environment variable is set. Verify that the Data Validation Option folder exists.
Run Error	Verify that the Data Validation Option folder is closed. Check the Informatica Domain name and Integration Service names. Verify that they are running. Verify that the PowerCenter connection name (connection to the Data Validation Option repository) is correct, and that the user has the privilege to use it. Open the session log and look for session errors. Most session failures are caused by an incorrect connection. If the error is “Cannot get object class for dvo/infact/PivotPluginImpl,” the dvoct.jar file cannot be read either because it is not on the server, because of its privileges, or because the information entered in the Administrator tool is incorrect. Verify that the user has the privilege to use the connection to the Data Validation Option repository specified in the **Tools** > **Preferences** > **Data Validation Option** database. This will also be apparent in the session log. If you install PowerCenter 9.0.1 or earlier and the data source is SAP, install ABAP program on the mapping generated by the test in the Designer tool.
No Results	Verify that there is data in the data set you are analyzing. Tables should have records, and filters and joins should not result in an empty set. Verify that the connection to the Data Validation Option repository specified in the Workflow Manager points to the Data Validation Option repository.

Troubleshooting Ongoing Errors

This section assumes that successful tests have been created and run before the error occurred.

In general, you should always check for the following sources of errors:

- Incorrect connection
- The Data Validation Option folder is open in the Designer, Workflow Manager, or Repository Manager

The following table describes common ongoing errors:

Error	Possible Cause and Solution
Installation or run errors	Verify that the Data Validation Option folder is closed in the Designer, Workflow Manager and Repository Manager. Verify that the PowerCenter environment is functioning correctly, for example, services are running and the repository is up. If the error occurred right after you created an expression either in a test editor dialog box or as a WHERE clause, check the syntax. Open the session log and verify that the PowerCenter connections are correct.
No results	Verify that there is data in the data set you are analyzing. Tables should have records. Filters or joins should not result in an empty set.
Inability to generate reports	Verify that you have read and write permissions on the Data Validation Option installation directory and subdirectories.
Inability to copy folders	Verify that the repository, data sources, and folders that contain the data sources have identical names in the source and the target workspaces. Object names in Data Validation Option are case sensitive. Verify that all data sources associated with the objects to copy exist in the target workspace in the same location and that the names match.

Troubleshooting Command Line Errors

I ran a DVOCmd command that got an error and used the redirection operator to write the messages to a file. The redirection operator does not redirect all messages to the output file.

When you run a DVOCmd command, the command line utility writes regular messages to the STDOUT output stream and writes error messages to the STDERR output stream. You use the redirection operator to write messages to an output file. To merge messages from both output streams, enter "2>&1" after the output file name.

For example, you encounter an error while refreshing folder "MyFolder" in Data Validation Option repository "DVORepo." To write all messages, including the error messages, to a text file called "Log.txt," use the following command:

```
DVOCmd RefreshRepository DVORepo --folder MyFolder > C:\Log.txt 2>&1
```

To append messages to an existing log file called "DVOLog.txt," use the following command:

```
DVOCmd RefreshRepository DVORepo --folder MyFolder >> C:\DVOLog.txt 2>&1
```

APPENDIX A

Datatype Reference

This appendix includes the following topics:

Test, Operator, and Datatypes Matrix for Table Pair Tests

Table pair tests use the following datatypes:

- String
- Numeric
- Date/time
- Binary

The following table describes the operators and datatypes for table pair tests:

Test	Operators Allowed	Datatypes Allowed: Approx. Operator	Datatypes Allowed: All Other
COUNT	All	string, numeric, date/time, binary/other	string, numeric, date/time, binary/other
COUNT_DISTINCT	All	string, numeric, date/time	string, numeric, date/time
COUNT_ROWS	All	string, numeric, date/time, binary/other	string, numeric, date/time, binary/other
MIN	All	numeric	string, numeric, date/time
MAX	All	numeric	string, numeric, date/time
AVG	All	numeric	numeric
SUM	All	numeric	numeric

Test	Operators Allowed	Datatypes Allowed: Approx. Operator	Datatypes Allowed: All Other
SET_AinB	--	string, numeric, date/time	string, numeric, date/time
SET_BinA	--	string, numeric, date/time	string, numeric, date/time
SET_AeqB	--	string, numeric, date/time	string, numeric, date/time
VALUE	All	numeric	string, numeric, date/time
OUTER_VALUE	All	numeric	string, numeric, date/time

Note: SET tests do not use operators and allow string, numeric and date/time datatypes.

Test, Operator, and Datatypes Matrix for Single-Table Constraints

Single-table constraints use the following datatypes:

- String
- Numeric
- Date/time
- Binary/other

The following table describes the operators and datatypes for single-table constraint tests:

Test	Operators Allowed	Datatypes Allowed	Result Expression Datatype
COUNT	All	string, numeric, date/time, binary/other	numeric
COUNT_DISTINCT	All	string, numeric, date/time	numeric
COUNT_ROWS	All	string, numeric, date/time, binary/other	numeric
MIN	All	string, numeric, date/time	numeric
MAX	All	string, numeric, date/time	numeric
AVG	All	numeric	numeric
SUM	All	numeric	numeric
VALUE	All	string, numeric, date/time	string, numeric, date/time
FORMAT	=, <>	string, numeric, date/time	string
UNIQUE	--	string, numeric, date/time	--

Test	Operators Allowed	Datatypes Allowed	Result Expression Datatype
NOT_NULL	--	string, numeric, date/time	--
NOT_BLANK	--	string, numeric, date/time	--

Selecting a Valid PowerCenter Transformation Data Type for an Expression

The PowerCenter transformation data type that you select for a field expression must be compatible with the data type of the field in the expression. To know the data type of a field, view the properties of the field in the Expression Editor. In the Expression Editor, select the appropriate PowerCenter transformation data type for the expression.

For example, if PowerCenter imports a table from an Oracle database, and the table includes a date field called PurchaseDate, PowerCenter converts the field to the `Date/Time` transformation data type. If you use the PurchaseDate field in an expression, set the data type of the expression to `Date/Time`. Set the precision and scale of the expression to the precision and scale of the `Date/Time` transformation data type.

The following table shows the transformation data types to which PowerCenter converts data types from various relational sources:

PowerCenter Transformation Data Type	IBM DB2 Data Type	Oracle Data Type	Microsoft SQL Server Data Type	Sybase Data Type	Teradata Data Type	Netezza Data Type
Bigint	Bigint	-	Bigint	Bigint	Bigint	BigInt
Binary	Blob Char for bit data, Varchar for bit data	Blob, Raw(L), LongRaw	Binary(L), Image, Timestamp, Varbinary(L)	Binary (n), Image, Timestamp, Varbinary (n)	Byte, Varbyte	-
Date/Time	Date, Time, Timestamp	Date, Timestamp	Datetime, Datetime2, Smalldatetime	Date, Datetime, Smalldatetime, Time, Timestamp (Sybase IQ)	Date, Time, Timestamp	Date, Time, Timestamp
Decimal	Decimal(P,S), Numeric(P,S)	Number(P,S)	Decimal(P,S), Money, Numeric(P,S), Smallmoney	Decimal (P,S), Money, Numeric (P,S), Smallmoney	Decimal	Numeric
Double	Float	Number	Float	Float	Float	Float8, Float4
Integer	Integer	-	Int, Numeric Identity	Int	Integer	Integer

PowerCenter Transformation Data Type	IBM DB2 Data Type	Oracle Data Type	Microsoft SQL Server Data Type	Sybase Data Type	Teradata Data Type	Netezza Data Type
Nstring	Graphic	Nchar, Nvarchar2	Nchar, Nvarchar, Sysname	Nchar (n), Nvarchar (n), Unichar, Univarchar	-	-
Ntext	Dbclob, Long Vargraphic	Nclob	Ntext	-	-	-
Real	-	-	Real	Real	-	Real
Small integer	Smallint	-	Smallint, Tinyint	Smallint, Tinyint	Byteint, Smallint	ByteInt, SmallInt
String	Char(L), Long Varchar, Varchar, Vargraphic	Char(L), Varchar(L), Varchar2(L)	Bit, Char(L), Varchar(L)	Bit, Char (n), Varchar (n)	Char, Varchar	Bool, Char, NChar(m), NVarchar(m), Varchar
Text	Clob	Clob, Long, XMLType based CLOB storage	Text	Text	-	-

The following table shows the PowerCenter transformation data types to which PowerCenter converts application data types:

PowerCenter Transformation Data Type	SAS Data Type	SAP Data Type	Salesforce Data Type
Bigint	Numeric	-	-
Binary	-	LRAW, PREC, RAW	-
Date/time	Date, Time, Datetime	ACCP, DATS, TIMS	Date, DateTime, Time
Decimal	Numeric	CURR, DEC	Currency, Double, Percent
Double	Numeric	FLTP	-
Integer	Numeric	INT1, INT2, INT4	Boolean, Int
Nstring	-	-	-
Ntext	-	-	-
Real	Numeric	-	-
Small integer	Numeric	-	-

PowerCenter Transformation Data Type	SAS Data Type	SAP Data Type	Salesforce Data Type
String	String	RAWSTRING, SSTRING, STRG	AnyType, Base64, DataCategoryGroupReference, Email, Encrypted String, ID, Multipicklist, Picklist, Reference, String, Textarea, Url
Text	-	CHAR, CLNT, CUKY, LANG, LCHR, NUMC, QUAN, UNIT, VARC	-

The following table shows the PowerCenter transformation data types to which PowerCenter converts flat file and COBOL data types:

PowerCenter Datatype	Flat File Data Type	COBOL Data Type
Bigint	Bigint	-
Binary	-	-
Date/time	Datetime	-
Decimal	Number	Number
Double	Double	-
Integer	Integer	-
Nstring	Nstring	-
Ntext	-	-
Real	-	-
Small integer	-	-
String	String	Nstring, String
Text	-	-

Properties of PowerCenter Transformation Data Types

When you create a field expression, you must know the precision values and scale values to set for the PowerCenter transformation data type that you select for the expression. Incorrect values can lead to errors

and result truncation. You can change the values of precision and scale for some PowerCenter transformation data types.

Precision is the maximum number of significant digits for numeric data types, or the maximum number of characters for string data types. Precision includes scale. Scale is the maximum number of digits after the decimal point for numeric values. Therefore, the value 11.47 has a precision of 4 and a scale of 2. The string Informatica has a precision (or length) of 11.

Table 1. The following table shows the PowerCenter transformation data type properties:

DataType	Size in Bytes	Description
Bigint	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision of 19, scale of 0 Integer value.
Binary	Precision	1 to 104,857,600 bytes You cannot use binary data for COBOL or flat file sources
Date/Time	16 bytes	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision of 29, scale of 9 (precision to the nanosecond) Combined date/time value.
Decimal	8 bytes (if high precision is off or precision is greater than 28) 16 bytes (if precision <= 18 and high precision is on) 20 bytes (if precision > 18 and <= 28)	Decimal value with declared precision and scale. Scale must be less than or equal to precision. Precision 1 to 28 digits, scale 0 to 28
Double	8 bytes	Double-precision floating-point numeric value. You can edit the precision and scale.
Integer	4 bytes	-2,147,483,648 to 2,147,483,647 Precision of 10, scale of 0 Integer value.
Nstring	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Ntext	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Real	8 bytes	Precision of 7, scale of 0 Double-precision floating-point numeric value.
Small Integer	4 bytes	-32,768 and 32,767 Precision of 5, scale of 0 Integer value.

DataType	Size in Bytes	Description
String	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Text	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.

The following table shows the PowerCenter transformation data type properties that you can change:

PowerCenter Transformation Data Type	Precision Can Be Changed	Scale Can Be Changed
Bigint	No	-
Binary	Yes	-
Date/time	No	No
Decimal	Yes	Yes
Double	No	-
Integer	No	-
Nstring	Yes	-
Ntext	Yes	-
Real	No	-
Small integer	No	-
String	Yes	-
Text	Yes	-

APPENDIX B

Reporting Views

This appendix includes the following topics:

Reporting Views Overview

The Data Validation Option reports run against the reporting views. You can build custom reports based on the reporting views.

The Data Validation Option installer installs the reporting views in the Data Validation Option repository. If you create a custom report, build the report based on the views. Do not build reports based on the underlying database tables because the Data Validation Option repository metadata can change between versions.

Data Validation Option stores version history for objects and test runs in the Data Validation Option repository. The Data Validation Option repository contains versions of single-table constraints, table pairs, join views, lookup views, SQL views, and the tables and files included in each of these objects. The Data Validation Option repository also stores the history of each test run. When you create a report, you can use the *_live and *_latest view columns to get the latest version of an object.

Using the Reporting Views

The reporting views contain information about table pairs, single-table constraints, join views, lookup views, SQL views, tests, and test runs. To extract information from the reporting views, you can build custom reports in a business intelligence tool or run SQL queries.

When you build a report, you specify an SQL query based on one or more reporting views. In the SQL query, you can transform the data based on the business requirements. For example, the results_summary_view contains the tr_start_time and tr_finish_time columns. The columns store the time in milliseconds since Jan 1, 1970.

You can run the following SQL statement to convert the milliseconds to a date and time format for a custom report:

```
SELECT TO_DATE('1970-01-01', 'YYYY-MM-DD')+(numtodsinterval(to_number(tr_finish_time)/
1000,'SECOND'))
FROM results_summary_view
```

Sample Queries for Custom Reports

You can run SQL queries against the reporting views to generate custom reports.

Test Run

You can run the following SQL query to get test run information for a specific test:

```
SELECT DISTINCT tr.*, tp.DESCRIPTION
FROM TABLE_PAIR tp, TEST_RUN tr
WHERE tr.TABLE_PAIR_OBJ_ID = tp.OBJ_ID
and tp.DESCRIPTION = <unique table pair or single-table constraint description>
```

Latest Test Run

You can run the following SQL query to get the latest test run information:

```
SELECT DISTINCT tr.*, tp.DESCRIPTION
FROM
TEST_RUN tr, TABLE_PAIR tp
WHERE
tr.IS_LATEST_RUN = 1
and tr.TABLE_PAIR_OBJ_ID = tp.OBJ_ID
and tp.DESCRIPTION = <unique table pair or single-table constraint description>
```

Bad Records for All Table Pairs and Single-Table Constraints

If you configured Data Validation Option to store bad records in the DVO schema, you can run the following SQL query to get the bad records for all table pairs and single-table constraints:

```
SELECT rsv.*,rsbd.*
FROM RESULTS_SUMMARY_VIEW rsv
LEFT OUTER JOIN RS_BAD_RECORDS_VIEW rsbd
ON rsv.TR_ID = rsbd.TR_ID
and rsv.TC_INDEX = rsbd.TC_INDEX
```

Bad Records for Latest Test Runs for a Table Pair or Single-Table Constraint

If you configured Data Validation Option to store bad records in the DVO schema, you can run the following SQL query to get the bad records for the latest test runs for a table pair or single-table constraint:

```
SELECT rsv.*,rsbd.*
FROM RESULTS_SUMMARY_VIEW rsv
LEFT OUTER JOIN RS_BAD_RECORDS_VIEW rsbd
ON rsv.TR_ID = rsbd.TR_ID
and rsv.TC_INDEX = rsbd.TC_INDEX
where
rsv.TP_DESCRIPTION = <unique table pair or single-table constraint description>
and rsv.tr_is_latest=1
```

results_summary_view

The results_summary_view view contains metadata for single-table constraints, table pairs, tests, and connections. The view also contains results of each test in each version of a single-table constraint or table pair.

Note: For each single-table constraint, the *_a columns in the view contain information about the table on which the single-table constraint is based. The *_b columns are null for single-table constraints.

The following table describes the prefixes in the names of the view columns:

Column Prefix	Description
tp_	Single-table constraint or table pair information.
tc_	Test condition information.
tc_rs_	Test results information.
tr_	Run-time information for single-table or table-pair tests.
ti_	Test installation information. Test installation refers to the creation and installation of a PowerCenter mapping based on a single-table constraint or table pair.
conn_	Connection information for the table or file in a single-table constraint or table pair.

The following table describes the view columns that provide run-time information about each test in single-table constraints and table pairs:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_state	Result state. Contains one of the following values: - 2. Install error. - 4. Run success. - 5. Run error.
tr_start_time	Start time of the single-table or table-pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_finish_time	End time of the single-table or table-pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the single-table or table-pair test. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
tr_finish_timestamp	End time of the single-table or table-pair test. The time corresponds to the millisecond value in the tr_finish_time column and appears in standard date time form.
tr_is_latest	Indicates whether this is the latest run of a given single-table or table-pair test. Contains one of the following values: - 0. Not latest. - 1. Latest.

View Column	Description
tr_error_msg	Error message if the test run has an error.
ti_id	ID for test installation. Do not use in reports.
ti_folder_name	Name of the PowerCenter folder that contains the PowerCenter mapping associated with the test.
ti_mapping_name	Name of the PowerCenter mapping associated with the test.
ti_session_name	Name of the PowerCenter session that contains the PowerCenter mapping associated with the test.
ti_workflow_name	Name of the PowerCenter workflow that contains the PowerCenter mapping associated with the test.

The following table describes the view columns that provide information about each single-table constraint and table pair:

View Column	Description
tp_user_id	User ID of the person who ran the test.
tp_username	User name of the person who ran the test.
tp_obj_id	ID of the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_name	Description of the single-table constraint or table pair. Same as tp_description.
tp_folder_id	ID of the folder that contains the single-table constraint or table pair.
tp_folder_name	Name of the folder that contains the single-table constraint or table pair.
tp_time_stamp	Time that the single-table constraint or table pair definition was last modified. Time is represented as the number of milliseconds since 1970 UTC.
tp_edit_timestamp	Time at which the single-table constraint or table pair definition was last modified. The time corresponds to the millisecond value in the tp_time_stamp column and appears in standard date time form.
tp_comments	Comments about the single-table constraint or table pair.
tp_description	Description of the single-table constraint or table pair. Same as tp_name.
tp_table_a, tp_table_b	Name of the file, table, or view included in the single-table constraint or table pair. File and table names also include the PowerCenter repository directory.
tp_table_version_a tp_table_version_b	Version of the join view, lookup view, or SQL view included as a table in the single-table constraint or table pair.

View Column	Description
tp_type_a, tp_type_b	Type of table included in the single-table constraint or table pair. Contains one of the following values: - 1. Relational. - 2. Flat file or XML file. - 3. SQL view. - 4. Lookup view. - 6. Join view.
tp_conn_name_a tp_conn_name_b	Name of the PowerCenter connection for the table or file in the single-table constraint or table pair.
tp_owner_name_a, tp_owner_name_b	Override of the owner or name of the relational table.
tp_src_dir_a, tp_src_dir_b,	Directory that contains the file when the connection is based on a flat file or XML file. The path is relative to the machine that runs the PowerCenter Integration Service.
tp_src_file_a, tp_src_file_b	Name of the file when the connection is based on a flat file or XML file. The file name includes the file extension.
tp_in_db_a, tp_in_db_b	Indicates whether the input data is already sorted or the database sorts the input data and performs aggregation. Contains one of the following values: - 0. Input data already sorted. - 1. Database sorts data and performs aggregation.
tp_where_clause_a, tp_where_clause_b	WHERE clause for the table included in the single-table constraint or table pair.
tp_is_where_clause_dsq_a, tp_is_where_clause_dsq_b	Indicates whether the database processes the WHERE clause. If not, the PowerCenter Integration Service performs the sampling. Contains one of the following values: - 0. False. - 1. True.
tp_is_large_a, tp_is_large_b	For a table pair, indicates whether the table is larger than the other table in the table pair. Contains one of the following values: - 0. False. Value is also '0' for single-table constraints. - 1. True.
tp_join_list_str	For a joined table pair, comma-separated list of join conditions defined in the table pair. For a single-table constraint, comma-separated list of columns used to define the primary key for the table in the single-table constraint.
tp_external_id	External ID of the single-table constraint or table pair.
tp_enable_sampling	Indicates whether data sampling is enabled. Contains one of the following values: - 0. False. - 1. True.
tp_sample_percentage	Percentage of rows that are to be included in the data sample.
tp_sample_seed	Starting value used to generate a random number for data sampling.

View Column	Description
tp_apply_sample_to	Table on which you want to perform data sampling.
tp_enable_native_sampilng	Indicates whether the database performs the data sampling. If not, the PowerCenter Integration Service performs the sampling. Contains one of the following values: - 0. False. - 1. True.

The following table describes the view columns that provide information about each test:

View Column	Description
tc_index	ID of the test in a single-table constraint or table pair.
tc_description	Description of the test.
tc_comment	Comments about the test.
tc_type	Type of test. Contains one of the following values: - AGG - SET_AinB - SET_BinA - SET_AeqB - SET_ANotInB - VALUE - OUTER_VALUE - FORMAT - UNIQUE - NOT_NULL - NOT_BLANK Displays 'AGG' for the aggregate tests COUNT, COUNT_DISTINCT, COUNT_ROWS, MIN, MAX, AVG, and SUM.
tc_agg_func	Type of aggregate test. Contains one of the following values: - COUNT - COUNT_DISTINCT - COUNT_ROWS - MIN - MAX - AVG - SUM
tc_column_a, tc_column_b	Field in the test condition .
tc_operator	Operator in the test condition.
tc_condition_a, tc_condition_b	Condition used to filter records for the test.
tc_tables_type	Type of table. Contains one of the following values: - 0. Table pair. - 1. Single-table constraint.

View Column	Description
tc_threshold	Allowable margin of error for an aggregate or value test that uses the approximate operator. The value is an absolute value or a percentage value.
tc_max_bad_records	Maximum number of records that can fail the comparison for a test to pass. The value is an absolute value or a percentage value.
tc_is_case_insensitive	Ignores case when you run a test that compares string data. Contains one of the following values: - 0. False. - 1. True.
tc_is_treat_nulls_equal	Allows null values in two tables to be considered equal. Contains one of the following values: - 0. False. - 1. True.
tc_is_trim_right_ws	Ignores trailing spaces when you run a test that compares string data. Data Validation Option does not remove the leading spaces in string data. Contains one of the following values: - 0. False. - 1. True.
tc_expression_a, tc_expression_b	An expression that represents a field value.

The following table describes the view columns that provide information about the results of each test:

View Column	Description
tc_rs_result	Result of a test. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.
tc_rs_failure_count	Number of bad records.
tc_rs_processed_count	Number of records processed.
tc_rs_count_rows_a, tc_rs_count_rows_b	Number of records in the table or file.
tc_rs_agg_value_a, tc_rs_agg_value_b	Value of the aggregate test. The value is null for other types of tests.

The following table describes the view columns that provide information about the each connection used in a single-table constraint and table pair:

View Column	Description
conn_connection_prop_a, conn_connection_prop_b	Connection string for relational connections. Null for all other connection types.
conn_connection_type_a, conn_connection_type_b	Type of connection. Contains one of the following values: - application - DB2 - Microsoft SQL Server - Netezza - Oracle - PWX DB2zOS - PWX NRDB Lookup - relational - SAP R3 - Salesforce Connection - SAS Connection - Sybase - Teradata The "application" value represents Adabas, DB2 for i5/OS, IMS, sequential files, and VSAM data sources. The "relational" value represents database connections that use an ODBC connection.
conn_user_name_a, conn_user_name_b	User name for the connection.

rs_bad_records_view

The rs_bad_records_view view contains information about bad records for tests. You can join this view with the results_summary_view view based on the tr_id and tc_index columns to get bad records associated with a single-table constraint, table pair, or test run.

Note: For each single-table constraint, the *_a columns in the view contain information about the table on which the single-table constraint is based. The *_b columns are null for single-table constraints.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tc_index	ID of the test in a single-table constraint or table pair.
tc_rs_br_key_a, tc_rs_br_key_b	Key of the bad record.
tc_rs_br_value_a, tc_rs_br_value_b	Value that does not meet the test condition.

View Column	Description
tc_rs_br_value_delim_a, tc_rs_br_value_delim_b	String value with leading and trailing delimiters that indicate the presence of whitespace.
tc_rs_br_value_difference	Difference in value for numeric data.
tc_rs_br_value_diff_perc	Percentage of difference in value for numeric data. Data Validation Option uses the following formula to calculate the percentage of difference: `(<larger value> - <smaller value>) ÷ <larger value> * 100`

results_id_view

The results_id_view view contains overall results for all tests in a single-table constraint or table pair. This view aggregates test-level information from the results_summary_view view.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_is_latest	Indicates whether this is the latest test run for the single-table constraint or table pair. Contains one of the following values: - 0. Not latest. - 1. Latest.
tr_start_time	Start time of the test run. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the test run. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
ti_id	ID for test installation. Do not include in reports.
tp_obj_id	ID of the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_user_id	User ID of the person who ran the tests.
tp_rs_result	Result of all tests in the single-table constraint or table pair. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.

meta_sv_view

The meta_sv_view view contains information about all SQL views and the relational table columns included in each SQL view. The view contains one record for each table column in each version of a SQL view.

The following table describes the view columns:

View Column	Description
sv_id	ID of the SQL view.
sv_name	Internal name of the SQL view.
sv_obj_id	ID of the table in the SQL view.
sv_version	Version of the SQL view.
sv_description	Description of the SQL view.
sv_comment	Comments about the SQL view.
sv_owner_name	Override for the owner name of the SQL view.
sv_conn_name	Name of the PowerCenter connection for all tables included in the SQL view.
sv_sql_query	SQL statement for the SQL view.
svf_name	Name of the column in the SQL view.
svf_business_name	Not used.
svf_datatype	Datatype of the column in the SQL view.
svf_precision	Precision of the column in the SQL view.
svf_scale	Scale of the column in the SQL view.
svf_is_key	Not used.

meta_lv_view

The meta_lv_view view contains information about all lookup views and the tables included in each lookup view. The view contains one record for each table in each version of a lookup view.

Note: For each single-table constraint, the *_a columns in the view contain information about the table on which the single-table constraint is based. The *_b columns are null for single-table constraints.

The following table describes the view columns:

View Column	Description
lv_id	ID of the lookup view.
lv_name	Internal name of the lookup view.
lv_obj_id	ID of the table in the lookup view.
lv_version	Version of the lookup view.
lv_tp_name	Name of the single-table constraint or table pair.
lv_time_stamp	Time that the lookup view was last modified. Time is represented as the number of milliseconds since 1970 UTC.
lv_comments	Comments about the lookup view.
lv_description	Description about the lookup view.
lv_table_a	Name of the source table or file.
lv_table_b	Name of the lookup table or file.
lv_type_a, lv_type_b	Type of data source for the table or file. Contains one of the following values: - 1. Relational. - 2. Flat file.
lv_conn_name_a, lv_conn_name_b	Name of PowerCenter connection for the table or file.
lv_owner_name_a, lv_owner_name_b	Override for the owner name of the table or file.
lv_src_dir_a, lv_src_dir_b	Directory that contains the file when the connection is based on a flat file or XML file. The path is relative to the machine that runs the PowerCenter Integration Service.
lv_src_file_a, lv_src_file_b	Name of the file when the connection is based on a flat file or XML file. The file name includes the file extension.
lv_where_clause_a, lv_where_clause_b	Not used.
lv_is_where_clause_dsq_a, lv_is_where_clause_dsq_b	Not used.
lv_join_list_str	Comma-separated list of source-to-lookup relationships.

meta_jv_view

The meta_jv_view view contains information about all join views and the tables included in each view. The view contains one record for each table in each version of a join view.

The following table describes the view columns:

View Column	Description
jv_id	ID of the join view.
jv_name	Internal name of the join view.
jv_obj_id	ID of the table in the join view.
jv_version	Version of the join view.
jv_description	Description of the join view.
jv_table_name	Name of the table in the join view.
jv_table_alias	Alias name of the table.
jv_table_join_type	Type of join. Contains one of the following values: - INNER - LEFT_OUTER - RIGHT_OUTER - FULL_OUTER
jv_table_position	Position of the table in the join.
jv_is_live	Indicates whether this version of the view is the latest. Contains one of the following values: - 0. False. - 1. True.

meta_ds_view

The meta_ds_view view contains metadata about Data Validation Option objects. Objects include single-table constraints, table pairs, SQL views, lookup views, and join views.

The following table describes the view columns:

View Column	Description
id	ID of the object.
object_name	Name of the object.
object_description	Description of the object.
object_folder_name	Folder that contains the object.

View Column	Description
object_id	ID of the object.
object_version	Version of the object in the Data Validation Option repository.
object_type	Type of object. Contains one of the following values: - 1. Single-table constraint or table pair. - 2. SQL view. - 3. Lookup view. - 4. Join view.
object_is_live	Indicates whether this version of the object is the latest. Contains one of the following values: - 0. False. - 1. True.
object_table_name	Name of the table included in the object.
object_user_name	User name under which the object metadata was imported from PowerCenter.
object_user_id	ID of the user.

meta_tp_view

The meta_tp_view view contains metadata about all versions of single-table constraints and table pairs. The view contains one record for each version of a single-table constraint or table pair.

Note: For each single-table constraint, the *_a columns in the view contain information about the table on which the single-table constraint is based. The *_b columns are null for single-table constraints.

The following table describes the view columns:

View Column	Description
tp_id	ID of the single-table constraint or table pair.
tp_name	Description of the single-table constraint or table pair.
tp_obj_id	ID of the single-table constraint or table pair.
tp_folder_name	Folder that stores the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_description	Description of the single-table constraint or table pair. Same as tp_name.
tp_time_stamp	Time that the single-table constraint or table pair was last modified. Time is represented as the number of milliseconds since 1970 UTC.
tp_comments	Comments for the single-table constraint or table pair.

View Column	Description
tp_external_id	External ID of the single-table constraint or table pair.
tp_table_a, tp_table_b	Name of the table or view. Table names also include the PowerCenter repository directory.
tp_conn_name_a tp_conn_name_b	Name of the PowerCenter connection for the table or file in the single-table constraint or table pair.
tp_owner_name_a, tp_owner_name_b	Override of the owner name for the table or file.
lv_src_dir_a, lv_src_dir_b	Directory that contains the file when the connection is based on a flat file or XML file. The path is relative to the machine that runs the PowerCenter Integration Service.
lv_src_file_a, lv_src_file_b	Name of the file when the connection is based on a flat file or XML file. The file name includes the file extension.
tp_where_clause_a, tp_where_clause_b	WHERE clause for the table included in a single-table constraint or table pair.
tp_is_where_clause_dsq_a, tp_is_where_clause_dsq_b	Indicates whether the database processes the WHERE clause. If not, the PowerCenter Integration Service performs the sampling. Contains one of the following values: - 0. False. - 1. True.
tp_join_list_str	Comma-separated list of all join fields created for the table pair. Value is null for single-table constraints.
tp_is_live	Indicates whether this version of the single-table constraint or table pair is the latest. Contains one of the following values: - 0. False. - 1. True.

meta_av_view

The meta_av_view view contains information about all aggregate views and the table columns included in each aggregate view. The view contains one record for each table column in each version of an aggregate view.

The following table describes the view columns:

View Column	Description
av_id	ID of the aggregate view.
av_name	Internal name of the aggregate view.

View Column	Description
av_obj_id	ID of the table in the aggregate view.
av_version	Version of the aggregate view.
av_description	Description of the aggregate view.
av_conn_name	Name of the PowerCenter connection for all tables included in the aggregate view.
avf_name	Name of the column in the aggregate view.
avf_business_name	Business name of the aggregate view.
avf_datatype	Datatype of the column in the aggregate view.
avf_precision	Precision of the column in the aggregate view.
avf_scale	Scale of the column in the aggregate view.
avf_is_key	Contains the value 1 if the column is the primary key or foreign key.

rs_jv_id_view

The rs_jv_id_view view contains information about single-table constraints and table pairs that are based on a join view. The view contains the run-time and test result information for each version of a single-table constraint or table pair.

The view might contain duplicate IDs. You can use SELECT DISTINCT to avoid duplicate IDs.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_is_latest	Indicates whether this is the latest run of a given single-table constraint or table pair. Contains one of the following values: - 0. Not latest. - 1. Latest.
tr_start_time	Start time of the single-table constraint or table pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the single-table constraint or table pair test. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
ti_id	ID of the test installation. Do not use in reports.
tp_obj_id	ID of the single-table constraint or table pair.

View Column	Description
tp_version	Version of the single-table constraint or table pair.
tp_user_id	User ID of the person who ran the test.
tp_rs_result	Single-table constraint or table pair result. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.
jv_id	ID of the join view.

rs_lv_id_view

The rs_lv_id_view view contains information about single-table constraints and table pairs that are based on a lookup view. The view contains the run-time and test result information for each version of a single-table constraint or table pair.

The view might contain duplicate IDs. You can use SELECT DISTINCT to avoid duplicate IDs.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_is_latest	Indicates whether this is the latest run of a given single-table constraint or table pair. Contains one of the following values: - 0. Not latest. - 1. Latest.
tr_start_time	Start time of the single-table constraint or table pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the single-table constraint or table pair test. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
ti_id	ID of the test installation. Do not use in reports.
tp_obj_id	ID of the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_user_id	User ID of the person who ran the test.

View Column	Description
tp_rs_result	Single-table constraint or table pair result. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.
lv_id	ID of the lookup view.

rs_sv_id_view

The rs_sv_id_view view contains information about single-table constraints and table pairs that are based on a SQL view. The view contains the run-time and test result information for each version of a single-table constraint or table pair.

The view might contain duplicate IDs. You can use SELECT DISTINCT to avoid duplicate IDs.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_is_latest	Indicates whether this is the latest run of a given single-table constraint or table pair. Contains one of the following values: - 0. Not latest. - 1. Latest.
tr_start_time	Start time of the single-table constraint or table pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the single-table constraint or table pair test. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
ti_id	ID of the test installation. Do not use in reports.
tp_obj_id	ID of the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_user_id	User ID of the person who ran the test.
tp_rs_result	Single-table constraint or table pair result. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.
sv_id	ID of the SQL view.

rs_av_id_view

The rs_av_id_view view contains information about single-table constraints and table pairs that are based on an aggregate view. The view contains the run-time and test result information for each version of a single-table constraint or table pair.

The view might contain duplicate IDs. You can use SELECT DISTINCT to avoid duplicate IDs.

The following table describes the view columns:

View Column	Description
tr_id	Test run ID for all tests associated with the single-table constraint or table pair. Increments for each test run.
tr_is_latest	Indicates whether this is the latest run of a given single-table constraint or table pair. Contains one of the following values: - 0. Not latest. - 1. Latest.
tr_start_time	Start time of the single-table constraint or table pair test. Time is represented as the number of milliseconds since 1970 UTC.
tr_start_timestamp	Start time of the single-table constraint or table pair test. The time corresponds to the millisecond value in the tr_start_time column and appears in standard date time form.
ti_id	ID of the test installation. Do not use in reports.
tp_obj_id	ID of the single-table constraint or table pair.
tp_version	Version of the single-table constraint or table pair.
tp_user_id	User ID of the person who ran the test.
tp_rs_result	Single-table constraint or table pair result. Contains one of the following values: - -2. Error. - -1. No results. - 0. Fail. - 1. Pass.
av_id	ID of the aggregate view.

APPENDIX C

Metadata Import Syntax

This appendix includes the following topics:

Metadata Import Syntax Overview

The following sections display examples of metadata syntax definition.

Table Pair with One Test

```
<TablePair>
 Name = "CLIDETAIL_CLISTAGE"
 Description = "CLIDETAIL-CLISTAGE"
 ExternalID = "cliTest"
 SaveDetaliedBadRecords = true
 SaveBadRecordsTo = "SCHEMA"
 BadRecordsFileName = ""
 ParameterFile = "C:\\test_param.txt"
 <TableA>
 Name = "pc_repository/cli_demo/Sources/demo_connection/cliDetail"
 Connection = "dvo_demo_connection"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <TableB>
 Name = "pc_repository/cli_demo/Targets/cliStage"
 Connection = "dvo_demo_connection"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <Parameter>
```

```
 Name = "$$NewParameter1"
 Type = "string"
 Precision = "10"
 Scale = "0"
 <TestCase>
 TestType = "AGG"
 Aggregate = "SUM"
 ColumnA = "ProductAmount"
 ColumnB = "CustomerAmount"
 Operator = "="
 Comments = ""
 CaseInsensitive = false
 TrimRightWhitespace = false
 TreatNullsEqual = true
```

Table Pair with an SQL View as a Source

```
<TablePair>
 Name = "Joined_MYVIEW_FACTORDERS"
 Description = "Joined MYVIEW-FACTORDERS"
 ExternalID = ""
 <TableA>
 Name = "SQLView_470"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <TableB>
 Name = "pc_repository/dvo_demo/Targets/factOrders"
 Connection = "dvo_demo_connection"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <Join>
 ColumnA = "MyID"
 ColumnB = "LineID"
 ColumnA = "MyCurrency"
 ColumnB = "CurrencyName"
 <TestCase>
 TestType = "VALUE"
 ColumnA = "MyCurrency"
 ColumnB = "CurrencyName"
 Operator = "="
 Comments = ""
 CaseInsensitive = false
 TrimRightWhitespace = true
 TreatNullsEqual = true
 <TestCase>
 TestType = "AGG"
 Aggregate = "SUM"
 <ExpressionA>
 Expression = "if(MyID>10,10,MyID)"
 Datatype = "integer"
 Precision = 10
 Scale = 0
 ColumnB = "LineID"
 Operator = "="
 Comments = ""
 CaseInsensitive = false
 TrimRightWhitespace = false
 TreatNullsEqual = true
```

Table Pair with Two Flat Files

```
<TablePair>
 Name = "FLATFILE_FLATFILE"
 Description = "FLATFILE-FLATFILE"
 ExternalID = ""
 SaveDetaliedBadRecords = true
 SaveBadRecordsTo = "FLAT_FILE"
 BadRecordsFileName = "test.txt"
 <TableA>
 Name = "pc_repository/dvo_demo/Sources/FlatFile/FlatFile"
 SourceDirectory = "C:\\FlatFile\\Sourcess"
 SourceFilename = "flatfile.txt"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <TableB>
 Name = "pc_repository/dvo_demo/Sources/FlatFile/FlatFile"
 SourceDirectory = "C:\\FlatFiles\\Targets"
 SourceFilename = "flatfile2.txt"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <TestCase>
 TestType = "SET_ANotInB"
 ColumnA = "order_name"
 ColumnB = "order_name"
 Operator = "="
 Comments = ""
 CaseInsensitive = true
 TrimRightWhitespace = true
 TreatNullsEqual = true
```

Table Pair with XML File and Relational Table

```
<TablePair>
    Name = "Joined_DIMEMPLOYEES_DIMEMPLOYEES"
    Folder = "TablePairs"
    Description = "TablePair_DIMEMPLOYEESXML_RELATIONALDVODEMO"
    ExternalID = "TP_DIMEMPLOYEESXML_RELATIONALDVODEMO"
    CacheSize = "Auto"
    SaveDetaliedBadRecords = false
    EnableSampling = false
    <TableA>
        Name = "Repository_Service_91/DVO_DEMO/Sources/XML_File/DIMEMPLOYEES"
        SourceDirectory = "C:\\Informatica\\9.1.0\\clients\\PowerCenterClient\\client\
\bin"
        SourceFilename = "DIMEMPLOYEES.xml"
        SourceFileType = "Direct"
        XML Group Name = "X_Import"
        WhereClause = "SALARY>94999"
        WhereClauseDSQ = false
        AlreadySorted = false
        IsLarge = false
        InDB = false
    <TableB>
        Name = "Repository_Service_91/DVO_DEMO/Targets/DIMEMPLOYEES"
        Connection = "DVO_DEMO"
        WhereClause = "SALARY>94999"
        WhereClauseDSQ = false
        AlreadySorted = false
        IsLarge = true
        InDB = false
```

```
    <Join>
        ColumnA = "name"
        ColumnB = "FIRSTNAME"
```

Single-Table Constraint

```
<SingleTable>
 Name = "DIMEMPLOYEES"
 Description = "DIMEMPLOYEES"
 ExternalID = ""
 <TableA>
 Name = "pc_repository/dvo_demo/Targets/dimEmployees"
 Connection = "dvo_demo_connection"
 WhereClause = ""
 WhereClauseDSQ = false
 InDB = false
 <Key>
 ColumnA = "EmployeeID"
 <TestCase>
 TestType = "AGG"
 Aggregate = "COUNT"
 ColumnA = "EmployeeID"
 <ExpressionB>
 Expression = "100,200"
 Datatype = "integer"
 Precision = 10
 Scale = 0
 Operator = "Between"
 Comments = ""
 CaseInsensitive = false
 TrimRightWhitespace = false
 TreatNullsEqual = true
 <TestCase>
 TestType = "NOT_NULL"
 ColumnA = "LastName"
 ColumnB = ""
 Operator = "="
 Comments = ""
 CaseInsensitive = false
 TrimRightWhitespace = false
 TreatNullsEqual = true
```

SQL View

```
<SQLView>
 Name = "SQLView_991"
 Description = "MyView991"
 <Table>
 Name = "pc_repository/dvo_demo/Sources/demo_connection/srcOrders"
 Connection = "dvo_demo_connection"
 SQLQuery = "Select * from srcOrders"
 Comments = "This is a comment"
 <Columns>
 <Column>
 Name = "MyID"
 Datatype = "int"
 Precision = 10
 Scale = 0
 <Column>
```

```
 Name = "MyCurrency"
 Datatype = "varchar"
 Precision = 30
 Scale = 0
 <Column>
 Name = "MyAmount"
 Datatype = "decimal"
 Precision = 10
 Scale = 2
```

Lookup View

```
<LookupView>
 Name = "LookupView"
 Description = "Lookup srcOrders --> dimProducts"
 <SourceTable>
 Name = "pc_repository/dvo_demo/Sources/demo_connection/srcOrders"
 Connection = "dvo_demo_connection"
 <LookupTable>
 Name = "pc_repository/dvo_demo/Targets/dimProducts"
 Connection = "dvo_demo_connection"
 <Join>
 ColumnA = "ProductID"
 ColumnB = "ProductID"
 ColumnA = "ProductName"
 ColumnB = "ProductName"
```

APPENDIX D

Jasper Reports

This appendix includes the following topics:

Jasper Reports Overview

You can generate and view reports in the JasperReports Server.

You must use a third-party reporting tool to run PowerCenter reports. You can download and install your own version of the recommended JasperReports Server to run Data Validation reports.

For information about the PowerCenter Reports, see the *Informatica PowerCenter Using PowerCenter Reports Guide*. For information about the PowerCenter repository views, see the *Informatica PowerCenter Repository Guide*.

You can generate reports at the following levels:

- Table pairs and Single tables
- Folders
- Data Validation Option user
- PowerCenter repository
- Views

When you generate reports, you can also select multiple table pairs and single tables, or folders. If you have a large data test data set use the report annotations to view the report details.

Data Validation provides the following administrative reports:

- External IDs Used In Table Pairs
- Views Used in Table Pairs/Tables
- Sources Used In Table Pairs/Tables/Views

The following table lists all the Jasper reports and the corresponding levels of reporting:

Report Name	Report Level
Run Summary	User, Folder, Table Pair/Single Table
Table Pair Summary	User, Folder, Table Pair/Single Table
Detailed Test Results	User, Folder, Table Pair/Single Table
Table Pair Run Summary	Table Pair/Single Table
Last Run Summary	User, Folder, Table Pair/Single Table
Percentage of Bad Rows	User, Folder, Table Pair/Single Table
Percentage of Tests Passed	User, Folder, Table Pair/Single Table
Tests Run Vs Tests Passed	User, Folder, Table Pair/Single Table
Total Rows Vs Percentage of Bad Rows	User, Folder, Table Pair/Single Table
Bad Rows	Folder, Table Pair/Single Table
Most Recent Failed Runs	User, Folder
Failed Runs	Folder
Failed Tests	Folder
Validation Failures	User, Folder
External IDs Used In Table Pairs	User, Folder, Table Pair/Single Table
Views Used in Table Pairs/Tables	Views
Sources Used In Table Pairs/Tables/Views	PowerCenter Repository

You can export Jasper reports in the following formats:

- PDF
- DOC
- XLS
- XLSX
- CSV
- ODS
- ODT

Note: You cannot run Jasper Reporting Service in a Kerberos environment.

Status in Jasper Reports

Reports display the status of a test, table pair, or single table based on the report type.

The following status are available in Jasper reports for tests:

- Pass. The test has passed.
- Fail. The test has failed.
- Error. The test encountered a run error or the test has no result.

The following status are available in Jasper reports for table pairs and single tables:

- Pass. Pass status for a table pair or single table can occur in the following scenarios:
  - All the tests have passed.
  - If there is at least one test with pass status and rest of the tests with no results.
- Fail. If one of the tests in a table pair or single table fail, reports display the status as fail.
- Error. If all the tests in a table pair or single table has an error or no result, reports display the status as error.

Tests and table pairs or single tables with error status do not appear in the bar charts.

Generating a Report

You can generate a report at the table pair, folder, repository, or user level.

1. Select the object for which you want to generate a report. You can select multiple of objects of the same type.
2. Click **Action** > **Generate Report**.

   The **Report Parameters** dialog box appears.
   You can also right-click the object and select Generate Report if you want to generate a report at the table pair or folder level.
3. Select the report type.

   The following table displays the report options that you can select for each report:

Report Name	User	Recency	Run Date
Run Summary	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Table Pair Summary	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Detailed Test Results	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Table Pair Run Summary	-	-	Last 24 hours/Last 30 days/Custom date range
Last Run Summary	All Users/Current User/Any User	-	Last 24 hours/Last 30 days/Custom date range

Report Name	User	Recency	Run Date
Percentage of Bad Rows	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Percentage of Tests Passed	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Tests Run Vs Tests Passed	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Total Rows Vs Percentage of Bad Rows	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Bad Rows	-	All/Latest	Last 24 hours/Last 30 days/Custom date range
Most Recent Failed Runs	All Users/Current User/Any User	All/Latest	-
Failed Runs	-	-	Last 24 hours/Last 30 days/Custom date range
Failed Tests	-	All/Latest	Last 24 hours/Last 30 days/Custom date range
Validation Failures	All Users/Current User/Any User	All/Latest	Last 24 hours/Last 30 days/Custom date range
Table pairs/Tables with External ID	-	-	-

Note: You can select the user only if you generate the report from the user level.

4. Optionally, enter the report subtitle.

   You can use the report subtitle to identify a specific report.

5. To view details of bad records, select **Show Bad Records Detail**.
6. Click **Run.**

   The report appears in the browser.

Jasper Report Types

You can generate different Jasper reports in Data Validation Option.

You can generate the following Jasper reports in Data Validation Option:

Summary of Tests Run

Summary of Testing Activities report displays the number of table pairs or single tables, the number of tests for each table pair or single table, and the overall test results.

Table Pair/Table Summary

Lists each table pair or single table with the associated tests. Data Validation Option displays each table pair or single table on a separate page. The report includes a brief description of each test and result.

Detailed Test Result

Displays each test on a separate page with a detailed description of the test definition and results. If one of the test sources is a SQL, join, or a lookup view, the report also displays the view definition.

The following information is available in the Detailed Test Result report:

- Test details
- Table pair/single table details
- Run-time information
- Bad row details, including information about what causes the bad rows

Note: If the first row of the bad records is null, the Difference and % Difference columns do not appear.

Table Pair/Table Run Summary

Displays the summary of all the tests run for a table pair or single table for a given time period. You can click on the date in the report to view the Detailed Test Results report.

Last Run Summary

Displays the details of the last run of tests at the table pair or single table, folder, or user level. Reports lists the last test runs for the objects in the given time period.

If you generate the report at the user level, the report displays the folder summary. You can click on the folder to view the Last Run Summary report for that folder.

If you generate the report at the folder level or table pair/single table level, you can view the last run for all the table pairs or single tables for the given time period. You can click on a table pair or single table to view the Detailed Test Results report.

Percentage of Bad Rows

Displays the aggregate percentage of bad rows in comparison with all the rows processed over a period of time. You can generate the report at the table pair or single table, folder, or user level. You can click on a bar for a particular date on the graph to view the Bad Rows report.

The bars on the graph appear for the days on which you ran the tests.

Percentage of Tests Passed

Displays the percentage of passed tests in comparison with total number of tests over a period of time. You can generate the report at the table pair or single table, folder, or user level.

The bars on the graph appear for the days on which you ran the tests.

Tests Run Vs Tests Passed

Displays the total number of tests run over a period of time as a bar chart. The number of tests passed is plotted across the bar chart. You can generate the report at the table pair or single table, folder, or user level. You can click on the test passed points on the graph to view the Validation Failure by Day report.

The bars on the graph appear for the days on which you ran the tests.

Total Rows Vs Percentage of Bad Rows

Displays the total number of rows tested over a period of time as a bar chart. The percentage of bad rows is plotted across the bar chart. You can generate the report at the table pair or single table, folder, or user level. You can click on the percentage of bad rows points on the graph to view the Bad Rows report.

The bars on the graph appear for the days on which you ran the tests.

Bad Rows

Displays the number of bad rows across test runs over a period of time in the form of a bar chart. You can click on a bar to view the Table Pair/Table Run Summary report. You can generate the Bad Rows report at the folder level or the table pair/single table level.

Most Recent Failed Runs

Displays the top ten most recent failed runs. You can run the Most Recent Failed Runs report at the folder or user level.

If you run the report at the user level, the report displays the top ten failed runs across all the folders in the Data Validation Option repository in the given time period. If you click on a folder, you can view the Most Recent Failed Runs report for the folder. You can also click on the table pair or single table to view the Detailed Test Result report.

If you run the report at the folder level, the report displays the top ten failed runs for that particular folder in the given time period. You can click on the table pair or single table to view the Detailed Test Result report.

Note: Select the recency as latest if you want to get the most recent state of the table pair or single table.

Failed Runs

Displays the number of failed runs for each table pair or single table over a period of time in the form of a bar chart. Each bar represents the number of failed runs for a table pair or single table. You can click on a bar to view the Table Pair/Table Run Summary report. You can run the Failed Runs report at the folder level.

Failed Tests

Displays the number of failed tests across test runs over a period of time in the form of a bar chart. Each bar represents the number of failed tests for a table pair or single table. If you click the Table hyperlink, you can view the report in a tabular format. You can click on a bar or the table pair/single table name to view the Detailed Test Result report. You can run the Failed Tests report at the folder level.

Validation Failure by Folder

Displays the number of failed table pairs or single table as a bar chart for a given time period. Each bar represents the folder in which the failure occurred. If you click the Table hyperlink, you can view the report in a tabular format. You can click on a bar or the folder name to view the Failed Tests report available as bar chart and tabular format. You can run the Validation Failure by Folder report at the user or folder level.

External IDs Used In Table Pairs

Displays the list of table pairs or single tables with external IDs across all users. You can run the External IDs Used In Table Pairs report at the user, folder, or table pair level.

Sources Used In Table Pairs/Tables/Views

Displays the list of table pairs, single tables, and views in which a data source is used. Right-click on the PowerCenter repository, a repository folder, or a data source in the repository and select Get Source Usage In Table Pairs/Tables/Views to generate this report. You cannot generate a report at the repository folder level or repository level if there are more than 100 sources.

Views Used In Table Pairs/Tables/Views

Displays the list of table pairs and single tables in which a view is used. Right-click on the view and select Get View Usage In Table Pairs/Tables/Views to generate this report.

Note: Data Validation Option reports might display the details of table pairs or single tables that you previously deleted from the Data Validation Client. If you generate a report after you modify the description of a table pair or a single table, the reports might display two entries for the object with the object ID appended to the new and old description with same object ID or in the annotations of the bar chart.

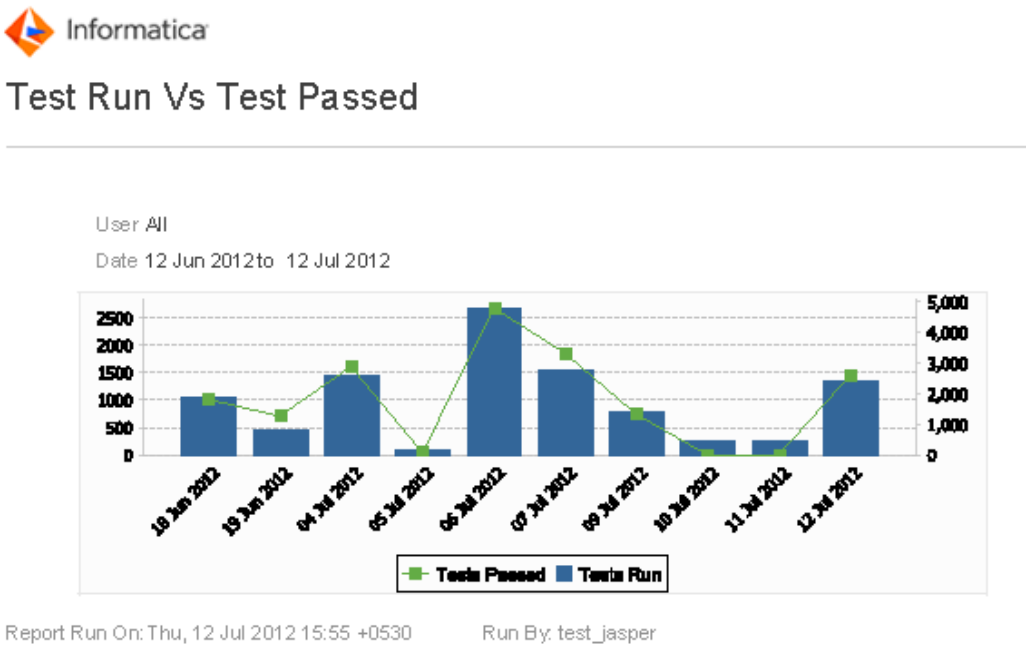
Jasper Report Examples

You can view the following reports:

- Tests Run Vs Tests Passed
- Total Rows Vs Percentage of Bad Rows
- Most Recent Failed Runs
- Last Run Summary

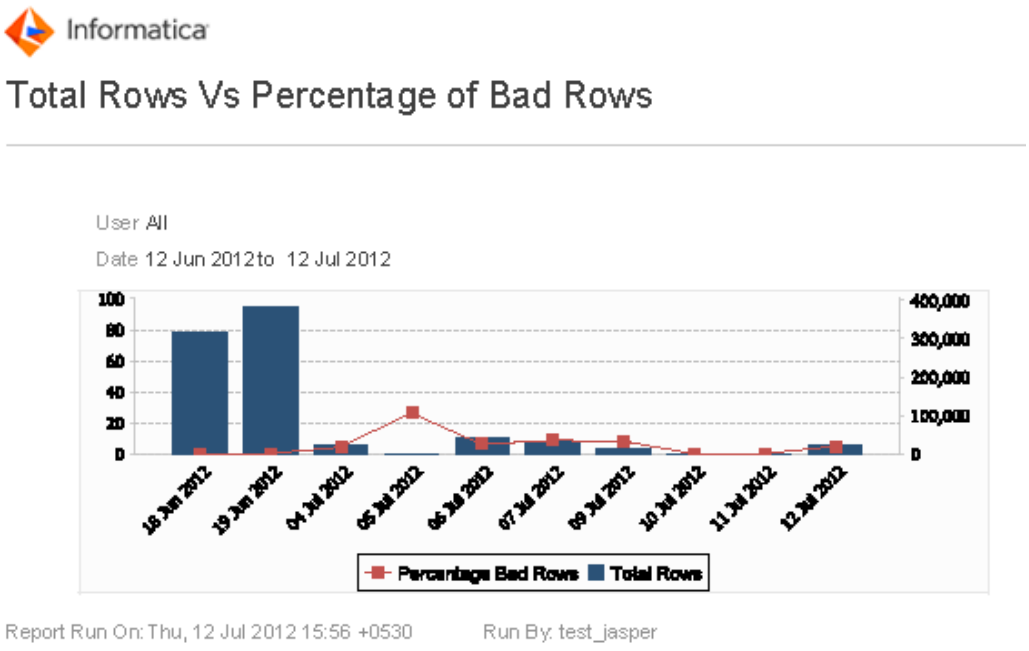
Tests Run Vs Tests Passed

The following figure shows an example of a Tests Run Vs Tests Passed report:



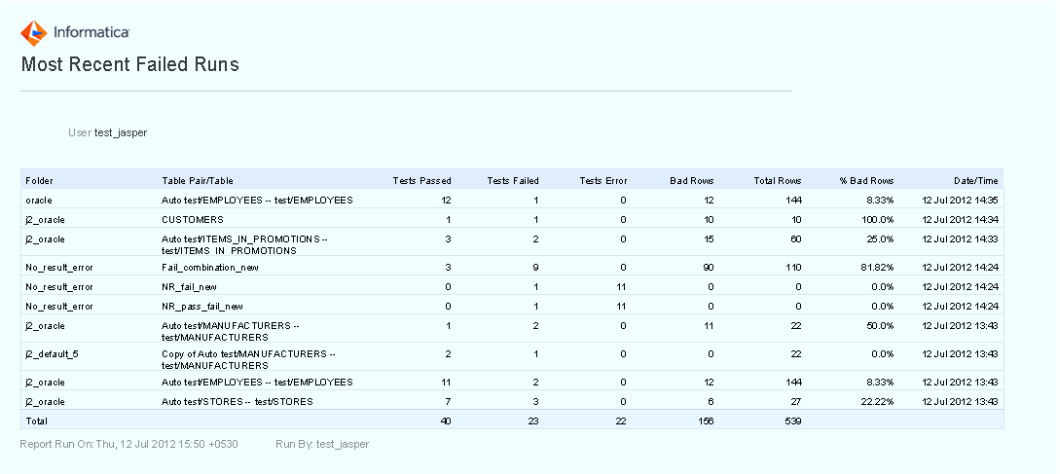
Total Rows Vs Percentage of Bad Rows

The following figure shows an example of a Total Rows Vs Percentage of Bad Rows report:



Most Recent Failed Runs

The following figure shows an example of a Most Recent Failed Runs report:



Informatica

Most Recent Failed Runs

User test_jasper

Folder	Table Pair/Table	Tests Passed	Tests Failed	Tests Error	Bad Rows	Total Rows	% Bad Rows	Date/Time
oracle	Auto test/EMPLOYEES -- test/EMPLOYEES	12	1	0	12	144	8.33%	12 Jul 2012 14:35
j2_oracle	CUSTOMERS	1	1	0	10	10	100.0%	12 Jul 2012 14:34
j2_oracle	Auto test/ITEMS_IN_PROMOTIONS -- test/ITEMS_IN_PROMOTIONS	3	2	0	15	60	25.0%	12 Jul 2012 14:33
No_result_error	Fail_combination_new	3	9	0	90	110	81.82%	12 Jul 2012 14:24
No_result_error	NR_fail_new	0	1	11	0	0	0.0%	12 Jul 2012 14:24
No_result_error	NR_pass_fail_new	0	1	11	0	0	0.0%	12 Jul 2012 14:24
j2_oracle	Auto test/MANUFACTURERS -- test/MANUFACTURERS	1	2	0	11	22	50.0%	12 Jul 2012 13:43
j2_default_5	Copy of Auto test/MANUFACTURERS -- test/MANUFACTURERS	2	1	0	0	22	0.0%	12 Jul 2012 13:43
j2_oracle	Auto test/EMPLOYEES -- test/EMPLOYEES	11	2	0	12	144	8.33%	12 Jul 2012 13:43
j2_oracle	Auto test/STORES -- test/STORES	7	3	0	6	27	22.22%	12 Jul 2012 13:43
Total		40	23	22	156	539		

Report Run On: Thu, 12 Jul 2012 15:50 +0530 Run By: test_jasper

Last Run Summary

The following figure shows an example of a Last Run Summary report:

Informatica

Last Run Summary

User test_jasper

Date 11 Jul 2012 to 12 Jul 2012

Table Pair/Table	Last Run	Tests	Passed	Failed	Error
Auto test\CUSTOMERS -- test\CUSTOMERS	12 Jul 2012, 13:40	12	10	2	0
Auto test\ORDER_ITEMS -- test/ORDER_ITEMS	12 Jul 2012, 13:40	5	3	2	0
Auto test\DISTRIBUTORS -- test/DISTRIBUTORS	12 Jul 2012, 13:41	6	4	2	0
Auto test\ITEMS -- test/ITEMS	12 Jul 2012, 13:42	9	7	2	0
Auto test\JOBS -- test\JOBS	12 Jul 2012, 13:42	3	1	2	0
Copy of Auto test/STORES -- test/STORES	12 Jul 2012, 13:42	10	0	0	10
Auto test\ORDERS -- test\ORDERS	12 Jul 2012, 13:42	9	7	2	0
Auto test\PROMOTIONS -- test/PROMOTIONS	12 Jul 2012, 13:43	6	4	2	0
Auto test\DEPARTMENT -- test\DEPARTMENT	12 Jul 2012, 13:43	3	1	2	0
Auto test\STORES -- test/STORES	12 Jul 2012, 13:43	10	7	3	0
Auto test\EMPLOYEES -- test/EMPLOYEES	12 Jul 2012, 13:43	13	11	2	0
Auto test\MANUFACTURERS -- test/MANUFACTURERS	12 Jul 2012, 13:44	3	1	2	0
Auto test\ITEMS_IN_PROMOTIONS -- test/ITEMS_IN_PROMOTIONS	12 Jul 2012, 14:34	5	3	2	0
CUSTOMERS	12 Jul 2012, 14:34	2	1	1	0
Total		96	60	26	10

Report Run On: Thu, 12 Jul 2012 15:45 +0530 Run By: test_jasper

APPENDIX E

Glossary

aggregate tests

Tests that check for lack of referential integrity in the source, incorrect ETL logic in the WHERE clauses, and row rejection by the target system.

bad records

Records that fail a test. You can view the bad records on the **Reports** tab. Aggregate tests do not display bad records.

constraint value

A representation of a constant value against which you can compare the field values.

count test

A test that compares the number of values in each field of the tables in a table pair.

data validation

The process of testing and validating data in a repository.

Data Validation Option folder

The folder in the Data Validation Option repository that stores the test metadata.

Data Validation Option repository

The database that stores the test metadata and test results.

Data Validation Option user

A user who creates and run tests in Data Validation Option. Data Validation Option stores the settings for each user in a unique user configuration directory.

DVOCmd

The command line program for Data Validation Option that you can use to do perform data validation tasks.

format test

A test that checks if the datatype of the fields in the source and target tables match.

inner join

A join that returns all rows from multiple tables where the join condition is met.

join view

A join view is a virtual table that contains columns from related heterogeneous data sources joined by key columns. You can use a join view in a table pair or single table.

lookup view

A view that looks up a primary key value in a lookup table or reference table with a text value from a source. Lookup view stores the primary key in the target fact table. Lookup view allows you to test the validity of the lookup logic in your transformations.

outer join

A join that returns all rows from one table and those rows from a secondary table where the joined fields are equal.

single table

Data Validation Option object that references a database table or flat file in a PowerCenter repository. Use a single table to create tests that require data validation on a table.

SQL view

A set of fields created from several tables and several calculations in a query. You can use an SQL View as a table in a single table or table pair.

table comparison

A table comparison compares tables in two different folders. Run a table comparison to generate table pairs and tests for tables in the folders.

table pair

A pair of sources on which you can create and run data validation tests.

threshold

Numeric value that defines an acceptable margin of error for a test. You can enter a threshold for aggregate tests and for value tests with numeric datatypes. If the margin crosses the threshold, the record Data Validation Option marks the record as a bad record.

unique test

A test to check if the value in a field is unique.

value test

A test to compare the values for fields in each row of the tables in a table pair that determines if the values match.

INDEX

F

I

J

L

M

N

O

P

R

S

T

U

V

W