Informatica® PowerExchange for PostgreSQL

10.5

User Guide for PowerCenter

Informatica PowerExchange for PostgreSQL User Guide for PowerCenter
10.5
March 2021

Table of Contents

Preface

Use the *Informatica® PowerExchange® for PostgreSQL User Guide* to learn how to read from and write to PostgreSQL by using the PowerCenter Client. Learn to create a PostgreSQL connection, develop mappings, and run sessions in an Informatica domain.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.

- View product availability information.

- Create and 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 find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.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

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

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

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica 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 and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find 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 extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction to PowerExchange for PostgreSQL

This chapter includes the following topic:

PowerExchange for PostgreSQL Overview

You can use PowerExchange for PostgreSQL to securely read data from or write data to PostgreSQL.

You can import PostgreSQL objects as sources and targets to create mappings, and run a session to extract data from or load data to PostgreSQL. When you use PostgreSQL objects in mappings, you must configure properties specific to PostgreSQL.

When you run a PostgreSQL session, the PowerCenter Integration Service writes data to PostgreSQL based on the defined data logic and the configured PostgreSQL connection.

PowerExchange for PostgreSQL Configuration

This chapter includes the following topics:

Installation and Configuration Overview

PowerExchange for PostgreSQL installs with PowerCenter. After you install or upgrade Informatica Services, you must register the PowerExchange for PostgreSQL plug-in with the PowerCenter repository.

Prerequisites

Before you use PowerExchange for PostgreSQL, perform the following tasks:

- Install or upgrade PowerCenter.
- Verify that you have read and write permissions on the following directories on each machine that runs the PowerCenter Integration Service and PowerCenter Repository Service:

  - `<Informatica installation directory>\server\connectors\cci\plugins`

  - `<Informatica installation directory>\server\bin\Plugin`

  - `<Informatica installation directory>\connectors\thirdparty`

- Verify that you have read and write permissions on the following directories on each machine where you installed the PowerCenter Client:

  - `<Informatica installation directory>\clients\PowerCenterClient`

  - `<Informatica installation directory>\clients\PowerCenterClient\bin\Help\<language>`

  For more information about product requirements and supported platforms, see the Product Availability Matrix on Informatica Network:
  https://network.informatica.com/community/informatica-network/product-availability-matrices.

Registering the Plug-in

After you complete the installation, register the PowerExchange for PostgreSQL plug-in with the repository.

To register the plug-in, the repository must be running in exclusive mode. Use the Administrator tool or the pmrep RegisterPlugin command line program to register the plug-in. If you do not have the correct privileges to register the plug-in, contact the user who manages the PowerCenter Repository Service.

The plug-in file is an `.xml` file that defines the functionality of the adapter. When you install the server component, the installer copies the plug-in file to the following directory:

```
<Informatica installation directory>/server/bin/Plugin
```

The name of the plug-in file for PowerExchange for PostgreSQL is `postgres_Plugin.xml`.

Registering the Plug-in from the Administrator Tool

Register a repository plug-in to add its functionality to the repository.

1. Run the PowerCenter Repository Service in exclusive mode.
2. In the **Navigator**, select the PowerCenter Repository Service to which you want to add the plug-in.
3. In the **Contents** panel, click the **Plug-ins** view.
4. In the **Actions** menu of the **Domain** tab, select **Register Plug-in**.
5. On the **Register Plugin** page, click the **Browse** button to locate the plug-in file.
6. Enter your user name and password.
7. Click **OK**.

    The PowerCenter Repository Service registers the plug-in with the repository. The results of the registration operation appear in the activity log.
8. Run the PowerCenter Repository Service in normal mode.

Registering the Plug-in from the Command Line Interface

You can use the `pmrep RegisterPlugin` command to register the plug-in from the command line interface.

1. Run the PowerCenter Repository Service in exclusive mode.
2. Run the `pmrep Connect` command to connect to the Repository Service with a user account that has the Administrator Repository privilege.

    The RegisterPlugin command uses the following syntax:

    ```
    pmrep connect -r <repository name> -d <domain_name> -n <domain user name> -x
    <domain_password>
    ```
3. Find `<adaptername>.xml` in the following directory:

    ```
    <Informatica installation directory>\server\bin\Plugin
    ```
4. Run the `pmrep RegisterPlugin` command to update the repository.

    The RegisterPlugin command uses the following syntax:

    ```
    pmrep registerplugin -i <Informatica installation directory\server\bin\Plugin
    \postgres_Plugin.xml -e
    ```

CHAPTER 3

PostgreSQL Sources and Targets

This chapter includes the following topics:

PostgreSQL Sources and Targets Overview

You can create a mapping with a PostgreSQL source to extract data from PostgreSQL. You can create a mapping with any source and a PostgreSQL target to load data to PostgreSQL. You can use tables as PostgreSQL sources.

When the PowerCenter Integration Service extracts data from the source or loads data to the target, it converts the data based on the data types associated with the source or the target.

Import PostgreSQL Source and Target Definitions

Use the **Create PowerExchange For PostgreSQL Source** or **Create PowerExchange For PostgreSQL Target** wizard to import PostgreSQL source and target definitions into the PowerCenter repository.

You must import PostgreSQL source and target objects before you create a mapping.

1. Start PowerCenter Designer and connect to a PowerCenter repository configured with a PostgreSQL instance.
2. Open a source or target folder.

3. Select **Source Analyzer** or **Target Designer**.

-   In the Source Analyzer, click **Sources** > **Create PowerExchange For PostgreSQL Source**.
    The following image shows the option to create a PostgreSQL source:
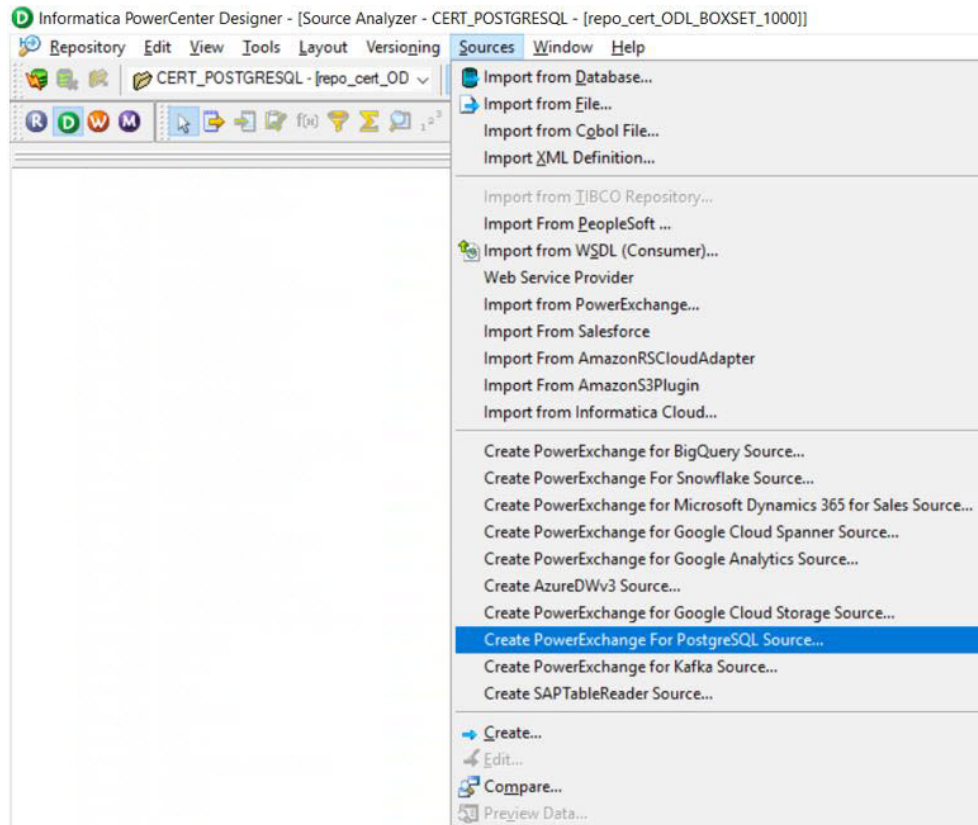
The **PostgreSQL Connection Details** dialog box appears.

- In the Target Analyzer, click **Targets** > **Create PowerExchange For PostgreSQL Target**.
The following image shows the option to create a PostgreSQL target:
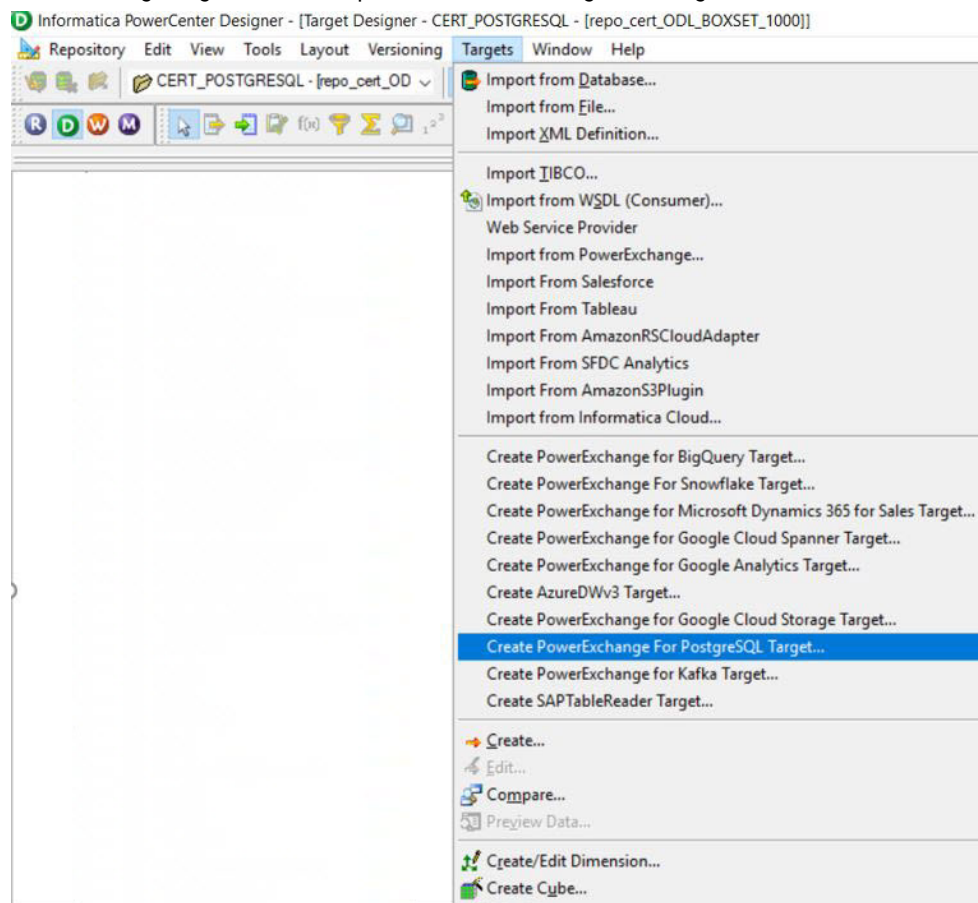
The **PostgreSQL Connection Details** dialog box appears.

4. Configure the PostgreSQL connection parameters.

5. Click **Test** to test the connection.

6. Click **Finish** to add the connection.

7. Click **Next**.

   The **Select Objects from** tab appears.

8. Select the database and schema in **Package Explorer**.

The following image shows the list of tables that appear:
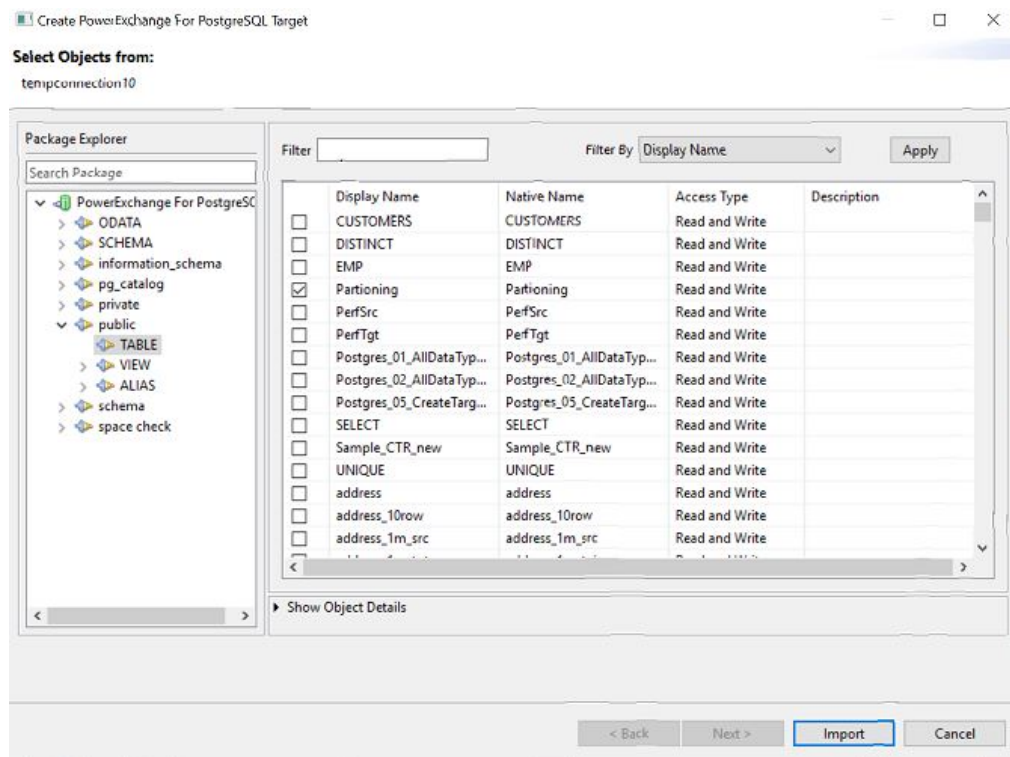
9. Select the table that you want to import, and then click **Import**.

To view the table metadata, select the table, and double-click the table name.

CHAPTER 4

PostgreSQL Mappings

This chapter includes the following topics:

PostgreSQL Mappings Overview

After you import a PostgreSQL source or target definition into the PowerCenter repository, you can create a mapping to extract data from a PostgreSQL source or load data to a PostgreSQL target.

You can extract data from one or more PostgreSQL sources, and load data to one or more PostgreSQL targets. You can join multiple PostgreSQL tables when you specify a join condition.

You can enter a filter condition to reduce the number of source rows the PowerCenter Integration Service returns from PostgreSQL sources. You can enter a single filter condition or a series of conditions.

Source Filter

You can enter a filter condition to reduce the number of source rows the PowerCenter Integration Service returns from PostgreSQL sources. You can enter a single filter condition or a series of conditions.

Use the source filter in the **Application Source Qualifier** to retrieve rows from an entity that meet a condition.

You can provide a source filter to improve the performance when you read from PostgreSQL.

Configuring a Source Filter

Configure a source filter from the **Application Source Qualifier**.

1. In the **Mapping Designer**, double-click the **Application Source Qualifier**.

    The **Edit Transformation** tab appears as shown in the following image:
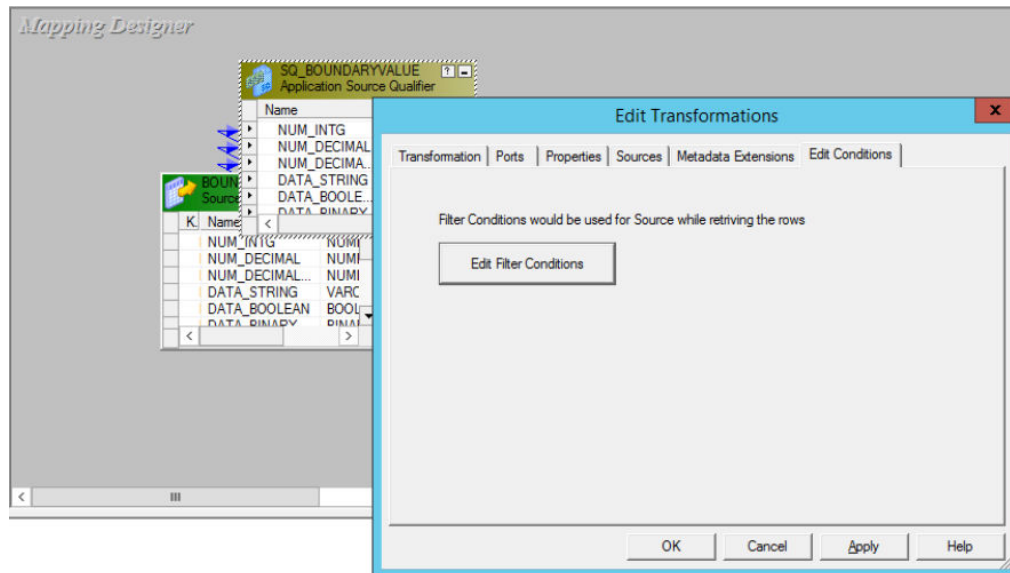
2. Click the **Edit Conditions** tab.

3. Click **Edit Filter Conditions**.

    The **Add Filter Condition** dialog box appears.

4. Select the filter field and operator that you want to specify in the condition, enter a value for the condition, and click **Add Condition**.

    The condition appears in the Filter Expression pane as shown in the following image:
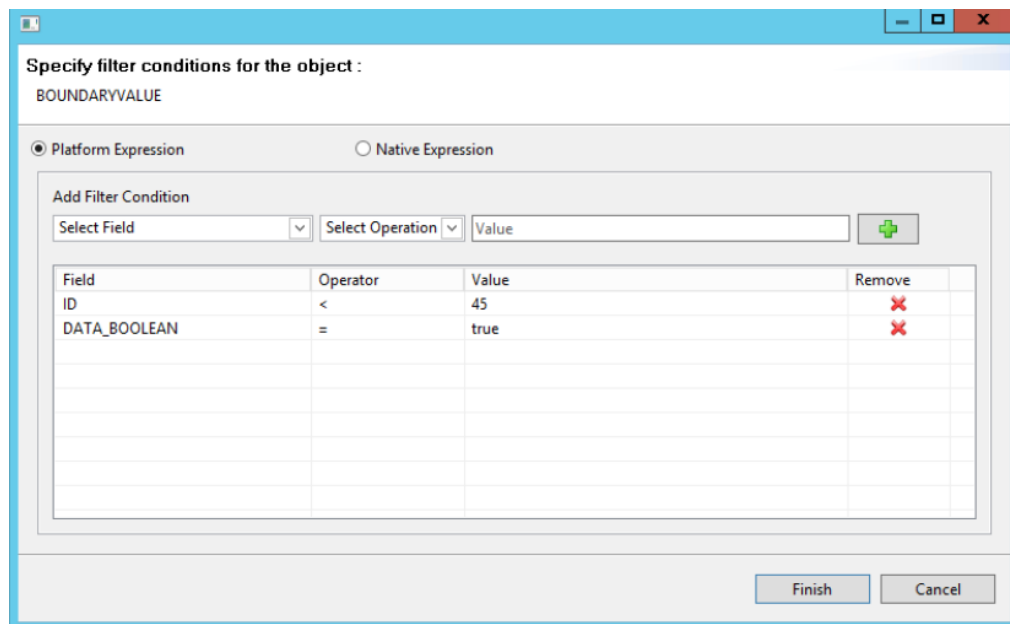
5. Click **Finish** to add the filter condition.

6. Click **OK**.

Source Join

You can enter a join condition to join multiple PostgreSQL source tables.

Configure the source join from the Application Source Qualifier of the parent table. When you configure the join in the Mapping Designer, retain the source qualifier only of the parent table and then link the fields from the child tables to the parent source qualifier.

Use the Platform Expression or Native Expression type to define the relationship of the tables that you want to join.

Configuring a Source Join

You can use the Application Source Qualifier of the parent table to join multiple tables.

1. In the **Mapping Designer**, retain the source qualifier only of the parent table and manually delete the source qualifiers for the child tables.
2. Link the fields from the child tables to the parent source qualifier.

   The following image shows an example mapping that contains the linked child tables with the parent source qualifier:

3. Double-click the **Application Source Qualifier** of the parent table.

   The **Edit Transformation** tab appears.
   The following image shows the **Edit Conditions** tab in the Application Source Qualifier Transformation type:
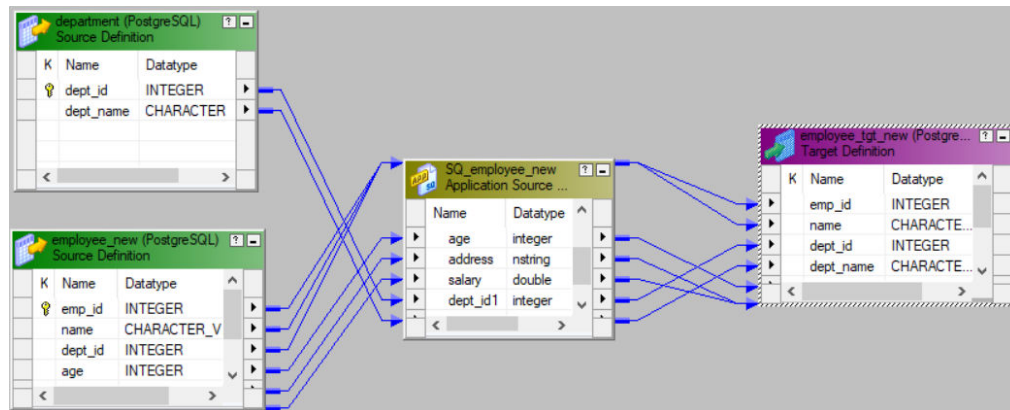
4. Click the **Edit Conditions** tab.
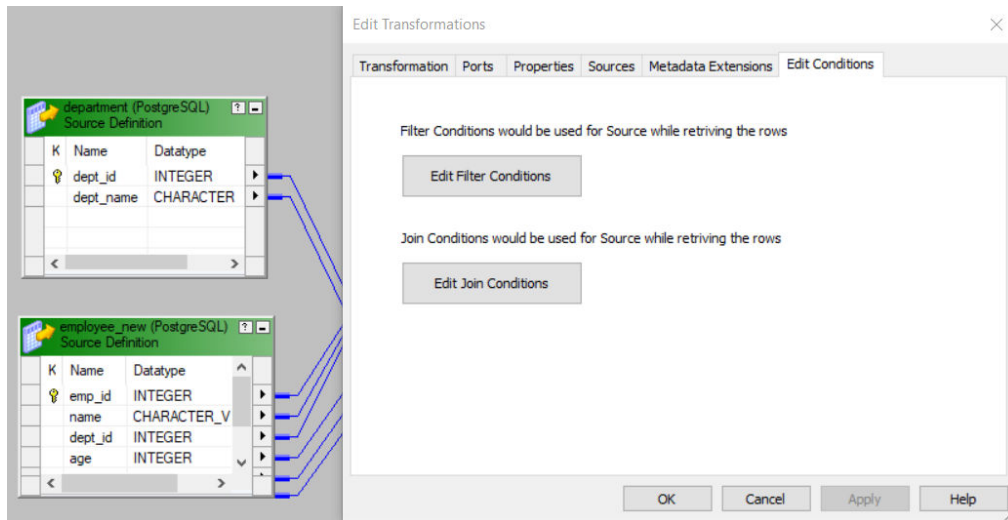
5. Click **Edit Join Conditions**.
6. To define a join condition for the tables using **Platform Expression**, perform the following tasks:
   a. In the **Tables** section, select the child table, and then select the join type for the child table with the parent table.

b. In the **Relationships** section, define the relationship for the join.

   The **Add** button in the **Relationships** section enables only when you select the child table.

   The following image shows the join types that you can configure and the relationship that you can define to join tables when you use the Infa expression:
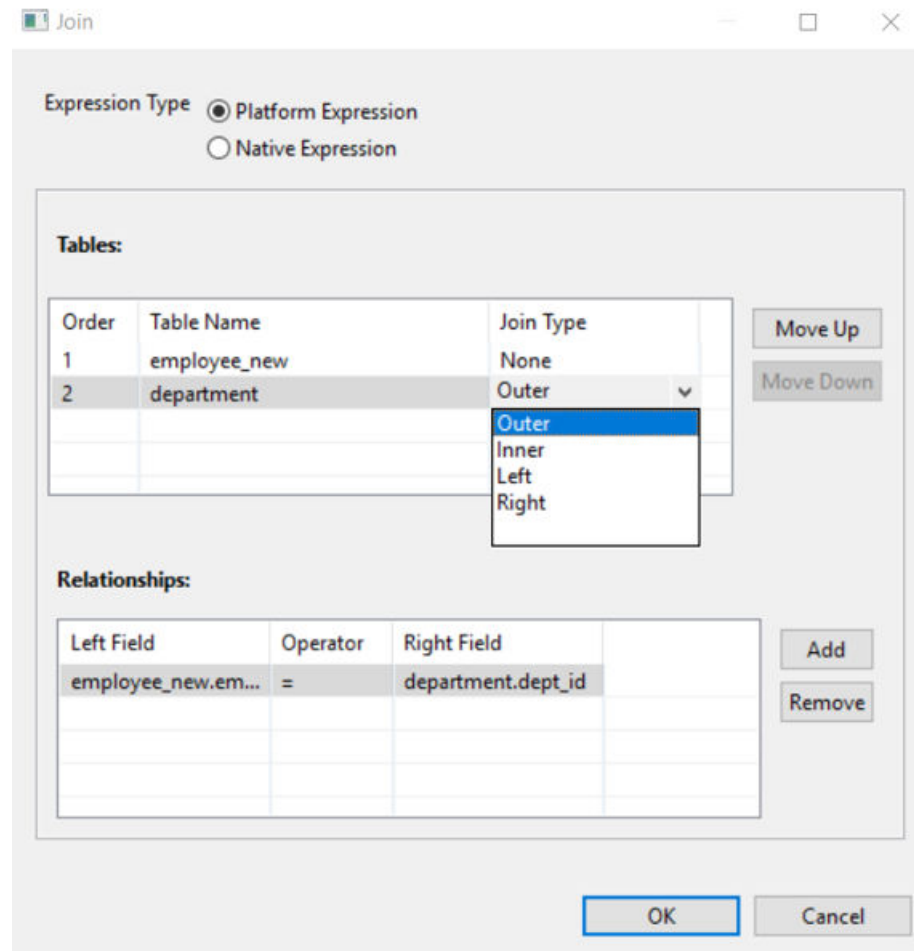
7. To define a join condition using **Native Expression**, select the fields and define a join query syntax:

The following image shows the fields that you add when you define the join query:
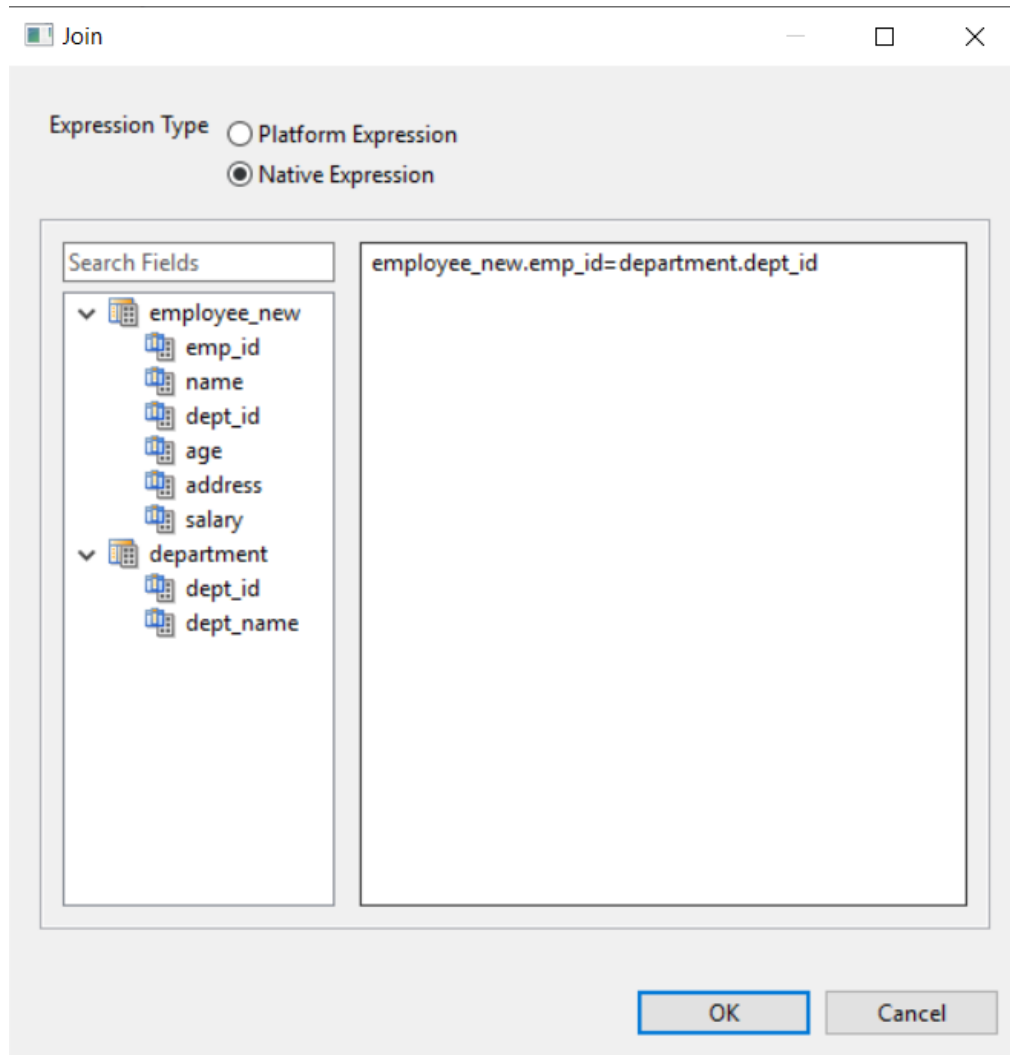
8. Click **OK**.

PostgreSQL Mapping Example

You work for an organization that stores purchase order details, such as customer ID, item codes, and item quantity in a Microsoft SQL Server. You need to analyze purchase order details and move data from the Microsoft SQL Server database to PostgreSQL.

Create a mapping to read all the purchase records from Microsoft SQL Server, process the data, and write them to a PostgreSQL target.

Perform the following tasks to move data from Microsoft SQL Server to PostgreSQL:

1. Import the Microsoft SQL Server source.
2. Import a PostgreSQL target.
3. Create a mapping with a Microsoft SQL Server source and a PostgreSQL target.

The following image shows the example mapping in the Mapping Designer:
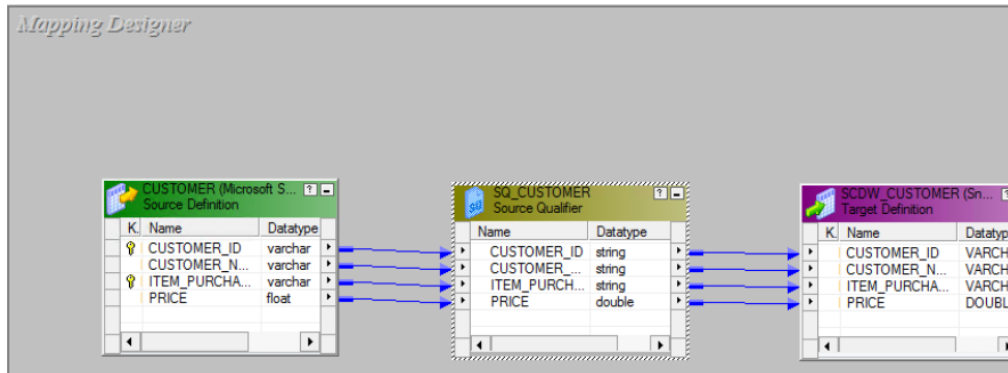
4. Create a session and configure the session properties to load the data to the PostgreSQL target.

The mapping contains the following objects:

Source Definition

The mapping source definition is a relational database object. In the **Source Analyzer**, import the Microsoft SQL Server source. The PowerCenter Integration Service reads the customer transaction details from the Microsoft SQL Server source.

The following table describes the structure of the source definition called CUSTOMER:

Field	Data Type
CUSTOMER_ID	Varchar
CUSTOMER_NAME	Varchar
ITEM_PURCHASED	Varchar
PRICE	Float

Mapping Target

The mapping contains a PostgreSQL target definition.

In the **Target Designer**, import a PostgreSQL target definition.

The following table describes the structure of the target definition called SCDW_CUSTOMER:

Field	Data Type
CUSTOMER_ID	Varchar
CUSTOMER_NAME	Varchar
ITEM_PURCHASED	Varchar
PRICE	Double

CHAPTER 5

PostgreSQL Sessions

This chapter includes the following topics:

PostgreSQL Sessions Overview

After you create mappings, you can create a session to extract and load data.

You must configure a PostgreSQL connection in the **Workflow Manager** to extract data from or load data to a PostgreSQL table. You can define properties in a session to determine how the PowerCenter Integration Service must extract data from a PostgreSQL source or load data to a PostgreSQL target.

PowerExchange for PostgreSQL Connection Properties

When you configure a PowerExchange for PostgreSQL connection, you define the connection attributes that the PowerCenter Integration Service uses to connect to the PostgreSQL database.

The following table describes the PostgreSQL connection properties:

Connection Property	Description
Host Name	Host name of the PostgreSQL server to which you want to connect.
Port	Port number for the PostgreSQL server to which you want to connect. Default is 5432.
Database	The PostgresSQL database name.
User Name	User name to access the PostgreSQL database.
Password	Password for the PostgreSQL database user name.
Encryption Method	Determines whether the data exchanged between the PowerCenter Integration Service and the PostgreSQL database server is encrypted: Select one of the following encryption methods: - noEncryption. Establishes a connection without using SSL. Data is not encrypted. - SSL. Establishes a connection using SSL. Data is encrypted using SSL. If the PostgreSQL database server does not support SSL, the connection fails. - requestSSL. Attempts to establish a connection using SSL. If the PostgreSQL database server does not support SSL, the PowerCenter Integration Service establishes an unencrypted connection. Default is noEncryption.
Validate Server Certificate	Applicable if you enable the encryption method to SSL or requestSSL. Select the **Validate Server Certificate** option so that the PowerCenter Integration Service validates the server certificate that is sent by the PostgreSQL database server. If you specify the **Hostname In Certificate** parameter, the PowerCenter Integration Service also validates the host name in the certificate.
TrustStore	Applicable if you select SSL or requestSSL as the encryption method and the Validate Server Certificate option. The path and name of the truststore file, which contains the list of the Certificate Authorities (CAs) that the PostgreSQL client trusts.
TrustStore Password	Applicable if you select SSL or requestSSL as the encryption method and the Validate Server Certificate option. The password to access the truststore file that contains the SSL certificate.
Host Name In Certificate	Optional when you select SSL or requestSSL as the encryption method and the Validate Server Certificate option. Specifying a host name ensures additional security and the PowerCenter Integration Service validates the host name included in the connection with the host name in the SSL certificate.

Connection Property	Description
KeyStore	Applicable if you select SSL as the encryption method and when client authentication is enabled on the PostgreSQL database server. The path and the file name of the key store. The keystore file contains the certificates that the PostgreSQL client sends to the PostgreSQL server in response to the server's certificate request.
KeyStore Password	Applicable if you select SSL as the encryption method and when client authentication is enabled on the PostgreSQL database server. The password for the keystore file required for secure communication.
Key Password	Applicable if you select SSL as the encryption method and when client authentication is enabled on the PostgreSQL database server. Required when individual keys in the keystore file have a different password than the keystore file.
Additional Connection Properties	Additional connection parameters that you want to use. You must provide the connection parameters as semicolon-separated key-value pairs. For example, `ConnectionRetryCount=2; ConnectionRetryDelay=5`
Crypto Protocol Versions	Required if you enable the encryption method to SSL or requestSSL. Specifies a cryptographic protocol or a list of cryptographic protocols when you use an encrypted connection. You can select from the following protocols: - SSLv3 - TLSv1 - TLSv1_1 - TLSv1_2

Configuring a PostgreSQL Connection

Configure a PostgreSQL connection in the Workflow Manager to define the connection attributes that the PowerCenter Integration Service uses to connect to the PostgreSQL database.

1. In the Workflow Manager, click **Connections** > **Relational**.

    The **Relational Connection Browser** dialog box appears.

2. Click **New**.

    The **Relational Connection Editor** dialog box appears.

3. Enter a name for the PostgreSQL connection.

4. Enter the PostgreSQL connection attributes.

5. Click **OK** to create a PostgreSQL connection.

Configure PostgreSQL Source Session Properties

You can configure the session properties for a PostgreSQL source on the **Workflow Manager** tab. Define the properties for the source instance in the session.

The following table describes the session properties that you can configure for a PostgreSQL source session:

Session Property	Description
Pre-SQL	The pre-SQL commands to run before the PowerCenter Integration Service reads data from the source.
Post-SQL	The post-SQL commands to run after the PowerCenter Integration Service reads data from source.
SQL Override	The SQL statement to override the default query generated from the specified source type to read data from the PostgreSQL source. Ensure that the list of selected columns, data types, and the order of the columns that appear in the query matches the columns, data types, and order in which they appear in the source object. Note: SQL override is not supported when you enable partitioning. If you specify an SQL override and configure partitioning, the mapping fails.
Filter Override Type	The type of filter expression that you want to override in the source qualifier. You can select Native or Platform filter expression type to override. Default is None.
Fetch Size	Determines the number of rows to read in one resultant set from PostgreSQL. Specifying a number limits the number of rows to fetch with each trip to the database and avoids unnecessary memory consumption. Default is 100000
Filter Override	The filter condition that overrides the filter condition you specify in the source qualifier. After you select Native or Platform Filter Override Type, specify the filter condition to override in the source qualifier. **Note:** When you configure a Platform filter override, you must specify the absolute qualified column name in the following format: `tablename.columnname`

Configure a Filter Override

When you read data from a PostgreSQL source, you can specify the filter override type and filter override condition in the PostgreSQL source session properties to override the filter condition you specify in the source qualifier.

You must select the corresponding Native or Platform expression as the **Filter Override Type** in the PostgreSQL source session properties based on the expression used in the source qualifier. Default is None.

After you select the filter expression, specify the filter condition in the **Filter Override** field. When you run the session, the PowerCenter overrides the filter condition you specify in the source qualifier and uses the condition you specified in the session properties to filter the PostgreSQL data.

The following image shows the filter override type and the filter override condition fields in the PostgreSQL source session properties:
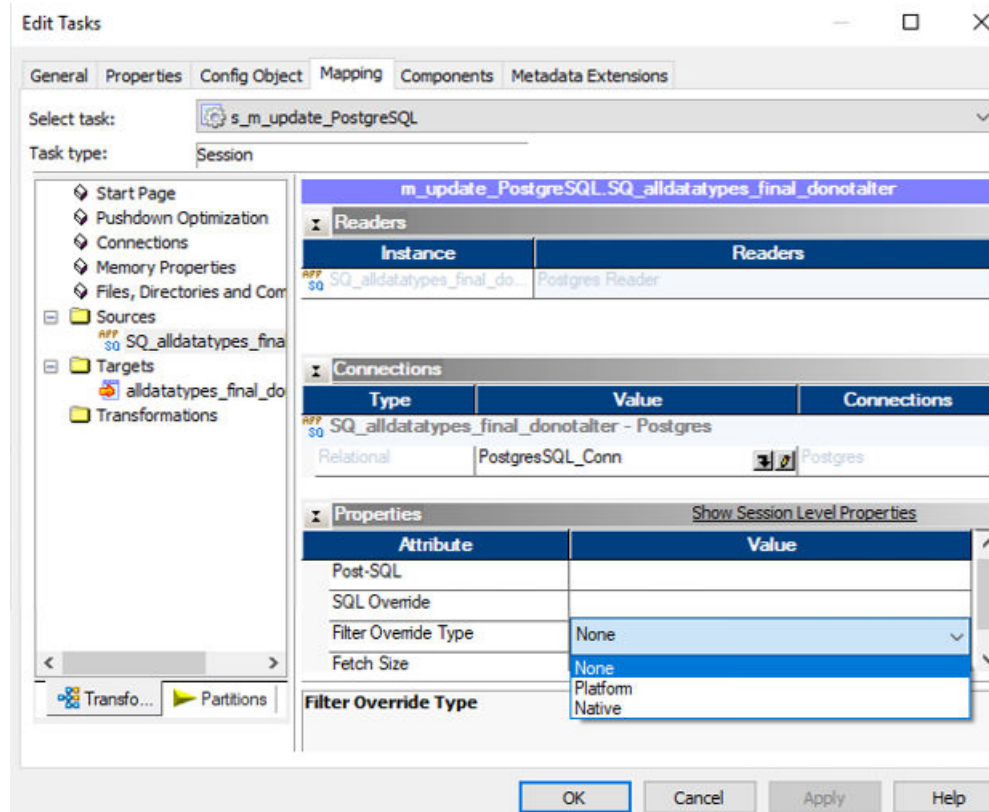
Configure PostgreSQL Target Session Properties

You can configure the session properties for a PostgreSQL target on the **Workflow Manager** tab. Define the properties for the target instance in the session.

The following table describes the session properties that you can configure for a PostgreSQL target session:

Session Property	Description
Update Mode	Specifies the mode to write data to PostgreSQL target. You can specify the following modes:- - Update As Update. Updates all rows flagged for update if the entries exist. - Update Else Insert. Updates all rows flagged for update if the entries exist in the target. If the entries do not exist, the PowerCenter Integration Service inserts the entries.
Pre-SQL	The pre-SQL commands to run before the PowerCenter Integration Service writes data to the target.
Post-SQL	The post-SQL commands to run after the PowerCenter Integration Service writes data to the target.
Truncate Target	The PowerCenter Integration Service truncates the target before writing the data.

Parameterization

You can parameterize the PostgreSQL connection and session properties.

You can create a configuration file with the connection and session properties that you want to override at run time.

Lookup

You can configure a cached connected Lookup transformation in a mapping. A connected Lookup transformation is connected to a PostgreSQL source or target in a mapping. When you create a cached lookup, the performance increases because the Integration Service caches a large lookup source or small lookup tables. When you cache the lookup source, the Integration Service queries the lookup cache instead of querying the lookup source for each input row.

When you run a mapping that contains a connected Lookup transformation, the Integration Service passes values from another transformation to input ports in the Lookup transformation. For each input row, the Integration Service queries the lookup source or cache based on the lookup ports and the lookup condition in the transformation. The Integration Service returns data from the query and passes it to the next transformation in the mapping.

Update Strategy Transformation

You can set the update strategy at the mapping or the session level:

- Within a session. When you configure a session, you can instruct the Integration Service to either treat all rows in the same way. You can flag all rows for insert, delete, or update, or you can select the data driven option, where the Integration Service follows instructions coded into Update Strategy transformations within the session mapping.

- Within a mapping. Within a mapping, you use the Update Strategy transformation to flag all rows for insert, delete, update, or reject. The Update Strategy transformation takes precedence over the update strategy you specify in the PostgreSQL session properties.

For more information, see the Update Strategy Transformation chapter in the *PowerCenter Transformation Guide*.

Partitioning

When you read from and write data to PostgreSQL, you can configure pass-through partitioning to optimize the session performance at run time.

When you specify pass-through partitioning for a PostgreSQL Source Qualifier transformation, you can specify filter conditions in the PostgreSQL session properties to override the filter condition you specify in the source qualifier. The Integration Service uses the filter condition you specify in the session properties when it filters data from the source.

To configure pass-through partitioning, select the Source Qualifier transformation, and add a partition point from the **Mapping** tab of the session properties. Add the number of partitions you require and select the partition type as pass through for each of the partitions.

Based on the number or partitions you add, the PowerCenter Integration Service adds those many number of partition fields for the **Filter Override** attribute in the session properties. Specify the filter override condition for each of the partitions. The PowerCenter Integration Service uses the filter conditions you specify to pass data through the appropriate partition.

Similarly, you can specify the fetch size for each of the partitions in the PostgreSQL source session properties.

Configuring Pass-Through Partitioning for a PostgreSQL Session

Configure pass-through partitioning for a session and the specify the filter override condition for each partition. When you configure a filter override in a pass-through partitioning, key range partitioning is enabled.

1. In the Workflow Manager, double-click the Session task to open the session properties, and click the **Mapping** tab.

2. In the **Partitions** tab, double-click the source qualifier and add the required number of partitions.

3. Select the **Partition Type** as **Pass Through**.
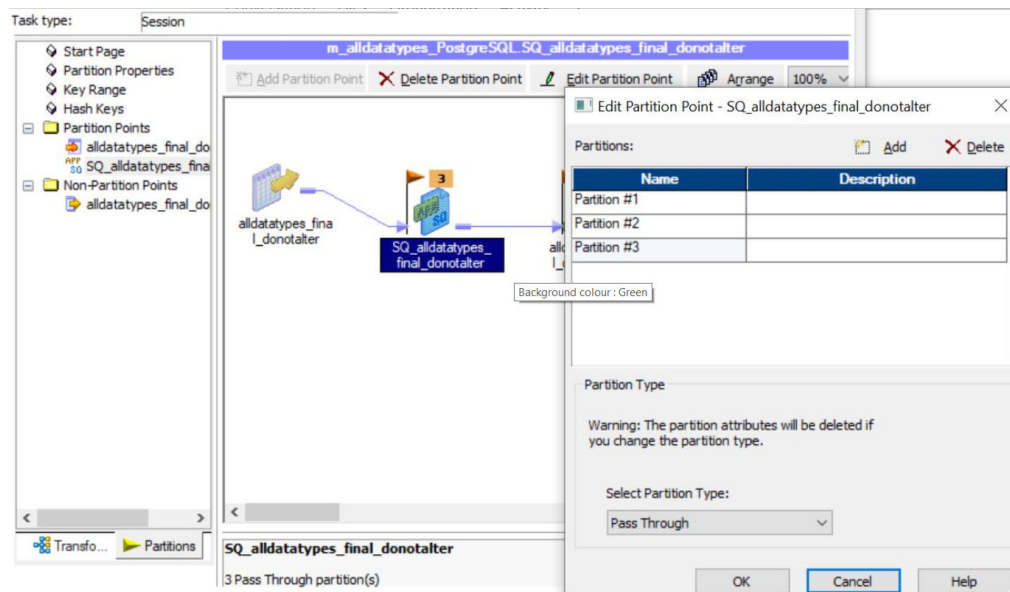
4. Click the **Transformations** tab, and navigate to the session properties.

5. In the Filter Override session property, specify the filter override condition for each partition.

Task type:	Session

m_alldatatypes_PostgreSQL.SQ_alldatatypes_final_donotalter

- ♦ Start Page
- ♦ Pushdown Optimization
- ♦ Connections
- ♦ Memory Properties
- ♦ Files, Directories and Com
- ⊟ ☐ Sources
  - SQ_alldatatypes_fina
- ⊟ ☐ Targets
  - alldatatypes_final_do
- ☐ Transformations

Readers

Instance	Readers
SQ_alldatatypes_final_donotalter	Postgres Reader

Connections

Type	Value	Connections
SQ_alldatatypes_final_donotalter - Postgres	Background colour : Green	
Relational	PostgresSQL_Conn	Postgres

Properties   Show Session Level Properties

Attribute	Value
⊟ Filter Override	
Partition #1	alldatatypes_final_donotalter.col_smallint<0
Partition #2	alldatatypes_final_donotalter.col_smallint=0
Partition #3	alldatatypes_final_donotalter.col_smallint>0
⊟ SQ_alldatatypes_final_donotalter - ERP Source Qualifier	

[Transfo... | ► Partitions]

6. Click **OK**.

CHAPTER 6

PostgreSQL Pushdown Optimization

This chapter includes the following topics:

PostgreSQL Pushdown Optimization Overview

You can use pushdown optimization to push transformation logic to source or target databases. Use pushdown optimization when you use database resources to improve mapping performance.

When you run a mapping configured for pushdown optimization, the mapping converts the transformation logic to an SQL query. The mapping sends the query to the database, and the database executes the query.

PostgreSQL supports full pushdown optimization for mappings.

Pushdown Optimization Functions, Operators, and Variables

The following table summarizes the availability of pushdown functions in an PostgreSQL database. Columns marked with an X indicate that the function can be pushed to the PostgreSQL database by using full pushdown optimization. Columns marked with a dash (-) symbol indicate that the function cannot be pushed to the database.

Function	PostgreSQL	Function	PostgreSQL	Function	PostgreSQL
ABORT()	-	INITCAP()	X	REG_MATCH()	-
ABS()	X	INSTR()	-	REG_REPLACE	-

Function	PostgreSQL	Function	PostgreSQL	Function	PostgreSQL
ADD_TO_DATE()	X	IS_DATE()	-	REPLACECHR()	-
AES_DECRYPT()	-	IS_NUMBER()	-	REPLACESTR()	-
AES_ENCRYPT()	-	IS_SPACES()	-	REVERSE()	-
ASCII()	X	ISNULL()	X	ROUND(DATE)	-
AVG()	X	LAST()	-	ROUND(NUMBER)	X
CEIL()	X	LAST_DAY()	X	RPAD()	X
CHOOSE()	-	LEAST()	-	RTRIM()	X
CHR()	X	LENGTH()	X	SET_DATE_PART()	-
CHRCODE()	-	LN()	X	SIGN()	X
COMPRESS()	-	LOG()	X	SIN()	X
CONCAT()	X	LOOKUP	X	SINH()	-
COS()	X	LOWER()	X	SOUNDEX()	-
COSH()	-	LPAD()	X	SQRT()	X
COUNT()	X	LTRIM()	X	STDDEV()	X
CRC32()	-	MAKE_DATE_TIME()	-	SUBSTR()	X
CUME()	-	MAX()	X	SUM()	X
DATE_COMPARE()	-	MD5()	-	SYSDATE()	X
DATE_DIFF()	X	MEDIAN()	-	SYSTIMESTAMP()	X
DECODE()	X	METAPHONE()	-	TAN()	X
DECODE_BASE64()	-	MIN()	X	TANH()	-
DECOMPRESS()	-	MOD()	X	TO_BIGINT	X
ENCODE_BASE64()	-	MOVINGAVG()	-	TO_CHAR(DATE)	X
EXP()	X	MOVINGSUM()	-	TO_CHAR(NUMBER)	X
FIRST()	-	NPER()	-	TO_DATE()	X
FLOOR()	X	PERCENTILE()	-	TO_DECIMAL()	X
FV()	-	PMT()	-	TO_FLOAT()	X
GET_DATE_PART()	-	POWER()	X	TO_INTEGER()	X

Function	PostgreSQL	Function	PostgreSQL	Function	PostgreSQL
GREATEST()	-	PV()	-	TRUNC(DATE)	-
IIF()	X	RAND()	-	TRUNC(NUMBER)	X
IN()	X	RATE()	-	UPPER()	X
INDEXOF()	-	REG_EXTRACT()	-	VARIANCE()	X

The following table lists the pushdown operators that can be used in an PostgreSQL database. Columns marked with an X indicate that the operator can be pushed to the PostgreSQL database by using source-side, target-side, or full pushdown optimization. Columns marked with an S indicate that the operator can be pushed to the PostgreSQL database by using source-side pushdown optimization.

Operator	PostgreSQL
+ - * /	X
%	X
||	X
= > < >= <= <>	X
!=	X
^=	X
not and or	X

The following table summarizes the availability of PowerCenter variables in relational databases. Columns marked with an X indicate that the variable can be pushed to the database by using source-side, target-side, or full pushdown optimization. Columns marked with a dash (-) symbol indicate that the variable cannot be pushed to the database.

Variable	PostgreSQL
SESSSTARTTIME	X
SYSDATE	X
WORKFLOWSTARTTIME	-

Configuring PostgreSQL ODBC Connections for Pushdown Optimization

You can set the pushdown optimization for the ODBC connection type that uses PostgreSQL ODBC drivers to enhances the mapping performance. To use an ODBC connection to connect to PostgreSQL, you must configure the ODBC connection.

After you configure the ODBC connection, select the value of the **Pushdown Optimization** property as **Full** accordingly in the session properties.

PostgreSQL supports ODBC PostgreSQL drivers on Windows and Linux systems. You must install the ODBC PostgreSQL 32-bit driver to import the source and target into the Designer. Use the 64-bit driver to run the session.

Configuring PostgreSQL ODBC Connection on Windows

Before you establish an ODBC connection to connect to PostgreSQL on Windows, you must configure the ODBC connection.

Perform the following steps to configure an ODBC connection on Windows:
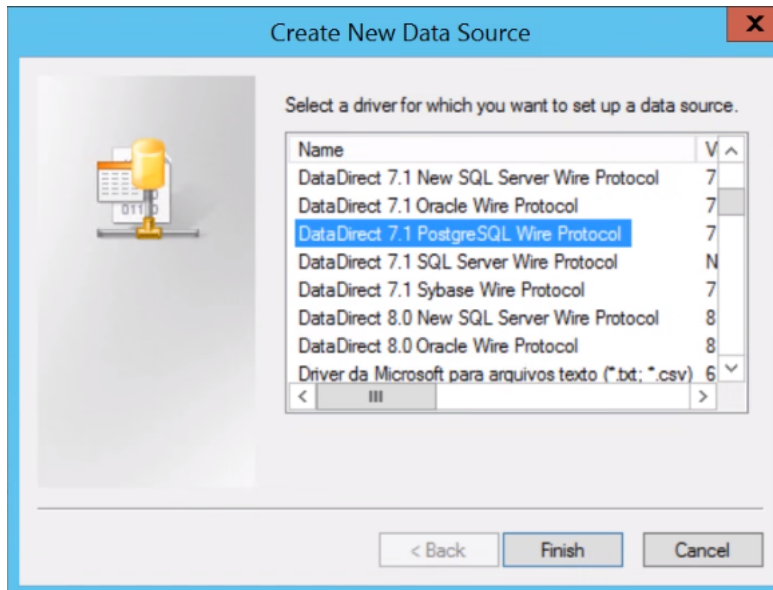
1. Run the `odbcad32.exe` file.

   The **ODBC Data Sources Administrator** dialog box appears.

2. Click **System DSN**.

   The following image shows the **System DSN** tab on the **ODBC Data Sources Administrator** dialog box:
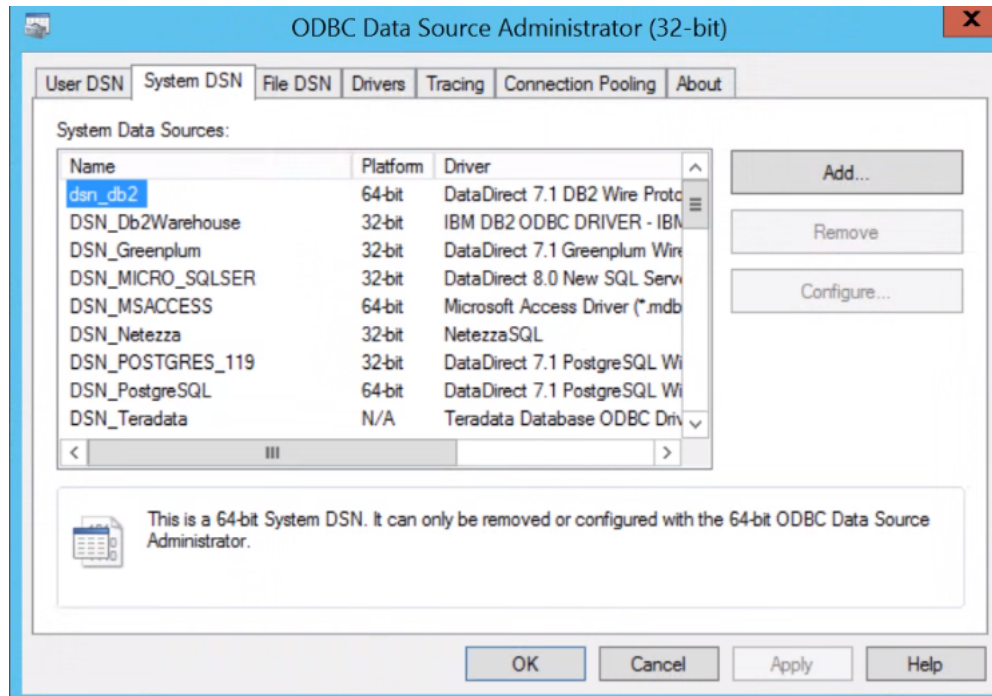

3. Click **Add**.

4. On the **Create New Data Source** dialog box, select the PostgreSQL driver.

5. On the **ODBC PostgreSQL Wire Protocol Driver Setup** dialog box, specify the following connection properties:

Property	Description
Data Source Name	Name of the data source.
Description	Description of the data source.
Host Name	Name of the PostgreSQL server.
Port Number	Port number of the PostgreSQL server.
Database Name	Name of the PostgreSQL database.

The following image shows the connection properties configured for the ODBC connection:
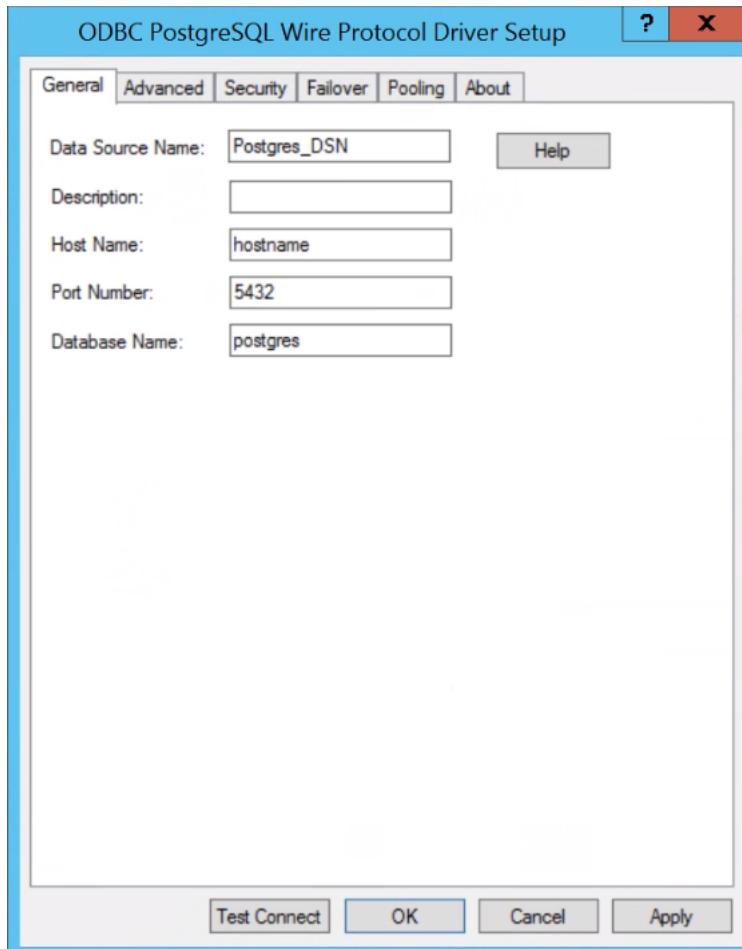
6. Click **Test Connect** to test the connection that you configured.

7. On the **Logon to PostgreSQL Wire Protocol** dialog box, specify the credentials of your PostgreSQL database.
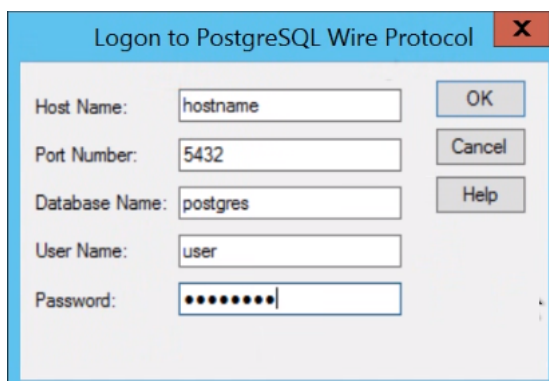
8. Click **OK**.

   The PostgreSQL ODBC connection is configured successfully on Windows.

After you configure the PostgreSQL ODBC connection, you must create an ODBC connection to connect to PostgreSQL.

Configuring PostgreSQL ODBC Connection on Linux

Before you establish an ODBC connection to connect to PostgreSQL on Linux, you must configure the ODBC connection.

Perform the following steps to configure an ODBC connection on Linux:

1. Download the PostgreSQL ODBC drivers from the PostgreSQL website.

   You must download the 32-bit or 64-bit driver based on your Linux system.

2. Install the PostgreSQL ODBC drivers on the machine that hosts the PowerCenter Integration Service.

3. Configure the `odbc.ini` file properties in the following format, for example:

   ```
   [PostgreSQL]
   Driver=/export/home/pc100_64/PCQA100_DOMAIN/Informatica/10.0.0/ODBC7.1/lib/
   DWpsql28.so
   Setup=/usr/lib64/libodbcpsqlS.so
   Servername=10.65.43.55
   Port=5432
   Database=postgres
   user=postgres
   password=root
   ```

4. Specify the following properties in the `odbc.ini` file:

Property	Description
ODBC Data Sources	Name of the data source.
Driver	Location of the PostgreSQL ODBC driver file.
Host	Location of the PostgreSQL host.
Port	Port number of the PostgreSQL server.
Database	Name of the PostgreSQL database.

   Note: You must specify the **Host**, **Port**, and **Database** values from the JDBC URL.

5. Add the `odbc.ini` file path in your source file in the following format:

   ```
   ODBCINI=/<odbc.ini file path>/odbc.ini
   ```

6. Restart the PowerCenter Server.

   The PostgreSQL ODBC connection on Linux is configured successfully.

After you configure the PostgreSQL ODBC connection, you must create an ODBC connection to connect to PostgreSQL.

Creating a PostgreSQL ODBC Connection

You must create an ODBC connection to connect to PostgreSQL after you configure the ODBC connection.

Perform the following steps to create an PostgreSQL ODBC connection on the **Connections** page:
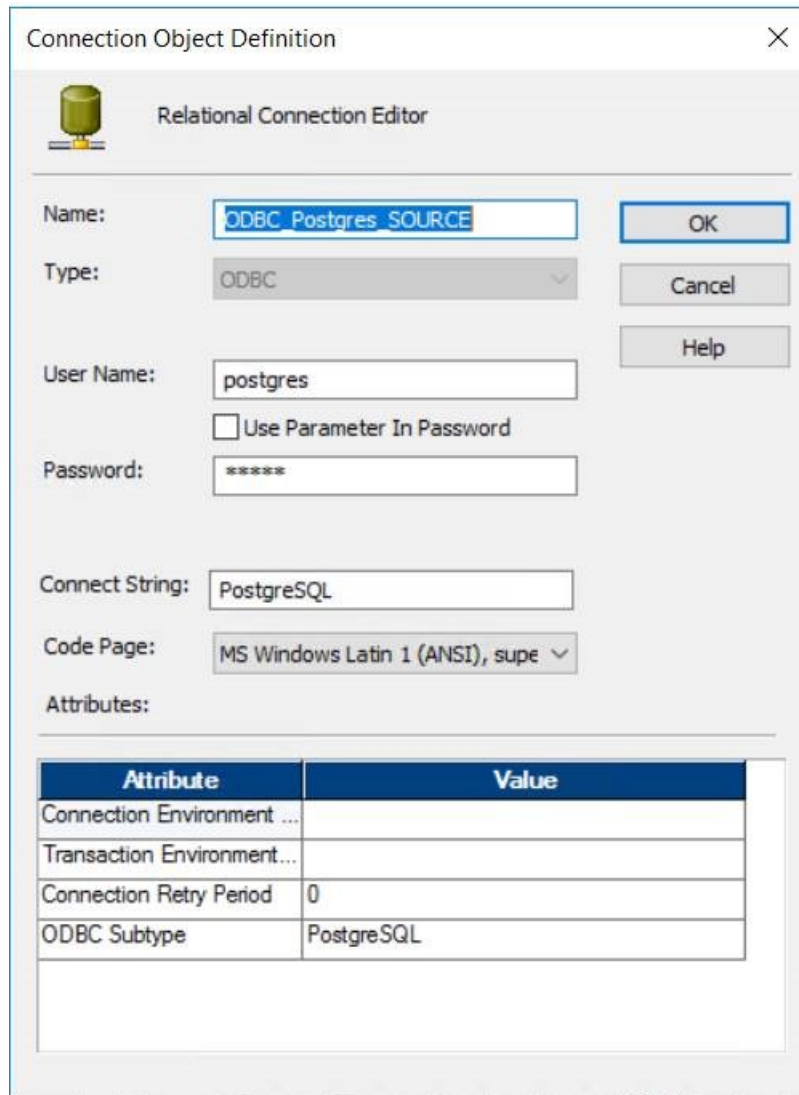
1. In the Workflow Manager, click **Connections**.

2. Select **Relational** from the list.

The **Relational Connection Browser** box appears. The following image shows the **Relational Connection Browser** box:
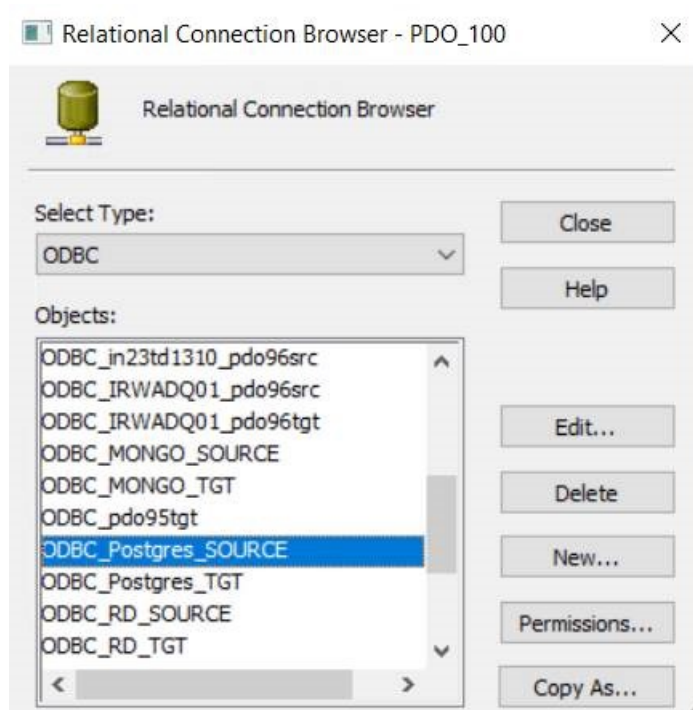
3. Select **Type** as **ODBC**.
4. Click **New**.

The **Connection Object Definition** box appears. The following image shows the **Connection Object Definition** box:

5. Configure the following relational connection properties:

Relational Connection Property	Description
Name	Enter a name for the connection.
Type	The connection type is set by default. You cannot edit this value.
User Name	Enter the user name to connect to the PostgreSQL database.
Password	Enter the password to connect to the PostgreSQL database.

Relational Connection Property	Description
Connect String	Enter the name of the ODBC data source that you created for the PostgreSQL database.
Code Page	Select the code page that the PowerCenter Integration Service must use to read or write data.
Attributes	Enter the **ODBC Subtype** attribute value as **PostgeSQL**.

The PostgreSQL ODBC connection is created successfully.

Rules and Guidelines for Functions in Pushdown Optimization

Use the following rules and guidelines when pushing functions to an PostgreSQL database:

- To push the TRUNC(DATE) function to the PostgreSQL database. you must define the date and format arguments.

- If you define only a string argument for the TO_DATE() and TO_CHAR() functions and omit the format argument, the PowerCenter Integration Service returns a string based on the default date format `MM/DD/YYYY HH24:MI:SS` specified in the session property.

- When you push the SYSTIMESTAMP() function to a PostgreSQL database, do not specify the format argument. If you specify the format for SYSTIMESTAMP, the database ignores the format and returns the complete time stamp.

- When you push the TO_BIGINT and TO_INTEGER functions to the PostgreSQL database, the PowerCenter Integration Service ignores the flag arguments.

- When you push the IN() function to the PostgreSQL database, the PowerCenter Integration Service ignores the CaseFlag argument.

- When you use the NS format string to set the nanoseconds for the ADD_TO_DATE() function, the PowerCenter Integration Service does not push the ADD_TO_DATE() function to PostgreSQL.

- The PowerCenter Integration Service cannot push TO_CHAR() and TO_DATE() functions to PostgreSQL if you use the following formats:

    NS

    JQW

    SSSSS

    RR

- You can use the following formats when you push the TRUNC(DATE) function to the PostgreSQL database:

    D

    HH24

    MI

    MM

    MS

    SS

    US

    YYYY

CHAPTER 7

PostgreSQL Data Type Reference

This chapter includes the following topics:

Data Type Reference Overview

PowerCenter uses the following data types in PostgreSQL mappings:

- PostgreSQL native data types. PostgreSQL data types appear in PostgreSQL definitions in a mapping.
- Transformation data types. Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the PowerCenter Integration Service uses to move data across platforms. They appear in all transformations in a mapping.

When the PowerCenter Integration Service reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the PowerCenter Integration Service writes to a target, it converts the transformation data types to the comparable native data types.

PostgreSQL and Transformation Data Types

The following table lists the PostgreSQL data types that PowerCenter supports and the corresponding transformation data types:

PostgreSQL Data Type	Transformation Data Type	Description
Smallint/Int2	Integer	Precision 10, scale 0
Int/Int4	Integer	Precision 10, scale 0
Bigint/int8	Bigint	Precision 19, scale 0
Decimal	Decimal	Precision 1 to 28, scale 0 to 28
Numeric	Decimal	Precision 1 to 28, scale 0 to 28

PostgreSQL Data Type	Transformation Data Type	Description
Real/Float4	Double	Precision 15, scale 0
Double/Float8	Double	Precision 15, scale 0
Smallserial/Int2	Integer	Precision 10, scale 0
Serial	Integer	Precision 10, scale 0
Bigserial/Serial8	BigInt	Precision 19, scale 0
Char	String	Precision 1
Char(n)	String(n)	n<=10485760
Varchar	String	Precision 104857600
Varchar(n)	String(n)	n <=10485760
Text	String	Precision 104857600
Bytea	Binary	Precision 104857600
Date	Date/Time	Precision 29, scale 9
Time	Date/Time	Precision 29, scale 9
Timestamp	Date/Time	Precision 29, scale 9
Timestamp with time zone	Date/Time	Precision 29, scale 9
Timestamp without time zone	Date/Time	Precision 29, scale 9
Boolean	String	Precision 6
Citext [1]	Text	Precision 104857600

[1]. Citext data type is considered as case-sensitive text in the following scenarios:
- When you configure a cached lookup and you define the lookup condition for the column of the Citext data type.
- When you configure an Expression transformation for a column of the Citext data type.

INDEX