Informatica®

Informatica® Cloud Data Integration

PostgreSQL Connector

Informatica Cloud Data Integration PostgreSQL Connector
November 2023

Publication Date: 2024-04-07

Table of Contents

Preface

Use *PostgreSQL Connector* to learn how to read from or write to PostgreSQL by using Cloud Data Integration. Learn to create a PostgreSQL connection, develop mappings, and run mapping, dynamic mapping, and data transfer tasks in Data Integration. You can also learn how to configure SQL ELT optimization using an ODBC connection.

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 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 Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

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 Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, on the Informatica Intelligent Cloud Services Status page, click **SUBSCRIBE TO UPDATES**. You can choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

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

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Introduction to PostgreSQL Connector

You can use PostgreSQL Connector to securely read data from or write data to the PostgreSQL databases.

You can also use PostgreSQL Connector to connect to the following databases:

- Amazon Aurora PostgreSQL
- Azure PostgreSQL private endpoint on a virtual network

PostgreSQL sources and targets represent tables in PostgreSQL. You can create a PostgreSQL connection and use the connection in mappings and mapping tasks to read from or write data to PostgreSQL. Create a mapping task to process data based on the data flow logic defined in a mapping or integration template.

When you run an PostgreSQL mapping or mapping task, the Secure Agent writes data to PostgreSQL based on the defined data logic and the configured PostgreSQL connection.

You can switch mappings to advanced mode to include transformations and functions that enable advanced functionality.

Example

You work for an organization that stores purchase order details, such as customer ID, item codes, and item quantity in a MySQL database. You need to analyze purchase order details and move data from the MySQL database to a cloud-based environment. Create a mapping to read all the purchase records from the MySQL database and write them to an PostgreSQL target for data analysis.

PostgreSQL Connector assets

Create assets in Data Integration to integrate data using PostgreSQL Connector.

When you use PostgreSQL Connector, you can include the following Data Integration assets:

- Data transfer task
- Dynamic mapping task
- Mapping
- Mapping task

For more information about configuring assets and transformations, see *Mappings*, *Transformations*, and *Tasks* in the Data Integration documentation.

CHAPTER 2

Connections for PostgreSQL

PostgreSQL connection enables you to read data from or write data to PostgreSQL. You can use PostgreSQL connections to specify sources or targets in mappings and mapping tasks.

Create an PostgreSQL connection on the **Connections** page and associate it with a mapping or mapping task. Define the source and target properties to read from or write to PostgreSQL.

Prepare for authentication

You can configure Database or Kerberos authentication method to connect to a PostgreSQL database. Before you configure the connection properties, you need to keep the authentication details handy based on the authentication type that you want to use. To configure Database authentication, you need the user name, password, host name, port, database name from your postgreSQL account. To configure Kerberos authentication, you need the service principle name, host name, port, and database name from your postgreSQL account.

Before you configure Kerberos authentication, you need to perform certain prerequisite tasks.

Prepare for Kerberos authentication

You can use Kerberos authentication to connect to PostgreSQL databases by placing the required configuration files on the Secure Agent machine.

When you configure Kerberos authentication to connect to PostgreSQL, consider the following guidelines:

- You can't use the Hosted Agent or serverless runtime environment.
- Ensure that the Secure Agent and database server that you use are registered in the KDC server.
- You can't add more than one KDC to a krb5.conf file.
- You can't generate a credential cache file for more than one Kerberos principal user.

Configuring the Kerberos authentication

Before you use Kerberos authentication to connect to PostgreSQL on Linux or Windows, the organization administrator needs to perform the prerequisite tasks.

1. To configure the Java Authentication and Authorization Service configuration file (JAAS), perform the following tasks:
   a. Create a JAAS configuration file on the Secure Agent machine.

b. Add the following entries to the JAAS configuration file:

```
JDBC_DRIVER_01 {
com.sun.security.auth.module.Krb5LoginModule required useTicketCache=true;
};
```

2. To configure the `krb5.conf` file, perform the following tasks:
   a. Create a `krb5.conf` file on the Secure Agent machine.
   b. Add the details of the Key Distribution Center (KDC) and admin server to the `krb5.conf` file in the following format:

```
[libdefaults]
default_realm = <Realm name>
forwardable = true
ticket_lifetime = 24h

[realms]
<REALM NAME> = {
kdc = <Location where KDC is installed>
admin_server = <Location where KDC is installed>
                                  }
 [domain_realm]
<domain name or host name> = <Domain name or host name of Kerberos>
<domain name or host name> = <Domain name or host name of Kerberos>
```

3. Set the following environment variables on the Secure Agent machine.
   For more information about the required environment variables, see "Setting environment variables" on page 9.
4. Restart the Secure Agent.
5. To generate the credential cache file on the Secure Agent machine and use Kerberos authentication to connect to PostgreSQL, perform the following tasks:
   a. On the Secure Agent machine, run the following command and specify the PostgreSQL user name and realm name:

```
Kinit <user name>@<realm_name>
```

   b. When prompted, enter the password for the Kerberos principal user.

Setting environment variables

To use Kerberos authentication to connect to PostgreSQL, you need to set the required environment variables on the Secure Agent machine.

Set the following environment variables:

- `setenv KRB5CCNAME <Absolute path and file name of the credentials cache file>`
- `setenv KRB5_CONFIG <Absolute path of the Kerberos configuration file>\krb5.conf`
- `setenv JAASCONFIG <Absolute path of the JAAS config file>\<File name>.conf`

After you set the environmental variables, you need to restart the Secure Agent.

Alternatively, you can add the `KRB5_CONFIG` and `JAASCONFIG` environment variables when you create a PostgreSQL connection.

To add the environment variables when you configure a connection with Kerberos authentication, you need to add the *KRB5_CONFIG* and *JAASCONFIG* properties in the **Additional Kerberos Properties** field in a PostgreSQL connection.

For example, add the properties in the following format:

```
KRB5_CONFIG=<Absolute path of the Kerberos configuration file>
\krb5.conf;JAASCONFIG=<Absolute path of the JAAS config file>\<File name>.conf
```

Note: Ensure that you separate each key-value pair with a semicolon.

Connect to PostgreSQL

Let's configure the PostgreSQL connection properties to connect to PostgreSQL.

Before you begin

Before you get started, get the required information from your PostgreSQL account based on the authentication method that you want to use.

Check out "Prepare for authentication" on page 8 to learn more about the authentication prerequisites.

Connection details

The following table describes the PostgreSQL connection properties:

Property	Description
Connection Name	Name of the connection. Each connection name must be unique within the organization. Connection names can contain alphanumeric characters, spaces, and the following special characters: _ . + -, Maximum length is 255 characters.
Description	Description of the connection. Maximum length is 4000 characters.
Type	PostgreSQL
Runtime Environment	The name of the runtime environment where you want to run the tasks. Select a Secure Agent, Hosted Agent, or serverless runtime environment.

Authentication types

You can configure one of the following authentication modes to connect to PostgreSQL databases:

- Database: Uses your PostgreSQL user name and password to connect to PostgreSQL.
- Kerberos: Uses Kerberos authentication to connect to PostgreSQL.
  When you choose this option on Windows, ensure that the user account that starts the Secure Agent service is available in the PostgreSQL database. You don't need to enter your credentials to access PostgreSQL.

  Note: You can't configure Kerberos authentication when you use a Hosted Agent or serverless runtime environment.

Select the required authentication type and then configure the authentication-specific parameters.

Default is Database.

Database authentication

To configure Database authentication, you need the user name, password, host name, port, database name from your postgreSQL account.

The following table describes the basic connection properties for Database authentication:

Property	Description
User Name	User name to access the PostgreSQL database.
Password	Password for the PostgreSQL database user name.
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 Name	The PostgreSQL database name.

Kerberos authentication

To configure Kerberos authentication, you need the service principle name, host name, port, and database name from your postgreSQL account.

The following table describes the basic connection properties for Kerberos authentication:

Property	Description
Service Principal Name	Service principal name that you want to use for Kerberos authentication. Specify the service principal name in the following format: `<Service_Name>/<Fully_Qualified_Domain_Name>@<REALM.COM>` - Service_Name is the name of the service hosting the instance. - Fully_Qualified_Domain_Name is the fully qualified domain name of the host machine. - REALM.COM is the domain name of the host machine. This value is optional. If you do not specify the realm name, the default realm is used.
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 Name	The PostgreSQL database name.

Advanced settings

The following table describes the advanced connection property for Kerberos authentication:

Property	Description
Additional Kerberos Properties	Additional connection properties to use Kerberos authentication to connect to PostgreSQL. This property appears only if you select the authentication mode as Kerberos. Enter properties in the following format: `<parameter name>=<parameter value>` If you enter more than one property, separate each key-value pair with a semicolon.

Advanced settings

The following table describes the advanced connection properties:

Property	Description
Schema Name	The schema name. If you don't specify the schema name, all the schemas available in the database are listed while importing the source object in Data Integration.
Additional Connection Properties	Additional connection parameters that you want to use. Provide the connection parameters as semicolon-separated key-value pairs.

Encryption types

The encryption method determines whether the data exchanged between the Secure Agent 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 can't configure SSL, the connection fails.
- requestSSL. Attempts to establish a connection using SSL. If the PostgreSQL database server can't configure SSL, the Secure Agent establishes an unencrypted connection.

Default is noEncryption.

Note: SSL is not applicable when you use the Hosted Agent. You can configure SSL when you use the Secure Agent or the serverless runtime environment.

SSL

The following table describes the advanced connection properties for SSL encryption:

Property	Description
Validate Server Certificate	Applicable if you select SSL or requestSSL as the encryption method. Select the Validate Server Certificate option so that the Secure Agent validates the server certificate that is sent by the PostgreSQL database server. If you specify the Host Name In Certificate property, the Secure Agent 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. For the serverless runtime environment, specify the following certificate path in the serverless agent directory: `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<TrustStore_filename>`
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. A host name for providing additional security. The Secure Agent validates the host name included in the connection with the host name in the SSL certificate.
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. For the serverless runtime environment, specify the following certificate path in the serverless agent directory: `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<KeyStore_filename>`
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.
Use SSLv3	Required if you select SSL or requestSSL as the encryption method. Select this option to use SSLv3 as the cryptographic protocol for an encrypted connection.
Use TLSv1.2	Required if you select SSL or requestSSL as the encryption method. Select this option to use TLSv1.2 as the cryptographic protocol for an encrypted connection.

Request SSL

The following table describes the advanced connection properties for Request SSL encryption:

Property	Description
Validate Server Certificate	Applicable if you select SSL or requestSSL as the encryption method. Select the Validate Server Certificate option so that the Secure Agent validates the server certificate that is sent by the PostgreSQL database server. If you specify the Host Name In Certificate property, the Secure Agent 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. For the serverless runtime environment, specify the following certificate path in the serverless agent directory: `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<TrustStore_filename>`
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. A host name for providing additional security. The Secure Agent validates the host name included in the connection with the host name in the SSL certificate.
Use SSLv3	Required if you select SSL or requestSSL as the encryption method. Select this option to use SSLv3 as the cryptographic protocol for an encrypted connection.
Use TLSv1.2	Required if you select SSL or requestSSL as the encryption method. Select this option to use TLSv1.2 as the cryptographic protocol for an encrypted connection.

Configure SSL with serverless runtime environment

You can use the serverless runtime environment with PostgreSQL Connector to connect to an SSL-enabled PostgreSQL database.

Before you configure a secure PostgreSQL connection using the serverless runtime environment, complete the following prerequisite tasks to add the SSL certificates to the serverless runtime location:

1. Create the following structure for the serverless agent configuration in AWS or Azure: `<Supplementary file location>/serverless_agent_config`
2. Add the truststore and keystore certificates in the Amazon S3 bucket or Azure container in the following location in your AWS or Azure account: `<Supplementary file location>/serverless_agent_config/SSL`
3. Copy the following code snippet to a text editor:

```
version: 1
agent:
  agentAutoApply:
```

```
general:
  sslStore:
    - fileCopy:
        sourcePath: SSL/<TrustStore_filename>
    - fileCopy:
        sourcePath: SSL/<KeyStore_filename>
```

where the source path is the directory path of the certificate files in AWS or Azure.

4. Ensure that the syntax and indentations are valid, and then save the file as `serverlessUserAgentConfig.yml` in the following AWS or Azure location: `<Supplementary file location>/serverless_agent_config`
   When the .yml file runs, the SSL certificates are copied from the AWS or Azure location to the serverless agent directory.
5. In the PostgreSQL connection properties, specify the following certificate path in the serverless agent directory in the **Trust Store** and **Key Store** fields: `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<cert_filename>`

CHAPTER 3

Mappings and mapping tasks with PostgreSQL Connector

Use the Data Integration Mapping Designer to create a mapping. When you create a mapping, configure a Source or Target transformation to represent a PostgreSQL object.

In advanced mode, the Mapping Designer updates the mapping canvas to include transformations and functions that enable advanced functionality.

Describe the flow of data from source and target along with the required transformations before the agent writes data to the target. When you create a mapping task, select the mapping that you want to use. Use the Mapping Task wizard to create a mapping task. Validate and run the mapping to read data from sources and write to a target. The mapping task processes data based on the data flow logic you define in the mapping.

When you configure a mapping, you can parameterize the source object and the PostgreSQL connection.

PostgreSQL sources in mappings

In a mapping, you can configure a Source transformation to represent a PostgreSQL source.

The following table describes the PostgreSQL source properties that you can configure in a Source transformation:

Property	Description
Connection	Name of the source connection, or create a connection parameter. If you want to overwrite the target connection properties at runtime, select the **Allow parameter to be overridden at run time** option. Specify the parameter file directory and name in the advanced session properties.
Source type	Type of the source object. Select Single Object, Multiple Objects, Query, or Parameter. When you select multiple objects as the source type to read from multiple PostgreSQL sources, you can use the advanced relationship option to define the relationship for the objects that you want to join. When you select query as the source type, specify the SQL statement in the **Query** field. You can partially parameterize the query source type. If you want to overwrite the query object at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, the Secure Agent uses the parameters from the file that you specify in the advanced session properties.

Property	Description
Object	Name of the source object.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the source object or click **New Parameter** to define a new parameter for the source object. The **Parameter** property appears only if you select parameter as the source type. If you want to overwrite the source object at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, the Secure Agent uses the parameters from the file that you specify in the advanced session properties.

The following table describes the PostgreSQL query options that you can configure in a Source transformation:

Property	Description
Filter	Filter value in a read operation. Click **Configure** to add conditions to filter records and reduce the number of rows that the Secure Agent reads from the source. You can specify the following filter conditions: - **Not parameterized**. Use a basic filter to specify the object, field, operator, and value to select specific records. - **Completely parameterized**. Use a parameter to represent the field mapping. - **Advanced**. Use an advanced filter to define a more complex filter condition.
Sort	Add conditions to sort records. You can specify the following sort conditions: - **Not parameterized.** Select the fields and type of sorting to use. - **Parameterized.** Use a parameter to specify the sort option. - **Sort Order.** Sorts data in ascending or descending order, according to a specified sort condition.

The following table describes the PostgreSQL advanced source properties that you can configure in a Source transformation:

Property	Description
Pre-SQL	The pre-SQL commands to run a query before you read data from PostgreSQL. You can partially parameterize pre-SQL with values specified in a parameter file.
Post-SQL	The post-SQL commands to run a query after you write data to a target. You can partially parameterize post-SQL with values specified in a parameter file.
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. You can specify a maximum fetch size of 2147483647. Default is 100000.
Schema Name	Overrides the schema name of the source object.
Source Table Name	Overrides the default PostgreSQL source table name.

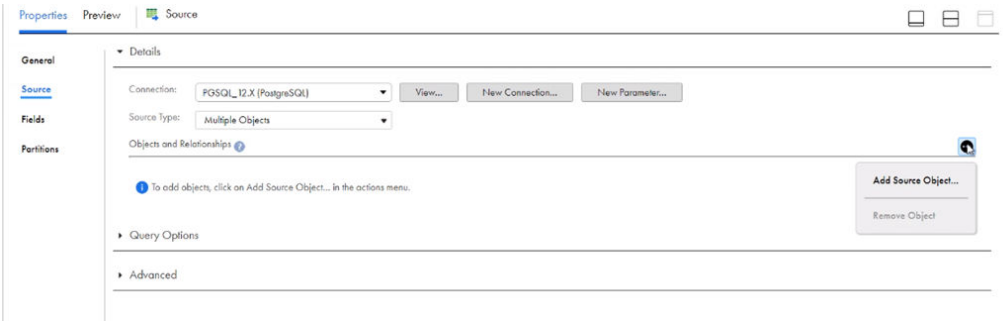
Property	Description
Tracing Level	Sets the amount of details that appear in the log file. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
SQL Override	The SQL statement to override the default query generated from the specified source type to read data from the PostgreSQL source. You can partially parameterize SQL override with values specified in a parameter file. 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 applicable when you enable partitioning. If you specify an SQL override and configure partitioning, the mapping fails.

Adding multiple source objects

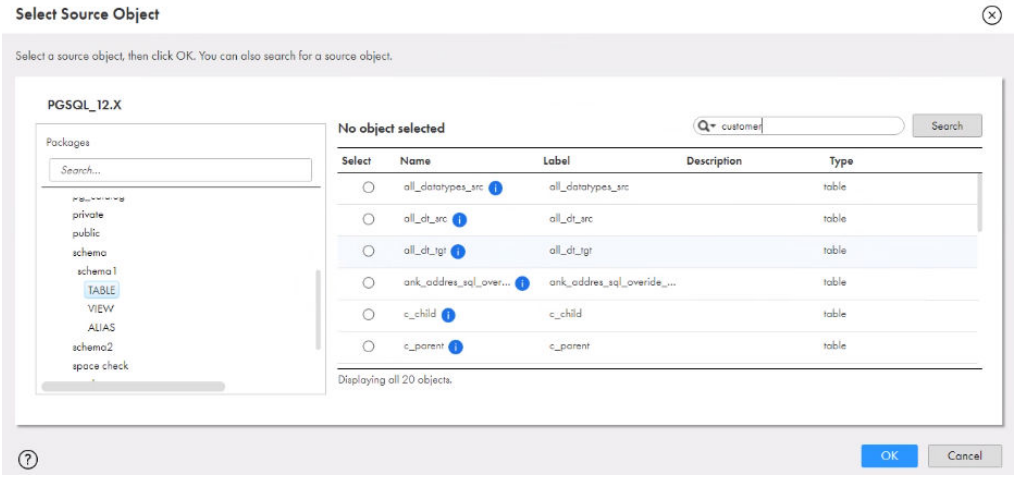
When you create a Source transformation, you can select multiple PostgreSQL objects as the source type, and then configure an advanced relationship to combine the tables.

Perform the following steps to join multiple objects in a Source transformation:

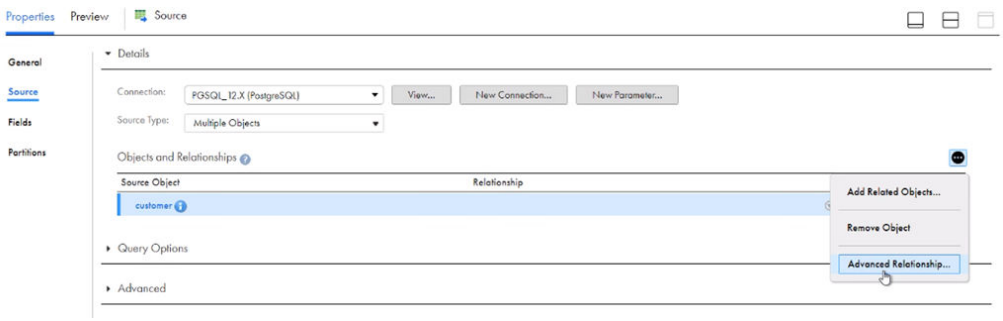
1. In the Source transformation, select the **Source Type** as **Multiple Objects**.
2. From the **Actions** menu, select **Add Source Object**.



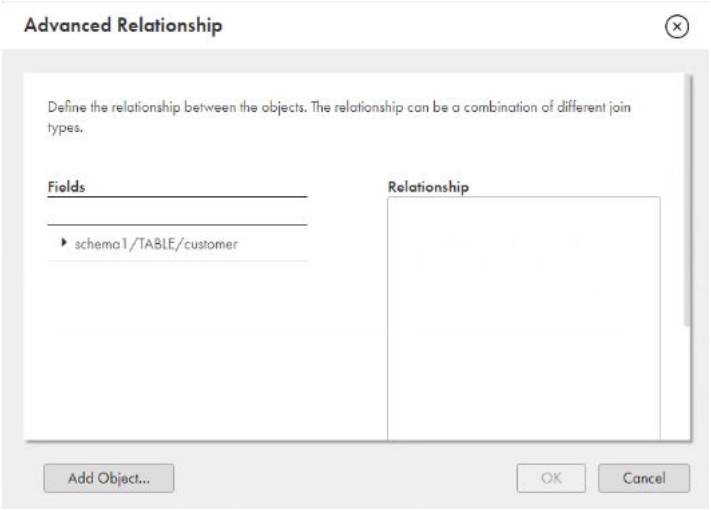
3. In the **Select Source Object** window, select the source object that you want to add, and then click **OK**.



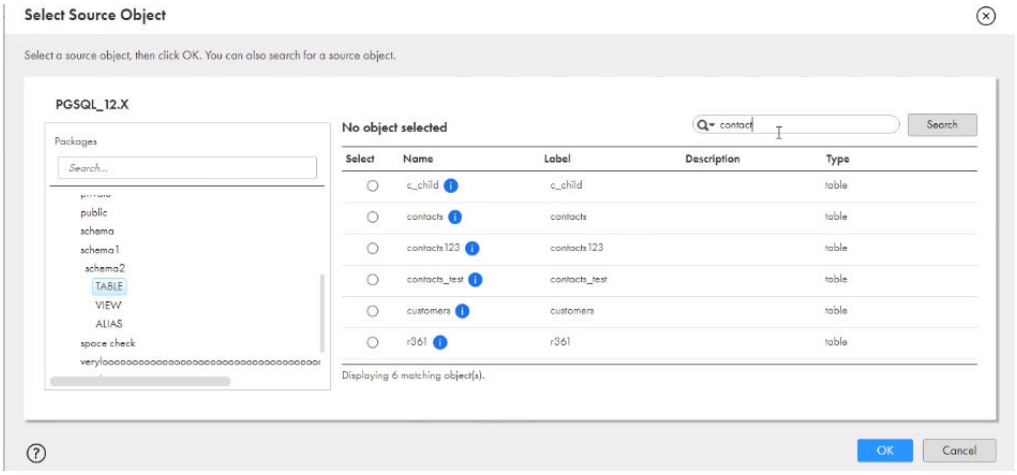
4. From the **Actions** menu, select **Advanced Relationship**.



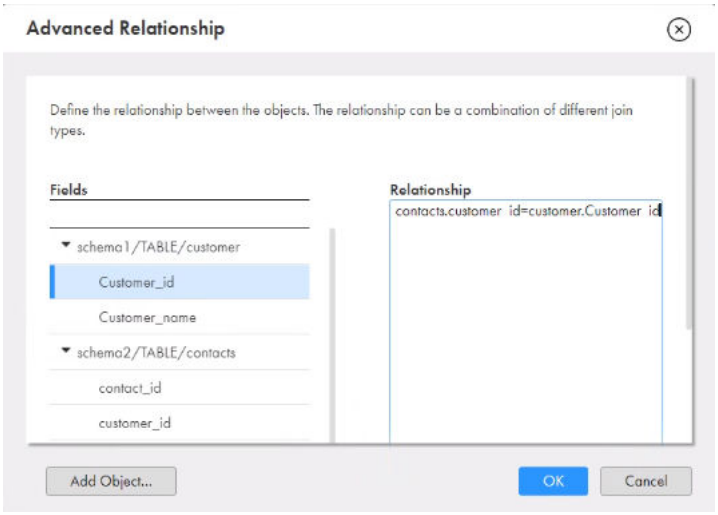
5. In the **Advanced Relationship** window, click **Add Object** to add more objects.



6. In the **Select Source Object** window, select the source object with which you want to define a relationship, and then click **OK**.

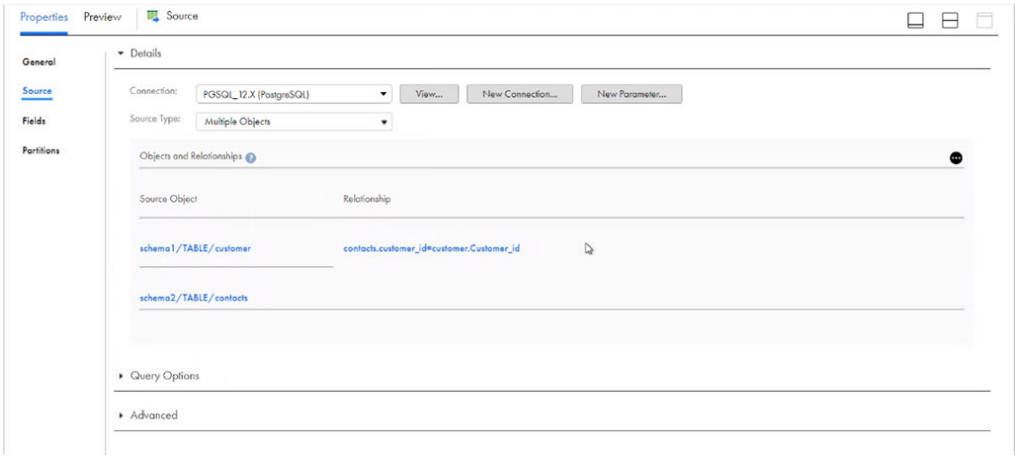


7. In the **Advanced Relationship** window, select the required fields, and set the conditions or specify a query to define a relationship between the tables.



8. Click **OK**.

   The following image shows an example of an advanced relationship condition defined between the PostgreSQL tables:



Rules and guidelines for adding multiple source objects

Consider the following rules and guidelines when you add multiple source objects:

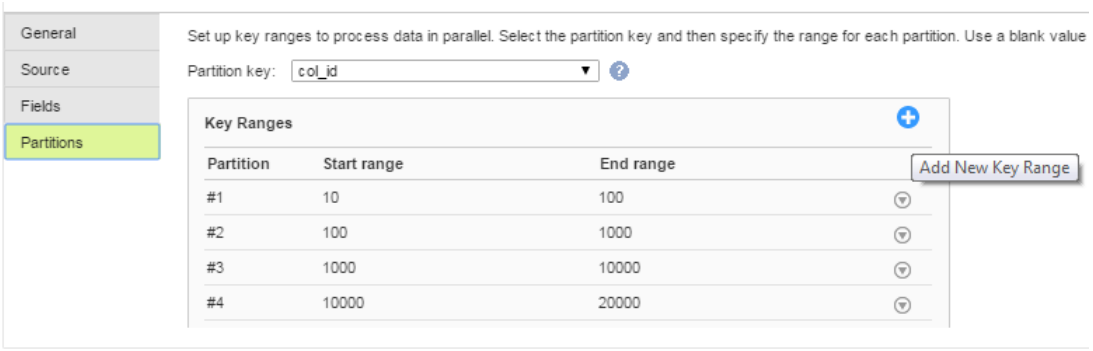
- You cannot use a self join when you add multiple source objects.
- You cannot search for a schema when you add a related source object. You must scroll down and manually select the schema.
- When you select parent and child objects that have a primary key and foreign key relationship, ensure that the foreign key of the related object is not a primary key in the table.

Configuring key range partition

Configure key range partition to partition PostgreSQL data based on field values. Key range is supported for only the numeric and date/time data types.

1. In **Source Properties**, click the **Partitions** tab.
2. Select the required **Partition Key** from the list.
3. Click **Add New Key Range** to add partitions.

   The following image displays the details of the **Partitions** tab.



4. Specify the **Start range** and **End range**.

PostgreSQL targets in mappings

To write data to PostgreSQL, configure a PostgreSQL object as the target in a mapping.

Specify the name and description of the PostgreSQL target. Configure the target and advanced properties for the target object .

The following table describes the target properties that you can configure in a Target transformation:

Property	Description
Connection	Name of the target connection, or create a connection parameter. If you want to overwrite the target connection properties at runtime, select the **Allow parameter to be overridden at run time** option. Specify the parameter file directory and name in the advanced session properties.
Target Type	Type of the target object. Select Single Object or Parameter.
Object	Name of the target object. You can select an existing target object from the displayed list or you can create a target at runtime. When you want to create a target at runtime, specify the target object name and the path for the target object. For a list of supported data types that you can write to PostgreSQL using the **Create New at Runtime** option, see the Data Type References chapter.

Property	Description
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the target object or click **New Parameter** to define a new parameter for the target object. The **Parameter** property appears only if you select parameter as the target type. If you want to overwrite the target object at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, the Secure Agent uses the parameters from the file that you specify in the advanced session properties.
Operation	Type of the target operation. Select Insert, Upsert, Update, Delete, and Data Driven.
Data Driven Condition	Enables you to define expressions that flag rows for an insert, update, delete, or reject operation. Appears only when the operation type is **Data Driven**. **Note:** When you configure the **DD_REJECT** operation in the data driven mode to reject data to PostgreSQL, the rejected records are not written to the error files and the session log shows the number of rejected rows as zero.
Update Columns	Specify the columns that you want to use as a logical primary key for performing update, upsert, and delete operations on the target. This field is not required if the target table already has a primary key. If the target table does not have a primary key, ensure that the columns selected in the **Update Columns** field has a unique constraint for the upsert operation. This property is not applicable for the insert operation.

The following table describes the PostgreSQL advanced target properties:

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 Secure Agent inserts the entries.
Override Target Query	An SQL statement to override the default update query that the Secure Agent generates for the update operation.
Schema Name	Overrides the schema name of the target object.
Target Table Name	Overrides the default PostgreSQL target table name.
Pre-SQL	The pre-SQL commands to run a query before you read data from a source. You can partially parameterize pre-SQL with values specified in a parameter file.
Post-SQL	The post-SQL commands to run a query after you write data to PostgreSQL. You can partially parameterize post-SQL with values specified in a parameter file.
Truncate Target	The Secure Agent truncates the target before writing the data.

Property	Description
Enable target bulk load	Performs bulk upload when you configure an insert operation to write to PostgreSQL. Select this option to improve the performance of inserting data in bulk to PostgreSQL. Default is unselected. **Note:** When you enable the target bulk mode to insert data to PostgreSQL, error files are not generated for rejected records.
Batch size	The number of rows that the Secure Agent writes in a single batch to PostgreSQL. Specify a batch size value that is greater than zero. Applicable if you select the **Enable target bulk load** option.
Reject File Directory	The directory that stores the rejected files. Specify the directory where you want to store the rejected files.
Reject File Name	Name of the rejected file that is stored in the reject file directory.
Forward Rejected Rows	Not applicable.

Specifying a target

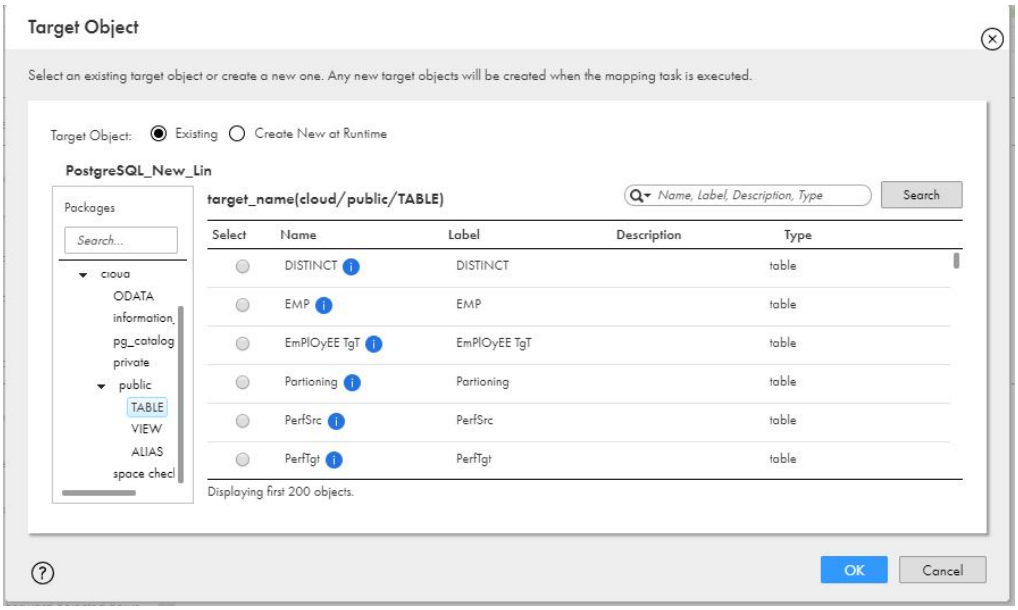
You can use an existing target or create a target to hold the results of a mapping.

To specify the target properties, perform the following tasks:

1. Select the Target transformation in the mapping.
2. On the **Incoming Fields** tab, configure field rules to specify the fields to include in the target.
3. To specify the target, click the **Target** tab.
4. Select the target connection as PostgreSQL.
5. For the target type, choose **Single Object** or **Parameter**.
6. Specify the target object or parameter. Click **Select**, choose a target object, and then click **OK**.

   You can select an existing target object or create a new target object at run time and specify the object name.
7. To select an existing target object, select **Existing** and then select the required target table.

The following image shows the available target tables:



8. To create a new target at run time, perform the following tasks:

   Note: When you choose to create a target at runtime and if the target table does not exist in the target database, the Secure Agent creates the table when you run the task. If a table with the same name already exists in the database, the Secure Agent uses the existing table.

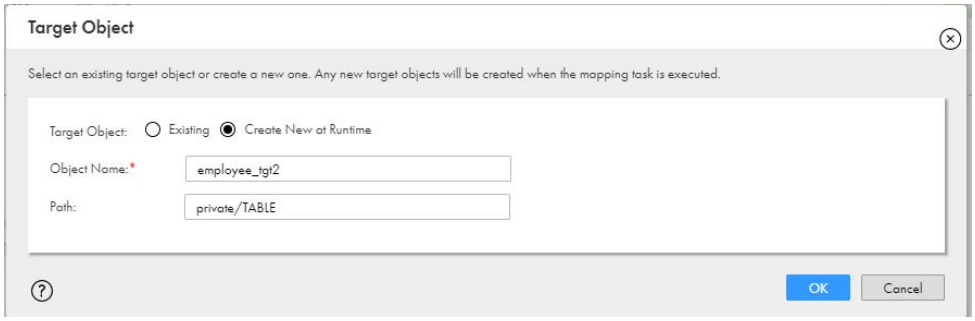
   a. Select **Create New at Runtime**.

   b. In the **Object Name** field, specify the table name for the PostgreSQL target.

   c. In the **Path** field, specify the schema for the PostgreSQL target.

      You must specify the schema in the following format: `<Schema_Name>/<TableType>`, where the TableType is TABLE. For example, if you specify the path as `private/TABLE`, the table is created in the schema named private in PostgreSQL. If you do not specify the path, the Secure Agent considers the schema from the connection properties. If you do not specify the schema in the connection properties, the target table is created in the public schema.

      The following image shows the target object properties for the **Create New at Runtime** option:



9. Specify the operation and advanced properties for the target, as required.

Override the target query

You can configure an override target query to override the update query that the Secure Agent generates for the update operation. To override the update query, select **Update As Update** from the **Update Mode** list in the advanced target properties.

You must specify the override target query with a valid SQL syntax because PostgreSQL Connector does not validate the update query.

Specify the override target query in the following format:

```
UPDATE <schema name>.<Target table name here>
  SET <Column1> = :TU.<Column1>,
          <Column2> = :TU.<Column2>,
          <ColumnN> = :TU.<ColumnN>  WHERE <Upadate Column1> = :TU.<Upadate Column1>
```

where, :TU. in the update query, that represents the incoming data source for the target port, must match the target table column names and the WHERE clause is mandatory in the query.

If the number of :TU columns are less than the total number of columns in the PostgreSQL target, you must map the required fields to the PostgreSQL target.

When you use the **Override Target Query** property for a PostgreSQL target, consider the following rules:

- You cannot use the override target query for an insert, upsert, delete, or data driven operation.
- You cannot change the order of the column mappings using the override target query.
- You cannot specify multiple override target queries for an update operation.

PostgreSQL lookups in mappings

In a mapping, you can configure a Lookup transformation to represent an PostgreSQL lookup.

The following table describes the PostgreSQL lookup properties that you can configure in a Lookup transformation:

Property	Description
Connection	Name of the lookup connection, or create a connection parameter. If you want to overwrite the lookup connection properties at runtime, select the **Allow parameter to be overridden at run time** option. Specify the parameter file directory and name in the advanced session properties.
Source type	Type of the lookup object. Select Single Object or Parameter.
Object	Name of the lookup object.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the lookup object or click **New Parameter** to define a new parameter for the lookup object. The **Parameter** property appears only if you select parameter as the lookup type. If you want to overwrite the lookup object at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, the Secure Agent uses the parameters from the file that you specify in the advanced session properties.

Property	Description
Multiple Matches	The behavior when the lookup condition returns multiple matches. You can return all rows, any row, the first row, the last row, or an error. You can select from the following options in the lookup object properties to determine the behavior: - Return first row - Return last row - Return any row - Return all rows - Report error
Schema Name	Overrides the schema name of the lookup object.
Source Table Name	Overrides the default PostgreSQL source table name in the Lookup transformation.

Rules and guidelines for custom query, SQL override, and batch size

Consider the following rules and gudelines when you configure a custom query or SQL override:

- If you use a custom query in the Source transformation and configure create target at runtime in the Target transformation, you must specify the `ExtendedColumnMetadata=true` property in the **Additional Connection Properties** field in the PostgreSQL connection. The property helps retain the constraints from the source in the target object.
- Ensure that the list of selected columns, the 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.
- When you configure an SQL override for a PostgreSQL source, partitioning is not applicable.
- To configure the batch size consider using the following formula: `BatchSize = (<Threshold Memory in Bytes>)/(3* <Approximate Row Size in Bytes>).`
  In the formula, the threshold memory in bytes represents the maximum memory consumption that you want the operation to consume. The approximate row size is the size of a row, which you can calculate from the target table field definition.

CHAPTER 4

PostgreSQL objects in mapping tasks

When you configure a mapping task, you can configure advanced properties for PostgreSQL sources and targets.

PostgreSQL sources in mapping tasks

For PostgreSQL source connections used in template-based mapping tasks, you can configure advanced properties in the Sources page.

You can configure the following advanced properties:

Property	Description
Pre-SQL	The pre-SQL commands to run a query before you read data from PostgreSQL. You can partially parameterize pre-SQL with values specified in a parameter file.
Post-SQL	The post-SQL commands to run a query after you write data to a target. You can partially parameterize post-SQL with values specified in a parameter file.
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. You can specify a maximum fetch size of 2147483647. Default is 100000.
Schema Name	Overrides the schema name of the source object.
Source Table Name	Overrides the default PostgreSQL source table name.

Property	Description
Tracing Level	Sets the amount of details that appear in the log file. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
SQL Override	The SQL statement to override the default query generated from the specified source type to read data from the PostgreSQL source. You can partially parameterize SQL override with values specified in a parameter file. 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 applicable when you enable partitioning. If you specify an SQL override and configure partitioning, the mapping fails.

PostgreSQL targets in mappings

For PostgreSQL target connections used in mapping tasks, you can configure advanced target properties in the **Targets** page of the Mapping Task wizard.

You can configure the following advanced target properties:

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 Secure Agent inserts the entries.
Override Target Query	An SQL statement to override the default update query that the Secure Agent generates for the update operation.
Schema Name	Overrides the schema name of the target object.
Target Table Name	Overrides the default PostgreSQL target table name.
Pre-SQL	The pre-SQL commands to run a query before you read data from a source. You can partially parameterize pre-SQL with values specified in a parameter file.
Post-SQL	The post-SQL commands to run a query after you write data to PostgreSQL. You can partially parameterize post-SQL with values specified in a parameter file.
Truncate Target	The Secure Agent truncates the target before writing the data.
Enable target bulk load	Performs bulk upload when you configure an insert operation to write to PostgreSQL. Select this option to improve the performance of inserting data in bulk to PostgreSQL. Default is unselected. **Note:** When you enable the target bulk mode to insert data to PostgreSQL, error files are not generated for rejected records.

Property	Description
Batch size	The number of rows that the Secure Agent writes in a single batch to PostgreSQL. Specify a batch size value that is greater than zero. Applicable if you select the **Enable target bulk load** option.
Reject File Directory	The directory that stores the rejected files. Specify the directory where you want to store the rejected files.
Reject File Name	Name of the rejected file that is stored in the reject file directory.
Forward Rejected Rows	Not applicable.

Rules and guidelines for error files

Error files are not generated for rejected records in a mapping task when one of the following conditions are true:

- You have enabled the target bulk mode to write data to PostgreSQL.
- You have set the **Stop on errors** session property value to **1** in the mapping task.

Dynamic schema handling for mappings

You can choose how Data Integration handles changes that you make to the data object schemas. To refresh the schema every time the mapping task runs, you can enable dynamic schema handling in the task.

A schema change includes one or more of the following changes to the data object:

- Fields added.
- Fields updated for data type, precision, or scale.

Configure schema change handling on the **Schedule** page when you configure the task.

The following table describes the schema change handling options:

Option	Description
Asynchronous	Default. Data Integration refreshes the schema when you edit the mapping or mapping task, and when Informatica Intelligent Cloud Services is upgraded.
Dynamic	Data Integration refreshes the schema every time the task runs. You can choose from the following options to refresh the schema: - **Alter and Apply changes**. Alters the target schema and adds the new fields from the source. - **Don't Apply DDL Changes**. Does not apply the schema changes to the target. - **Drop Current and Recreate**. Drops the existing target table and then recreates the target table at runtime using all the incoming metadata fields from the source.

For more information, see the "Schema change handling" topic in *Tasks* in the Data Integration documentation.

Rules and guidelines for dynamic schema handling in mappings

Consider the following rules and guidelines when you enable dynamic schema change handling for tasks:

- You must use an existing target in the Target transformation to write the data.
- You can't rename the column in the target.
- When you use an exiting target table to write the data and delete a column from the source table, the deleted column still appears in the target table even though you run the task.
- You can't enable the **Alter and Apply changes** option for the task when you change any constraint such as primary key, not null, or unique key in the source schema.

CHAPTER 5

Lookup transformation

You can configure a connected Lookup transformation when you use a PostgreSQL connection in a mapping to return data from a PostgreSQL source based on a specified lookup condition.

You can configure either a cached or uncached lookup to determine whether to cache the lookup data during the runtime session.

When you enable caching, the Data Integration Server queries the lookup source once and caches the values for use during the session. Caching the lookup values improves the session performance. When you disable caching, each time a row passes into the transformation, a SELECT statement gets the lookup values.

When you configure a Lookup transformation, you can specify an SQL statement you want to use to override the default SQL statement for querying lookup values. The Data Integration Server uses the specified lookup SQL override statement and overrides the default SQL statement to query the lookup table. You must use the SQL override with lookup caching enabled.

The Data Integration Server performs the following steps when you run a connected lookup transformation:

1. The Data Integration Server passes values from another transformation to input ports in the Lookup transformation.
2. For each input row, the Data Integration Server queries the lookup source or the cache based on the lookup ports and the lookup condition in the transformation:
   - If the transformation is uncached, the Data Integration Server returns values from the source based on the lookup query.
   - If the transformation is enabled for caching, the Data Integration Server queries the lookup cache during the session and returns the values from the lookup cache.
3. The Data Integration Server then passes the returned data to the next transformation in the mapping.

Configuring a cached or uncached connected Lookup transformation

To configure a connected Lookup transformation, do not select the **Unconnected Lookup** checkbox in the general properties of the Lookup object. To configure a cached or uncached lookup in the Lookup transformation, in the advanced properties of the lookup object, enable or disable the **Lookup Caching Enabled** based on your requirement. The **Lookup Caching Enabled** checkbox is enabled by default.

The default **Lookup Cache Directory Name** is $PMCacheDir, which is the directory to store the cached lookup data when you select Lookup Caching Enabled. If you do not want a cached lookup, clear the checkbox.

For more information about configuring the lookup properties in a Lookup transformation, see *Transformations*.

Overriding the default lookup query in a Lookup transformation

When you configure a lookup, you specify the lookup condition to query the lookup table in the **Lookup Condition** tab of the Lookup transformation. When you run the mapping, the Data Integration Server finds the data in the lookup source using the query generated from the lookup condition.

If you want to override the lookup query generated from the lookup condition, you can specify an SQL statement to override the lookup query. The Data Integration Server uses the specified lookup SQL override statement to query the lookup table.

To override, you must specify the SQL statement in the **SQL Override** field in the advanced properties of the **Lookup Object** tab. Ensure that you configure the SQL override with lookup caching enabled. To do this, select the **Lookup Caching Enabled** field in the **Advanced** tab of the Lookup transformation.

Rules and guidelines for uncached lookups

Consider the following rules and guidelines when you configure uncached lookups :

- You can't use a lookup object as a query for uncached lookups.
- To configure an uncached lookup in the Lookup transformation, map at least one lookup port to the target.

CHAPTER 6

SQL transformation

You can configure an SQL transformation in a PostgreSQL mapping to process SQL queries and stored procedures in PostgreSQL.

When you add an SQL transformation to the mapping, on the **SQL** tab, you define the database connection and the type of SQL that the transformation processes.

You can use an SQL transformation to process the following types of SQL statements:

Stored procedure

You can configure an SQL transformation to call a stored procedure in PostgreSQL. The stored procedure must exist in the PostgreSQL database before you create the SQL transformation.

When the SQL transformation processes a stored procedure, it passes input parameters to the stored procedure. The stored procedure returns the values to the output fields of the transformation.

SQL Query

You can configure an SQL transformation to process an entered query that you define in the SQL editor. Do not use more than one SQL query in an SQL transformation.

The SQL transformation processes the query and returns the rows. The SQL transformation also returns any errors that occur from the underlying database or if there is an error in the user syntax.

For more information about SQL queries and stored procedures, see *Transformations* in the Data Integration documentation.

Rules and guidelines for SQL transformations

Consider the following rules and guidelines when you configure an SQL transformation in mappings:

General guidelines

- You can't use the saved query type in an SQL transformation.
- To enable the autocommit behavior in an SQL query and stored procedure, set the `EnableSqlTxAutoCommitSP` custom property value to **true** for the Secure Agent.

SQL query

- You can't validate an entered query that you define in the SQL editor.
- You can't use the following properties when you process an entered query:
  - Continue on SQL Error within row

- In-out and input parameters
- Stop on error

Stored procedure

- You can't use the same name for multiple stored procedures.
- Mappings fail when the stored procedure contains special characters or Unicode characters.
- You can't configure the **On Stored Procedure Error** advanced property in a mapping task.
- When you import data from PostgreSQL that contains the numeric data type, the Secure Agent imports the numeric data type with a default precision of 28 and scale of 26 irrespective of the precision you define for the numeric data type.
- When you import data from PostgreSQL that contains the char, varchar, text and bytea data types, the Secure Agent imports the data types with a precision of 32000 irrespective of the precision you define for the data types.
- If you add a new stored procedure to the database while you have the mapping open, the new stored procedure does not appear in the list of available stored procedures. To refresh the list, close and reopen the mapping.
- You can't use the bytea, time, boolean, citext, and timestamp with timezone data types in a stored procedure.

CHAPTER 7

Migrating a mapping

You can configure a connection and mapping in one environment and then migrate and run the mapping in another environment.

Data Integration uses the configured runtime attributes from the earlier environment and runs the mapping successfully in the new environment.

You can also migrate mappings configured in advanced mode. After the migration, you can change the connection properties from the Administrator service, but you do not need to modify the mapping.

Consider a scenario where you develop a mapping in the development organization (Org 1) and then you migrate and run the mapping in the production organization (Org 2). After you migrate, you might want to use the same or a different connection endpoint or object path in Org 2. Based on your requirement, follow the guidelines in this section before you plan the migration.

Use the same object path for the migrated mapping

If you want the migrated mapping in Org 2 to use the same object path as in Org 1, you must maintain the same schema and table in the PostgreSQL connection for Org 2.

For example, if you have two different connections, Connection1 used for Org 1 and Connection2 used for Org 2, the object path for the schema and table name must be the same in both the connections:

Connection1: SCHEMA1/TABLE1

Connection2: SCHEMA1/TABLE1

In this scenario, you do not need to override the schema and table in the advanced properties.

Use a different object path for the migrated mapping

After you migrate the mapping, you can use a different object path to run the mapping from the new environment.

In this scenario, before you migrate the mapping, you can change the object metadata or the runtime attributes to reflect the object path in the migrated environment. You do not have to edit or update the mapping in the new environment.

As a rule, when you specify the schema and table in the advanced properties or object properties, Data Integration honors the attributes in the following order of precedence:

1. **SQL override**. The SQL statement to override the default query generated from the specified source type in the Source or Lookup transformation in a mapping.
2. **Custom query**. The SQL statement to reduce the number of columns to query in the Source or Lookup transformation in a mapping.
3. **Runtime advanced attributes**. The advanced properties such as schema name and table name in the Source, Target, or Lookup transformation in a mapping.
4. **Connection attributes**.The schema attribute in the connection properties.
5. **Object metadata**. The object type selected in the Source, Target, or Lookup transformation in a mapping.

Migration options

When you migrate, you can choose from one of the following options to update the object path:

Option 1. Update the connection properties to reference the new object

When you import the mapping into Org 2, in the **Review Connections** section, you can change the existing connection to map to the connection that has access to the specified schema and table in Org 2.

Option 2. Override the properties from the advanced properties

Before the migration, specify the required schema and table name for the object from Org 2 in the advanced properties of the Org 1 mapping.

After the migration, when you run the mapping, the Secure Agent uses the configured advanced parameters to override the object specified in the mapping imported from Org 1.

Option 3. Parameterize the properties in the mapping

You can choose to parameterize the advanced attributes, such as the schema and table name before the migration. You can configure input parameters, in-out parameters, and parameter files in the mapping.

If you have used in-out parameters, you can change the schema and table attributes using the parameter file so that the changes are applied when the task runs.

You cannot override the input parameters from the parameter file in the advanced properties. You must use the in-out parameter to override the value from the parameter file.

Parameterizing only the advanced properties, but not the object in the mapping

If you want to parameterize only the advanced properties and use them at runtime, select a placeholder object in the object properties in the mapping and then specify an override to this placeholder object from the advanced properties. Ensure that the placeholder object contains the same metadata as the corresponding table that you specify as an override. When you run the mapping, the value specified in the advanced property overrides the placeholder object.

Parameterizing both the object and the advanced properties

If you want to keep both the PostgreSQL object type and the advanced fields parameterized, you must leave the **Allow parameter to be overridden at runtime** option unselected in the input parameter window while adding the parameters, and then select the required object at the task level. When you run the task, the values specified in the advanced properties take precedence.

Rules and guidelines for migrating a mapping

Consider the following rules and guidelines when you use the same or a different object path for the migrated mapping:

- The following table lists the transformation, object type, and the fields in the advanced properties of a mapping that you can retain when you migrate to the new environment:

Transformations	Object Type	Advanced Fields
Source	Single object	Schema and table name
Lookup	Single object **Note:** Applicable for unconnected, and connected cached and uncached.	
Target	Single object	

- You cannot dynamically refresh the data object schema at runtime. You must maintain the same metadata for the table selected in the source, target, or lookup transformations and the corresponding advanced field overrides as schema change handling is not applicable.
- You cannot migrate a mapping that uses the Hosted Agent as the runtime environment.
- If the mapping contains an advanced filter for the object in the query options section and an override to the object in the advanced properties, you must use the object that you specify as an override in the advanced properties to define the condition.
- You cannot override the schema and table name from the advanced properties if the mapping contains multiple objects.

CHAPTER 8

PostgreSQL SQL ELT optimization

You can use SQL ELT optimization to push the transformation logic to the PostgreSQL database.

When you use an ODBC connection, you can configure SQL ELT optimization to push transformation logic to PostgreSQL source or target databases. Use SQL ELT optimization when you use database resources to improve the performance of the task.

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

You cannot configure SQL ELT optimization for a mapping in advanced mode.

PostgreSQL Connector supports Full and Source SQL ELT optimization for the ODBC connection type that uses PostgreSQL ODBC drivers.

SQL ELT optimization functions and operators

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

Function	PostgreSQL
ABORT()	-
ABS()	X
ADD_TO_DATE()	X
AES_DECRYPT()	-
AES_ENCRYPT()	-
ASCII()	X
AVG()	X
CEIL()	X
CHOOSE()	-

Function	PostgreSQL
CHR()	X
CHRCODE()	-
COMPRESS()	-
CONCAT()	X
COS()	X
COSH()	-
COUNT()	X
CRC32()	-
CUME()	-
DATE_COMPARE()	-
DATE_DIFF()	X
DECODE()	X
DECODE_BASE64()	-
DECOMPRESS()	-
ENCODE_BASE64()	-
EXP()	X
FIRST()	-
FLOOR()	X
FV()	-
GET_DATE_PART()	-
GREATEST()	-
IIF()	X
IN()	SF
INDEXOF()	-
INITCAP()	X
INSTR()	-
IS_DATE()	-

Function	PostgreSQL
IS_NUMBER()	-
IS_SPACES()	-
ISNULL()	X
LAST()	-
LAST_DAY()	X
LEAST()	-
LENGTH()	X
LN()	X
LOG()	X
LOOKUP	X
LOWER()	X
LPAD()	X
LTRIM()	X
MAKE_DATE_TIME()	-
MAX()	X
MD5()	-
MEDIAN()	-
METAPHONE()	-
MIN()	X
MOD()	X
MOVINGAVG()	-
MOVINGSUM()	-
NPER()	-
PERCENTILE()	-
PMT()	-
POWER()	X
PV()	-

Function	PostgreSQL
RAND()	-
RATE()	-
REG_EXTRACT()	-
REG_MATCH()	-
REG_REPLACE	-
REPLACECHR()	-
REPLACESTR()	-
REVERSE()	-
ROUND(DATE)	-
ROUND(NUMBER)	X
RPAD()	X
RTRIM()	X
SET_DATE_PART()	-
SIGN()	X
SIN()	X
SINH()	-
SOUNDEX()	-
SQRT()	X
STDDEV()	X
SUBSTR()	X
SUM()	X
SYSDATE()	X
SYSTIMESTAMP()	X
TAN()	X
TANH()	-
TO_BIGINT	X
TO_CHAR(DATE)	X

Function	PostgreSQL
TO_CHAR(NUMBER)	X
TO_DATE()	X
TO_DECIMAL()	X
TO_FLOAT()	X
TO_INTEGER()	X
TRUNC(DATE)	-
TRUNC(NUMBER)	X
UPPER()	X
VARIANCE()	X

The following table lists the SQL ELT optimization operators that can be used in a PostgreSQL database:

Operator	SQL ELT optimization
+	Supported
-	Supported
*	Supported
/	Supported
%	Supported
\|\| | Supported |
> | Supported |
= | Supported |
>= | Supported |
<= | Supported |
!= | Supported |
AND | Supported |
OR | Supported |
NOT | Supported |
^= | Supported |

SQL ELT optimization operations and transformations

The following operations are supported for SQL ELT optimization:

- Insert
- Delete
- Update
- Date driven

The following transformations are supported for SQL ELT optimization:

- Filter
- Sorter
- Joiner
- Lookup
- Aggregator
- Union
- Expression

Configuring a PostgreSQL ODBC connection

You can set the SQL ELT optimization for the ODBC connection type that uses the PostgreSQL ODBC driver to enhance the mapping performance. To use an ODBC connection to connect to PostgreSQL, you must configure the ODBC connection.

After you create a PostgreSQL ODBC connection, navigate to the **SQL ELT Optimization** section, and then from the **SQL ELT Optimization** list, select **Full** or **To Source**. You cannot configure target-side SQL ELT optimization by using PostgreSQL ODBC driver. To verify that the SQL ELT optimization has taken place, you can check the session log for the job. In Monitor, view the log for jobs.

PostgreSQL supports PostgreSQL ODBC drivers on Windows and Linux systems. You must install the PostgreSQL ODBC 64-bit driver based on your system requirement.

Configuring a PostgreSQL ODBC connection on Windows

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

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

1. Download the PostgreSQL ODBC 64-bit driver.
2. Install the PostgreSQL ODBC driver on the machine where the Secure Agent is installed.
3. Open the folder in which ODBC data source file is installed.
4. Run the `odbcad32.exe` file.

   The **ODBC Data Source Administrator** dialog box appears.
5. Click **System DSN**.

The **System DSN** tab appears.

6. Click **Add**.

   The **Create New Data Source** dialog appears.

7. Select the PostgreSQL driver and click **Finish**.
8. Click **Configure**.

   The PostgreSQL Configuration Dialog appears.

9. Specify the required connection properties.
10. 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 a 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 64-bit driver.
2. Install the PostgreSQL ODBC driver on the machine where the Secure Agent is installed.
3. Configure the `odbc.ini` file properties with the required database details.

   For example, see the following sample ODBC.ini snippet:

   ```
   [ODBC_PostgreSQL]
   Driver=/root/ODBC_Drivers/DWpsql27.so
   Description=PostgreSQL DSN
   HostName=<hostname>
   PortNumber=5432
   Database=<databasename>
   ```

4. Run the following command to export the `odbc.ini` file:

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

5. Restart the Secure Agent.

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.

Create an ODBC connection

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

Perform the following steps to create a PostgreSQL ODBC connection on the **Connections** page:

1. In Administrator, click **Connections**.

   The Connections page appears.

2. Click **New Connection**.

   The **New Connection** page appears.

3. Configure the following connection details in the **Connection Details** section:

Property	Description
Connection Name	Name of the ODBC connection. For example, psql_odbc.
Description	Description of the connection.
Type	Type of the connection. Select the type of the connection as **ODBC**.

4. Configure the following connection details in the **ODBC Connection Properties** section:

Property	Description
Runtime Environment	Runtime environment that contains the Secure Agent you can use to access the system.
User Name	Username to log in to the PostgreSQL database.
Password	Password to log in to the PostgreSQL database.
Data Source Name	Enter the name of the ODBC data source name that you created for the PostgreSQL database.
Schema	Name of the PostgreSQL schema.
Code Page	The code page of the database server or flat file defined in the connection.
ODBC Subtype	Enter the value of the **ODBC Subtype** field as **PostgreSQL**.
Driver Manager for Linux	The driver that the PostgreSQL ODBC driver manager sends database calls to.

The PostgreSQL ODBC connection is created successfully.

Cross-schema SQL ELT optimization

You can use cross-schema SQL ELT optimization for a mapping task to read from or write data to PostgreSQL objects associated with different schemas within the same PostgreSQL database.

To use cross-schema SQL ELT optimization, create two PostgreSQL ODBC connections and specify the schema in each connection. Ensure that the schema in the source connection is different from the schema in the target connection, but both the schemas must belong to the same database. When you configure SQL ELT optimization for the mapping task, enable cross-schema SQL ELT optimization in the advanced session properties. By default, the check box is selected.

Configuring cross-schema SQL ELT optimization for PostgreSQL mapping

Create two PostgreSQL mappings. For example, perform the following steps to configure cross-schema SQL ELT optimization for a PostgreSQL mapping task:

1. Create the following two PostgreSQL ODBC connections, each defined with a different schema:
   a. Create an `psql_odbc1` PostgreSQL ODBC connection and specify `PSQL_SCHEMA1` schema in the connection properties.
   b. Create `psql_odbc2` PostgreSQL ODBC connection and specify `PSQL_SCHEMA2` schema in the connection properties.
2. Create a PostgreSQL mapping, `m_psql_pdo_acrossSchema`. Perform the following tasks:
   a. Add a Source transformation and include a PostgreSQL source object and connection `psql_odbc1` to read data using `PSQL_SCHEMA1`.
   b. Add a Target transformation and include a PostgreSQL target object and connection `psql_odbc2` to write data using `PSQL_SCHEMA2`.
3. Create a PostgreSQL mapping task, and perform the following tasks:
   a. Select the configured PostgreSQL mapping, `m_psql_pdo_acrossSchema`.
   b. On the **Schedule** tab, in the **SQL ELT Optimization** section, set the SQL ELT optimization value to **Full**.
   c. In the **Advanced Session Properties** section, select the **Enable cross-schema SQL ELT optimization** check box.
   d. Save the task and click **Finish**.

   When you run the mapping task, the Secure Agent reads data from the PostgreSQL source object associated with the `PSQL_SCHEMA1` schema and writes data to the PostgreSQL target object associated with `PSQL_SCHEMA2` schema.

CHAPTER 9

Data type reference

Data Integration uses the following data types in mappings and mapping tasks with PostgreSQL:

PostgreSQL native data types

PostgreSQL data types appear in the source and target transformations when you choose to edit metadata for the fields.

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 Secure Agent uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

When Data Integration reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When Data Integration 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 Data Integration 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
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

PostgreSQL Data Type	Transformation Data Type	Description
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	Text	Precision 104857600

PostgreSQL and transformation data types supported for create target at runtime option

The following table lists the transformation data types that Data Integration supports and the corresponding PostgreSQL data types when you use the **Create New at Runtime** option:

Transformation Data Type	PostgreSQL Data Type
Integer	Integer
Bigint	Bigint
Decimal	Numeric
Double	Double precision
String	Character varying
Text	Text

Transformation Data Type	PostgreSQL Data Type
Binary	Bytea
Date/Time	Timestamp

Rules and guidelines for data types

Consider the following rules and guidelines for PostgreSQL data types:

- The default precision for reading or writing string, text, or bytea data types is 32,000. If you configure a mapping to read or write data with the text, string, or bytea data types whose precision is more than 32,000, you must manually increase the precision of these data types in the mapping accordingly.
- 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