Informatica® PowerExchange for JDBC V2 10.5

User Guide

Informatica PowerExchange for JDBC V2 User Guide
10.5
March 2021

Publication Date: 2022-04-24

Table of Contents

Preface 5

Informatica Resources 5

- Informatica Network 5
- Informatica Knowledge Base 5
- Informatica Documentation 5
- Informatica Product Availability Matrices 6
- Informatica Velocity 6
- Informatica Marketplace 6
- Informatica Global Customer Support 6

Chapter 1: Introduction to PowerExchange for JDBC V2 7

PowerExchange for JDBC V2 Overview 7

Chapter 2: PowerExchange for JDBC V2 Configuration Overview 8

PowerExchange for JDBC V2 Configuration Overview 8

Prerequisites 8

Install and Configure the Type 4 JDBC Driver 8

Chapter 3: JDBC V2 Connections 10

JDBC V2 Connections Overview 10

JDBC V2 Connection Properties 10

Creating a JDBC V2 Connection 12

Chapter 4: PowerExchange for JDBC V2 Data Objects 13

JDBC V2 Object Overview 13

JDBC V2 Data Object Properties 13

JDBC V2 Object Read Operation 14

- JDBC V2 Object Read Operation Properties 14
- Source Properties of the Data Object Read Operation 14
- Output Properties of the Data Object Read Operation 15

JDBC V2 Data Object Write Operation Properties 18

- Input Properties of the Data Object Write Operation 18
- Target Properties of the Data Object Write Operation 20

Importing a JDBC V2 Data Object 21

Creating a JDBC V2 Target 21

Creating a JDBC V2 Object Read or Write Operation 23

Rules and Guidelines for JDBC V2 Data Objects 24

Chapter 5: JDBC V2 Mappings 25

Mapping Overview 25

Validation Environments. 25
Configure a Mapping. 26
Configuring a Mapping to Run in a Native or Non-Native Environment. 26
JDBC V2 Mapping Example. 27
Rules and Guidelines for JDBC V2 Mappings. 29

Chapter 6: PowerExchange for JDBC V2 Dynamic Mappings. 31

JDBC V2 Dynamic Mapping Overview. 31
Developing and Running Dynamic Mappings. 32
JDBC V2 Dynamic Mapping Example. 32
Rules and Guidelines for JDBC V2 Dynamic Mappings. 33

Chapter 7: JDBC V2 Lookup. 34

PowerExchange for JDBC V2 Lookup Overview. 34
General Properties. 35
Ports Properties. 35
Run-time Properties. 36
Lookup Properties. 36
Advanced Properties. 37
Adding a JDBC V2 Data Object Operation as a Lookup in a Mapping. 37

Chapter 8: JDBC V2 Run-Time Processing. 38

JDBC V2 Run-Time Processing Overview. 38
Join Expression. 38
Filter Expression. 38
    Native Expression. 39
    Platform Expression. 39
Parameterization for JDBC V2 Sources and Targets. 39
Partitioning. 40
    Key Range Partitioning. 40
    Dynamic Partitioning. 42
    Fixed Partitioning. 43
    Audits. 44

Chapter 9: JDBC V2 Datatype Reference. 45

Datatype Reference Overview. 45
JDBC V2 and transformation data types. 45

Index. 49

Preface

Use the *Informatica® PowerExchange® for JDBC V2 User Guide* to learn how to read from and write to Aurora PostgreSQL, Azure SQL Database, and databases with the Type 4 JDBC driver using the Developer tool. Learn to create a JDBC V2 connection, develop and run mappings in the native, Hadoop, or Databricks environment.

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 JDBC V2

This chapter includes the following topic:

- PowerExchange for JDBC V2 Overview, 7

PowerExchange for JDBC V2 Overview

You can use PowerExchange for JDBC V2 to read data from or write data to Aurora PostgreSQL, Azure SQL, SAP HANA, and other databases that support the Type 4 JDBC driver.

You can use JDBC V2 objects as sources and targets in mappings. When you use JDBC V2 objects in mappings, you must configure properties specific to the database that you want to connect to. You can validate and run mappings in native or non-native environments such as Spark and Databricks.

You use the JDBC V2 connection in mappings to connect to the SAP HANA database and read from or write to the HANA tables. You can also read from HANA modelling views, such as attribute, analytic, and calculation views. You can validate and run these mappings on the Spark or Databricks Spark engine.

The Data Integration Service uses the Type 4 JDBC driver to communicate with the database.

CHAPTER 2

PowerExchange for JDBC V2 Configuration Overview

This chapter includes the following topics:

- PowerExchange for JDBC V2 Configuration Overview, 8
- Prerequisites, 8
- Install and Configure the Type 4 JDBC Driver , 8

PowerExchange for JDBC V2 Configuration Overview

PowerExchange for JDBC V2 installs with the Informatica Services. You can enable PowerExchange for JDBC V2 with a license key.

Prerequisites

Before you can use PowerExchange for JDBC V2, perform the following tasks:

- Ensure that PowerExchange for JDBC V2 license is activated.
- Verify that you have write permissions on all the directories within the `<INFA_HOME>` directory.
- To run mappings on Hortonworks and Amazon EMR distributions that use non-Kerberos authentication, configure user impersonation.
  For information about configuring user impersonation, see the *Data Engineering Integration User Guide*.
- To run mappings on MapR secure clusters, configure the MapR secure clusters on all the nodes.
  For information about configuring MapR secure clusters, see the *Data Engineering Integration User Guide*.

Install and Configure the Type 4 JDBC Driver

Install and configure the database-specific Type 4 JDBC driver from a third-party vendor.

1. Download the Type 4 JDBC driver JAR files based on the database to which you want to connect.

Informatica includes certifications for the following drivers:

- SAP HANA Database: ngdbc-2.4.70.jar
- Aurora PostgreSQL: postgresql-42.2.6.jar
- Azure SQL Database: mssql-jdbc-7.2.2.jre8.jar

2. To connect to the JDBC V2 sources and targets in the native environment, perform the following tasks:
   a. To import metadata in the Developer tool, copy the JAR files to the following directory on the Developer tool machine:

      ```
      <Informatica installation directory>/clients/DeveloperClient/connectors/thirdparty/informatica.jdbc_v2/common
      ```

   b. To run mappings, copy the JAR files to the following directory on the machine where the Data Integration Service runs:

      ```
      <Informatica installation directory>/connectors/thirdparty/informatica.jdbc_v2/common
      ```

3. To connect to the JDBC V2 sources and targets in the non-native environment, perform the following tasks:
   a. To import metadata into the Developer tool, copy the JAR files to the following directory on the Developer tool machine:

      ```
      <Informatica installation directory>/clients/DeveloperClient/connectors/thirdparty/informatica.jdbc_v2/common
      ```

   b. To run mappings, copy the JAR files to the following directory on the machine where the Data Integration Service runs:

      ```
      <Informatica installation directory>/connectors/thirdparty/informatica.jdbc_v2/spark
      ```

4. Restart the Data Integration Service and the Developer tool.

CHAPTER 3

JDBC V2 Connections

This chapter includes the following topics:

- JDBC V2 Connections Overview, 10
- JDBC V2 Connection Properties, 10
- Creating a JDBC V2 Connection, 12

JDBC V2 Connections Overview

You can use the JDBC V2 connection to read data from or write data to any database that supports the Type 4 JDBC driver.

When you create a JDBC V2 connection, select the appropriate database type, **Aurora PostgreSQL** or **Azure SQL Database**, to which you want to connect to in the connection properties. Select **Other** to connect to any other database that supports the Type 4 JDBC driver.

You can use the connection to create data objects and run mappings. The Developer tool uses the connection when you create a data object. The Data Integration Service uses the connection when you run mappings.

You can create an connection from the Developer tool or the Administrator tool. The Developer tool stores connections in the domain configuration repository. Create and manage connections in the connection preferences.

You can use the connection in a mapping that runs on the native or non-native environment such as Spark and Databricks.

JDBC V2 Connection Properties

When you set up a JDBC V2 connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the JDBC V2 connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select JDBC V2.

The **Details** tab contains the connection attributes of the JDBC V2 connection. The following table describes the connection attributes:

Property	Description
Username	The database user name. User name with permissions to access the database that supports the Type 4 JDBC driver.
Password	The password for the database user name.
Schema Name	Optional. The schema name to connect in the database. If you do not specify the schema name, all the schemas available in the database are listed.

JDBC Driver Class Name	Name of the JDBC driver class.
	The following list provides the driver class name that you can enter for the applicable database type:
	- JDBC driver class name for Azure SQL Database:
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`
	- JDBC driver class name for Aurora PostgreSQL:
	`org.postgresql.Driver`
	- JDBC driver class name for SAP HANA Database:
	`com.sap.db.jdbc.Driver`
	For more information about which driver class to use with specific databases, see the third-party vendor documentation.

Connection String	Connection string to connect to the database.
	Use the following connection string:
	`jdbc:<subprotocol>:<subname>`
	The following list provides sample connection strings that you can enter for the applicable database type:
	- Connection string for Azure SQL Database JDBC driver:
	`jdbc:sqlserver://<host>:<port>;database=<database_name>`
	- Connection string for Aurora PostgreSQL JDBC driver:
	`jdbc:postgresql://<host>:<port>[/<database_name>]`
	- Connection string for SAP HANA Database driver:
	`jdbc:sap://<host>:<port>/?databaseName=<Database_Name>`
	For more information about the connection string to use with specific drivers, see the third-party vendor documentation.

Property	Description
Sub Type	The database type to which you want to connect. You can select from the following database types to connect: - **Azure SQL Database**. Connects to Azure SQL database. - **PostgreSQL**. Connects to Aurora PostgreSQL database. - **SAP HANA Database**. Connects to SAP HANA database. - **Others** . Connects to any database that supports the Type 4 JDBC driver.
Support Mixed-case Identifiers	Enable if the database uses case-sensitive identifiers. When enabled, the Data Integration Service encloses all identifiers within the character selected for the **SQL Identifier Character** property. For example, Aurora PostgreSQL database supports mixed-cased characters. You must enable this property to connect to the Aurora PostgreSQL database. When the **SQL Identifier Character** property is set to none, the **Support Mixed-case Identifiers** property is disabled.
SQL Identifier Character	Type of character that the database uses to enclose delimited identifiers in SQL queries. The available characters depend on the database type. Select (None) if the database uses regular identifiers. When the Data Integration Service generates SQL queries, the service does not place delimited characters around any identifiers. Select a character if the database uses delimited identifiers. When the Data Integration Service generates SQL queries, the service encloses delimited identifiers within this character. **Note:** Select **SQL Identifier Character** as **None** when you specify the SAP HANA Database subtype.

Creating a JDBC V2 Connection

Create a JDBC V2 connection before you create a JDBC V2 data object.

1. In the Developer tool, click **Window** > **Preferences**.
2. Select **Informatica** > **Connections**.
3. Expand the domain in the **Available Connections**.
4. Select the connection type **Database** > **JDBC V2**, and click **Add**.
5. Enter a connection name and an optional description.
6. Select JDBC V2 as the connection type.
7. Click **Next**.
8. Configure the connection properties.
9. Click **Test Connection** to verify the connection to the database that you want to connect to.
10. Click **Finish**.

CHAPTER 4

PowerExchange for JDBC V2 Data Objects

This chapter includes the following topics:

- JDBC V2 Object Overview, 13
- JDBC V2 Data Object Properties, 13
- JDBC V2 Object Read Operation, 14
- JDBC V2 Data Object Write Operation Properties, 18
- Importing a JDBC V2 Data Object, 21
- Creating a JDBC V2 Target, 21
- Creating a JDBC V2 Object Read or Write Operation, 23
- Rules and Guidelines for JDBC V2 Data Objects, 24

JDBC V2 Object Overview

A JDBC V2 data object is a physical data object that represents a table as a source or target from the database with Type 4 JDBC driver. A JDBC V2 data object is the representation of data that is based on the object from the database to which you connect. You can configure the data object read and write operation properties that determine how the Data Integration Service reads from or writes data to the specified database.

To read data, create a data object read operation based on the JDBC V2 data object. Configure the read operation properties to determine how the Data Integration Service must read data. Add the read operation as a Read transformation in a mapping.

To write data, create a data object write operation based on the JDBC V2 data object. Configure the write operation properties to determine how the Data Integration Service must write data. Add the write operation as a Write transformation in a mapping.

JDBC V2 Data Object Properties

The JDBC V2 Overview view displays general information about the JDBC V2 data object and the object properties that apply to the table you import.

General Properties

You can configure the following properties for a JDBC V2 data object:

- Name. Name of the JDBC V2 data object.
- Description. Description of the JDBC V2 data object.
- Connection. Name of the JDBC V2 connection.

JDBC V2 Object Read Operation

The Data Integration Service reads data from the JDBC V2 table based on the data object read operation properties that you specify.

When you create a data object read operation, the Developer tool creates a Source transformation and an Output transformation.

The Source transformation represents the data that the Data Integration Service reads from the JDBC V2 table.

The Output transformation represents the data that the Data Integration Service passes into the mapping pipeline.

JDBC V2 Object Read Operation Properties

The Data Integration Service reads data from a JDBC V2 object based on the data object read operation. The Developer tool displays the data object read operation properties of the JDBC V2 data object in the Data Object Operation view.

You can view or configure the data object read operation from the source and output properties.

Source properties

Represents data that the Data Integration Service reads from the JDBC V2 object. Select the source properties to view the general and column properties of the JDBC V2 object.

Output properties

Represents data that the Data Integration Service passes into the mapping pipeline. Select the output properties to edit the port, query, runtime, and advanced properties of the data object read operation.

Source Properties of the Data Object Read Operation

When you create a data object, the source properties populate based on the JDBC V2 object that you add. The source properties of the data object read operation include general and column properties that apply to the JDBC V2 object.

You can view the source properties of the data object read operation from the **General** and **Column** tabs.

General Properties

Represents data that the Data Integration Service reads from the JDBC V2 object. Select the source properties to view the general and column properties of the JDBC V2 object.

The following table describes the source general properties of the data object read operation:

Property	Description
Name	Name of the JDBC V2 source object.
Description	Description of the data object read operation.
Physical Name	The physical name of the source object. For example, `Customer`.
Path Information	The path to which the source object belongs. For example, the path for the source object is `/public/TABLE/Customer`.

Column Properties

The column properties display the data types, precision, and scale of the source property in the data object read operation.

The following table describes the source column properties of the data object read operation:

Property	Description
Name	Name of the column.
Native Name	The name of the attribute in the database server.
Type	Native data type of the column.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Partition Key	The port or set of ports that you define as the partition key.
Access Type	Indicates whether the field has read and write permissions.
Description	Description of the column.

Output Properties of the Data Object Read Operation

The output properties represent data that the Data Integration Service passes into the mapping pipeline. Select the output properties to edit the port properties of the data object read operation.

The output properties of the data object read operation include general properties that apply to the data object operation. The output properties also include port, source, query, runtime, and advanced properties that apply to the JDBC V2 object.

You can view and change the output properties of the data object read operation from the **General**, **Ports**, **Sources**, **Query**, **Runtime**, and **Advanced** tabs.

General Properties

The following table describes the source general properties of the data object read operation:

Property	Description
Name	Name of the JDBC V2 source object.
Description	Description of the data object read operation.

Ports Properties

The column properties display the data types, precision, and scale of the source property in the data object read operation.

The following table describes the source column properties of the data object read operation:

Property	Description
Name	Name of the column.
Type	Native data type of the column.
Precision	Maximum number of significant digits for Numeric data types, or maximum number of characters for String data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Description	Description of the column.

Sources Properties

The sources properties list the JDBC V2 objects used in the data object read operation.

Query Properties

Use the query property to select specific records from the database that you are connected to.
The following table describes the query properties that you configure for a data object read operation:

Property	Description
Join	The Join expression in a read operation to join data from multiple tables. Select more than one source in the **Data Object** tab and define the join condition in the **Query** tab.
Filter	The Filter value in a read operation. The filter specifies the where clause of select statement. Use a filter to reduce the number of rows that the Data Integration Service reads from the source. When you enter a source filter, the Developer tool adds a WHERE clause to the default query. You can use the Native or Platform expression to select specific records.

Run-time Properties

The run-time properties displays the name of the connection used for the data object read operation.

The following table describes the run-time properties that you configure for the JDBC V2 source:

Property	Description
Connection	Name of the JDBC V2 connection.
Partition	Define the partition type to read data from the JDBC V2 data object. You can select one of the partitioning types: - **None**. Partitioning is disabled. By default, the Data Integration Service creates a single partition. Applicable for mappings in the native and non-native environment. - **Fixed**. The Data Integration Service distributes rows of data based on the number of partitions you specify. Applicable for mappings in the non-native environment. - **Key Range**. The Data Integration Service distributes rows of data based on a port or set of ports that you define as the partition key. You can define a range of values for each port. The Data Integration Service uses the key and ranges to send rows to the appropriate partition. Applicable for mappings in the native environment.

Advanced Properties

Use the advanced properties to specify the data object read operation properties to read data from JDBC V2 objects.

The following table describes the advanced properties that you configure in the data object read operation:

Property	Description
Operation Type	The read operation for the JDBC V2 data object.
Pre SQL	The SQL query that the Data Integration Service runs before reading data from the source.
Post SQL	The SQL query that the Data Integration Service runs after reading data from the source. Applicable only in the native environment.
Fetch Size	The number of rows that the Data Integration Service fetches from the database in a single call.
Table Name	Overrides the table name used in the metadata import with the table name you specify.
Schema Name	Overrides the schema name of the source object. If you specify the schema name both in the connection and the source read operation properties, the Data Integration Service considers the schema name specified in the source properties.

JDBC V2 Data Object Write Operation Properties

The Data Integration Service writes data to a JDBC V2 object based on the data object write operation. The Developer tool displays the data object write operation properties for the JDBC V2 data object in the Data Object Operation section.

You can view the data object write operation from the Input and Target properties.

Input properties

Represent data that the Data Integration Service reads from a JDBC V2 object. Select the input properties to edit the port, target, runtime, and specify the advanced properties of the data object write operation.

Target properties

Represent data that the Data Integration Service writes to the JDBC V2 target. Select the target properties to view data, such as the name and description of the JDBC V2 object.

Input Properties of the Data Object Write Operation

Input properties represent data that the Data Integration Service writes to a JDBC V2 object. Select the input properties to edit the port, properties of the data object write operation. You can also specify advanced data object write operation properties to write data to JDBC V2 objects.

The input properties of the data object write operation include general properties that apply to the data object write operation. Input properties also include port, source, and advanced properties that apply to the data object write operation.

You can view and change the input properties of the data object write operation from the General, Ports, Targets, Run-time, and Advanced tabs.

General Properties

The general properties list the name and description of the data object write operation.

Ports Properties

The input ports properties list the data types, precision, and scale of the data object write operation.

The following table describes the input ports properties that you must configure in the data object write operation:

Property	Description
Name	The name of the port.
Type	The data type of the port.
Precision	The maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Detail	The detail of the data type.

Property	Description
Scale	The maximum number of digits after the decimal point for numeric values.
Description	The description of the port.

Target Properties

The Target properties list the JDBC V2 object in the data object write operation.

Run-time Properties

The run-time properties display the name of the connection used in the write operation.

The following table describes the run-time properties that you configure for a JDBC V2 write operation:

Property	Description
Connection	Name of the JDBC V2 connection.
Partition Type	Type of the partition used. The partition type determines how the Data Integration Service redistributes data across partition points. You can select one of the following partition types: - **None**. Partitioning is disabled. By default, the Data Integration Service creates a single partition. Applicable for mappings in the native and non-native environment. - **Dynamic**. The Data Integration Service determines the number of partitions that it must create at run time based on the number of partitions you specified at the source. - **Fixed**. The Data Integration Service creates the partitions at runtime based on the number of partitions you specify. Applicable for mappings in the non-native environment.

Advanced Properties

You can use the advanced properties to specify data object write operation properties to write data to a JDBC V2 server.

The following table describes the advanced properties that you configure for a JDBC V2 write operation:

Property	Description
Operation Type	The write operation for the JDBC V2 data object.
Pre SQL	The SQL statement to run before writing data to the target.
Post SQL	The SQL statement to run after writing data to the target.
Truncate Target	Truncates the target table before inserting records to the target.
Reject Truncated/ Overflow Rows	Writes truncated and overflow data to the reject file. If you select Reject Truncated/Overflow Rows, the Data Integration Service sends all truncated rows and any overflow rows to the reject file.
Table Name	Overrides the table name used in the metadata import with the table name you specify.

Property	Description
Schema Name	Overrides the schema name of the target object. If you specify the schema name both in the connection and the target write operation properties, the Data Integration Service considers the schema name specified in the target write operation properties.
UpdateMode	Not applicable.

Target Properties of the Data Object Write Operation

The target properties represent the data that is used to populate the JDBC V2 data object that you added when you created the data object. The target properties of the data object write operation include general and column properties that apply to the JDBC V2 objects. You can view the target properties of the data object write operation from the **General** and **Column** tabs.

General Properties

The general properties display the name and description of the JDBC V2 objects.

The following table describes the target general properties of the data object write operation:

Property	Description
Name	Name of the JDBC V2 target object.
Description	Description of the data object write operation.
Physical Name	The physical name of the target object. Fro example, `Customer`.
Path Information	The path to which the target object belongs. For example, the path for the source object is `/public/TABLE/Customer`.

Column Properties

The column properties display the data types, precision, and scale of the target property in the data object write operation.

You can view the following target column properties of the data object write operation:

Property	Description
Name	Name of the column
Native Name	The native name of the JDBC V2 data object.
Type	Native data type of the column property
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values

Property	Description
Partition Key	The port or set of ports that you define as the partition key.
Access Type	The access type of the port or column.
Description	Description of the column property

Importing a JDBC V2 Data Object

Import a JDBC V2 data object to add to a mapping.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Object**.
3. Select **JDBC V2 Data Object** and click **Next**.

   The **JDBC V2 Data Object** dialog box appears.
4. Enter a name for the data object.
5. Click **Browse** next to the **Location** option, and select the project or folder.
6. Click **Browse** next to the **Connection** option, and select the JDBC V2 connection from which you want to import the JDBC V2 resource metadata.
7. To add a resource, click **Add** next to the **Selected Resources** option.

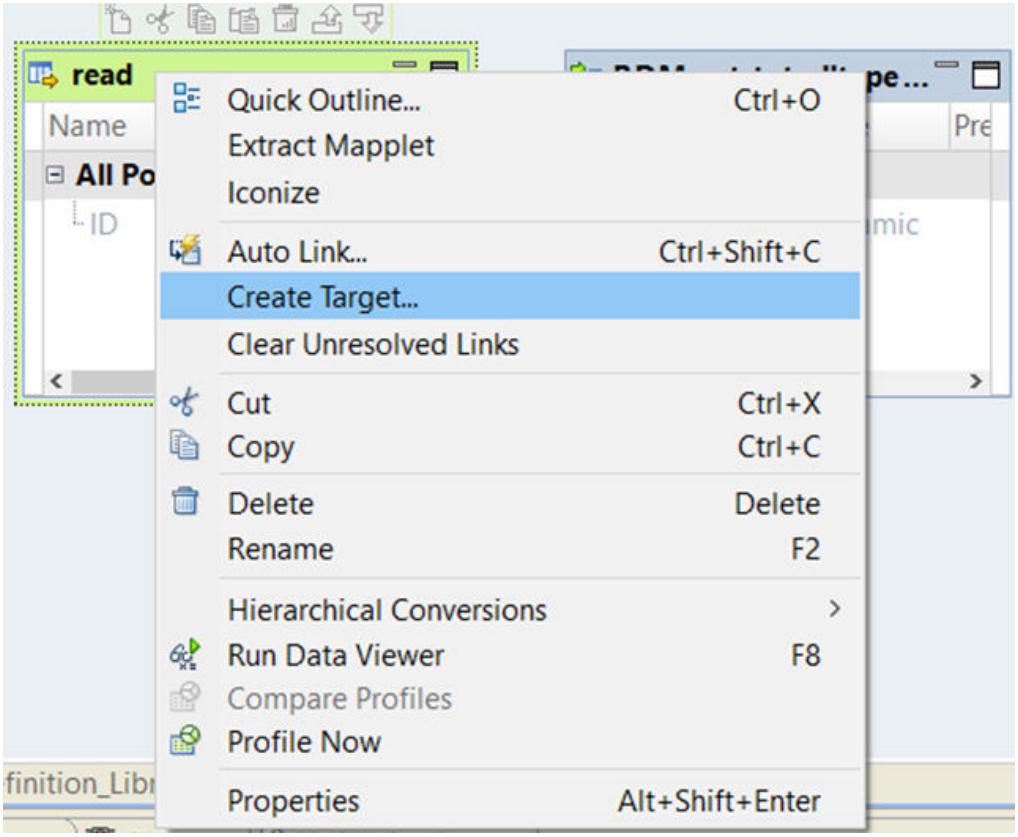
   The **Add Resource** dialog box appears.
8. From the Package Explorer, select the table, view, or synonym from which you want to import the schema.
9. You can perform one of the following tasks to import an JDBC V2 table, and then click **OK**:
   - Navigate to the JDBC V2 table that you want to import.
   - Search for the JDBC V2 table, enter the name of the JDBC V2 table entity that you want to add, and click **OK**.
10. Click **Finish**.
    The data object appears under Data Objects in the project or folder in the **Object Explorer** view.

Creating a JDBC V2 Target

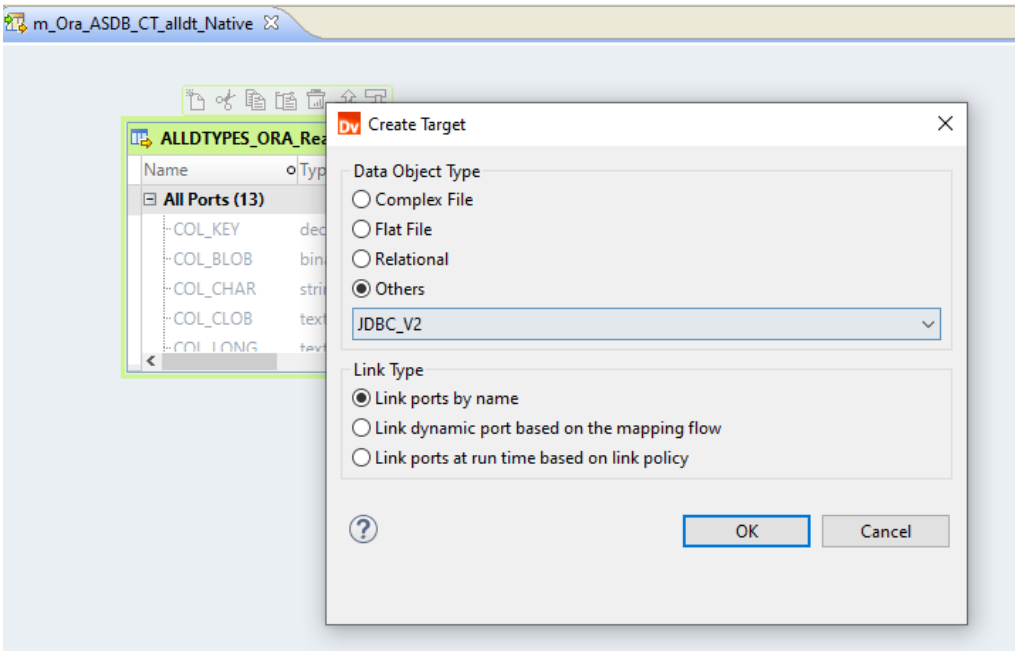
You can create a JDBC V2 target using the **Create Target** option.

1. Select a project or folder in the **Object Explorer** view.
2. Select a source or a transformation in the mapping.
3. Right-click the Source transformation and select **Create Target**.
   The **Create Target** dialog box appears.

The following image shows the **Create Target** option:

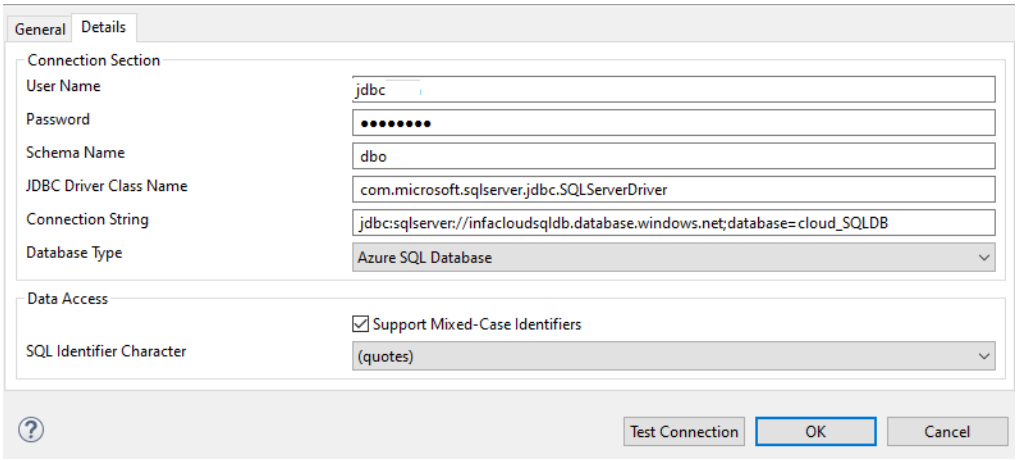


4. Select **Others** and then select **JDBC V2** data object from the list in the **Data Object Type** section.
The following image shows the **Data Object Type** section:

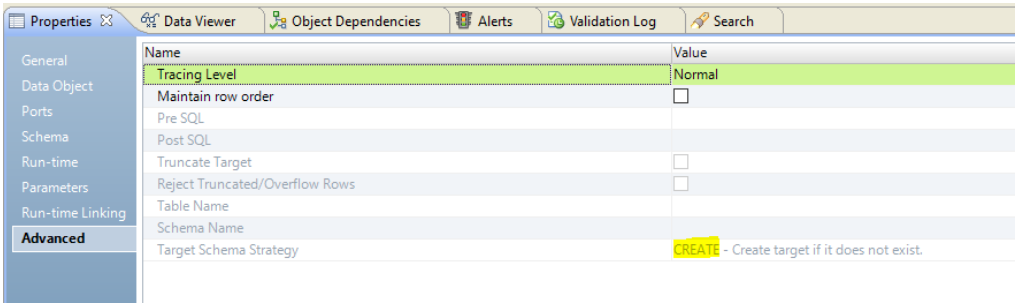


5. Click **OK**.
The **New JDBC V2 Data Object** dialog box appears.

6. Enter a name for the data object.
7. Click **Finish**.
   The new target appears under the **Physical Data Objects** category in the project or folder in the **Object Explorer** view.
8. In the JDBC V2 connection **Details** tab, verify the connection details.
   The following image shows the connection properties:



9. In the JDBC V2 advanced target properties, select the target schema strategy as **Create - Create target if it does not exist**.
   The following image shows the configured advanced properties:



Creating a JDBC V2 Object Read or Write Operation

You can add a JDBC V2 data object read or write operation to a mapping as a source. You can create the data object read or write operation for one or more JDBC V2 data objects.

Before you create a JDBC V2 data object read or write operation, you must create at least one JDBC V2 data object.

1. Select the data object in the Object Explorer view.
2. Right-click and select **New** > **Data Object Operation**.

   The **Data Object Operation** dialog box appears.
3. Enter a name for the data object read or write operation.
4. In the **Capabilities** field, select **tableRead** or **tableWrite** as the type of data object operation.

5. Click **Add**.

   The **Select Resources** dialog box appears.

6. Select the JDBC V2 object for which you want to create the data object read or write operation, and click **OK**.
7. Click **Finish**.

The Developer tool creates the data object read or write operation for the selected JDBC V2 data object.

Rules and Guidelines for JDBC V2 Data Objects

When you use the JDBC V2 connection, some rules and guidelines apply while you import data or configure queries.

Consider the following rules and guidelines for JDBC V2 data objects:

- When you import the JDBC V2 object metadata, you cannot import synonyms.
- You cannot select the JDBC V2 connection from the Connection Explorer.
- When you add resources while importing the JDBC V2 data object, the advanced search is disabled. Type the table name in the **Name** field in the **Add Resources** dialog box to search the table.
- If the schema of the tables is modified in the JDBC V2 database, you cannot synchronize the data objects. The synchronize option for the data object in the object explorer is disabled.
- Insert and truncate queries configured for data objects in mappings that run on the Spark engine are not logged in the Spark logs.

CHAPTER 5

JDBC V2 Mappings

This chapter includes the following topics:

- Mapping Overview, 25
- Validation Environments, 25
- Configure a Mapping, 26
- Configuring a Mapping to Run in a Native or Non-Native Environment, 26
- JDBC V2 Mapping Example, 27
- Rules and Guidelines for JDBC V2 Mappings, 29

Mapping Overview

When you run a mapping, you can choose to run the mapping in the native environment or in a non-native environment, such as Hadoop or Databricks. Configure the run-time environment in the Developer tool to optimize mapping performance and process data that is greater than 10 terabytes.

When you run mappings in the native environment, the Data Integration Service processes and runs the mapping. When you run mappings in a non-native environment, the Data Integration Service pushes the processing to a compute cluster, such as the Spark engine on Hadoop or to Databricks.

Validation Environments

The properties in the **Validation Environments** indicate whether the Developer tool validates the mapping definition for the native or non-native execution environment.

You can configure the following properties for the **Validation Environments**:

Native

Default environment. The Data Integration Service runs the mapping in a native environment.

Hadoop

Run the mapping in the Hadoop environment. The Data Integration Service pushes the transformation logic to the Hadoop cluster through a Hadoop connection. Select the Spark engine to process the mapping.

Databricks

Run the mapping in the Databricks environment. The Data Integration Service pushes the transformation logic to the Databricks cluster through a Databricks connection. The Databricks cluster processes the mapping on the Databricks Spark engine.

When you validate the mapping, validation occurs for the engine that you choose in the **Validation Environments**.

Configure a Mapping

Create a mapping to move data between sources and targets and transform the data.

1. Select a project or folder in the Object Explorer view.
2. Click **File** > **New** > **Mapping**.
3. Enter a mapping name.
4. Click **Finish**.

   An empty mapping appears in the editor.
5. Adding the configured objects to the mapping.
   a. Drag a data object to the editor and select Read to add the data object as a source.
   b. Drag a data object to the editor and select Write to add the data object as a target.
6. Connect the mapping objects through the ports.
7. Select **Native**, **Hadoop**, or **Databricks** as the value for the validation and execution environment.
8. Validate and run the mapping.

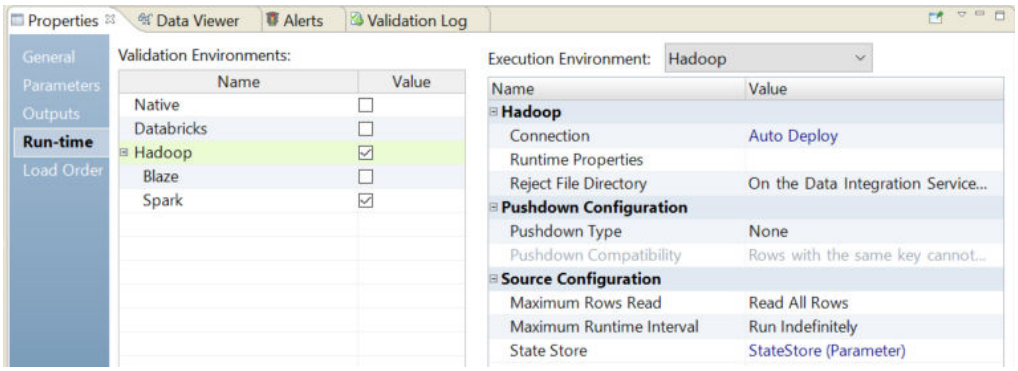
Configuring a Mapping to Run in a Native or Non-Native Environment

You can configure a mapping to run in a native or non-native environment. To configure a mapping, you must select a validation environment and an execution environment.

1. Select a mapping from a project or folder from the **Object Explorer** view to open in the editor.
2. In the **Properties** view, select the **Run-time** tab.
3. Select **Native**, **Hadoop**, or **Databricks** as the value for the validation environment.

   When you select the Hadoop environment, select the Spark engine. Disable the engines that you do not want to use.
4. Select **Native**, **Hadoop**, or **Databricks** for the execution environment.

   The following image displays the validation environments that you can select:



5. Select **Connection** and use the drop down in the value field to browse for a connection or to create a connection parameter:
   - To select a connection, click **Browse** and select a connection.
   - To create a connection parameter, click **Assign Parameter**.
6. Configure the rest of the properties for the execution environment.
7. Right-click an empty area in the editor and click **Validate**.

   The Developer tool validates the mapping.
8. View validation errors on the **Validation Log** tab.
9. To view the execution plan for a mapping that runs in the non-native environment, right-click the editor and select **Show Execution Plan**.

JDBC V2 Mapping Example

You work in the Human Resources department and you manage employee information. You want to filter the employee details from the engineering department and write the results to a flat file.

You can use the following objects in the mapping:

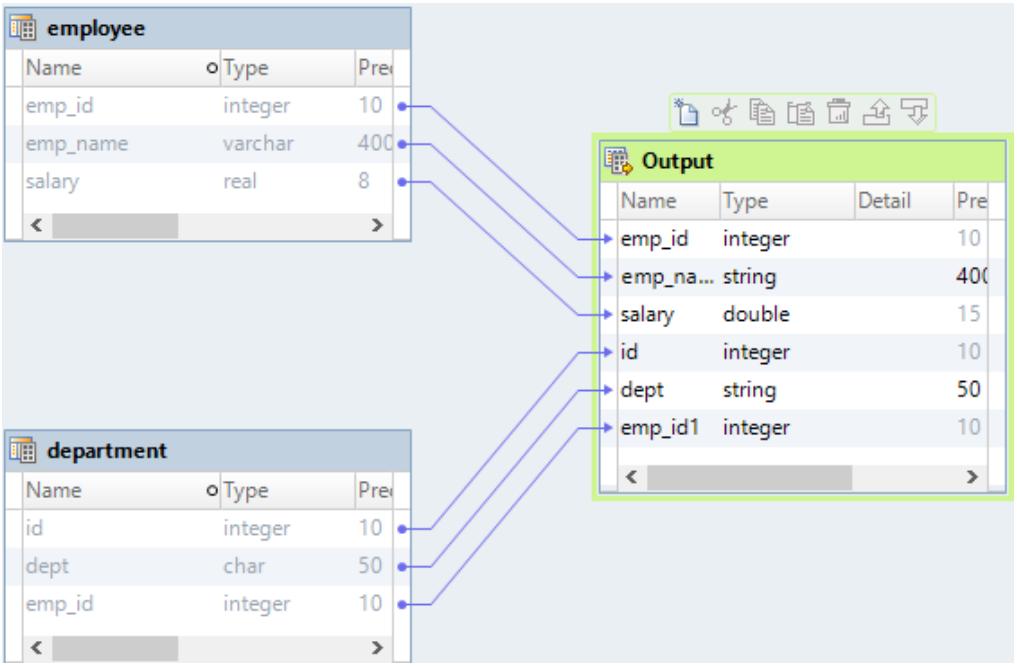
Aurora PostgreSQL Input

The input file is an Aurora PostgreSQL table that contains the employee details from all the departments in the organization.

Create an Aurora PostgreSQL data object read operation. Configure the Aurora PostgreSQL connection and specify the table that contains the employee data as a source for the data object. Drag the source data object into the mapping and specify the data object access as read.

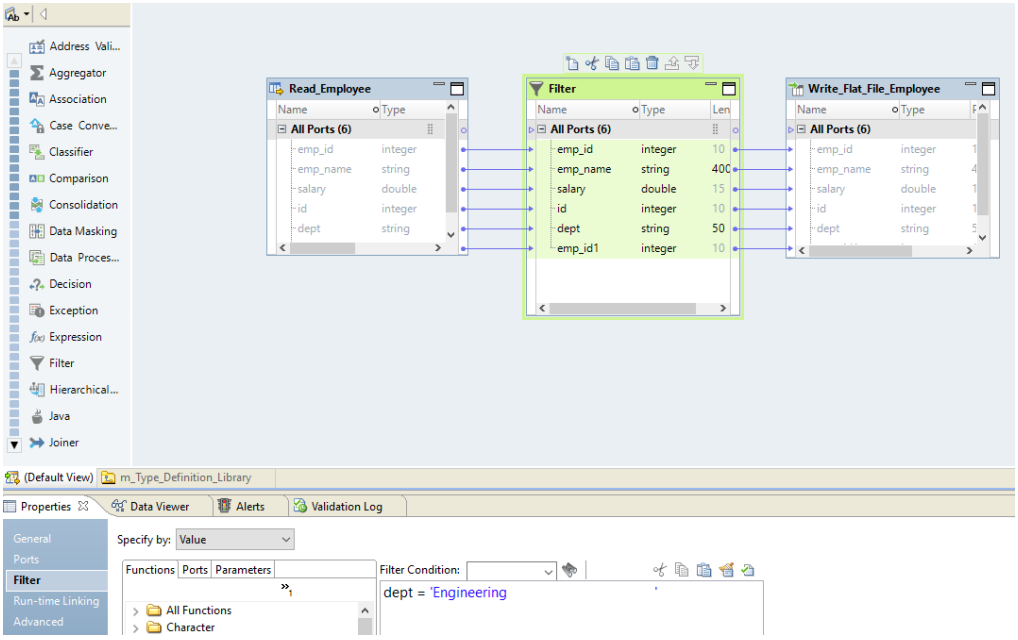
The input files are two Aurora PostgreSQL tables, one containing the employee details and the other containing the department details from the entire organization. Use the Join query to get data from the

employee and the department tables.



Transformations

Add a Filter transformation to filter details of employees from the `Engineering` department.



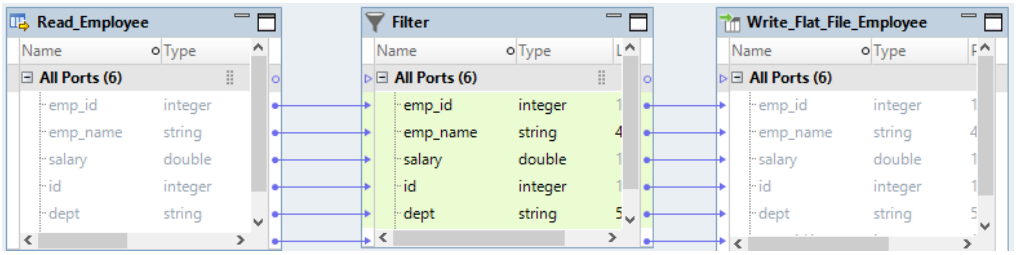
The Filter transformation filters the source data based on the value you specify for the `Dept` column. The Data Integration Service returns the rows that meet the filter condition.

Flat File Output

Create a flat file data object write operation. Configure the flat file connection and specify the flat file object as the target data object. Drag the target data object into the mapping and specify the data object access as write.

The output file is a flat file that contains the details of the employees working in the Engineering department.

The following image shows the mapping example:



When you run the mapping, the Data Integration Service reads the employee records from the Aurora PostgreSQL table, filters out the Engineering employee details, and writes that data to a flat file.

Rules and Guidelines for JDBC V2 Mappings

Consider the following rules and guidelines when you configure a JDBC V2 mapping:

General guidelines

Consider the following general guidelines for JDBC V2 mappings:

- You cannot read from or write to tables where the table name exceeds 128 characters. The Integration Service fails to validate the data and an error occurs.
- When you specify a native filter, enclose the table name and column name in the filter condition in quotes. For example, `"Supplier".s_suppkey<=5`
- When you run a JDBC V2 mapping to read or write data that contains time(4), time(5), and time(6) data types, the data is truncated beyond precision 3.
  When you import multiple source objects, ensure that the table and column names do not contain Unicode characters.

SAP HANA Database mappings

In a mapping that uses the JDBC V2 connection with the SAP HANA Database subtype, consider the following general guidelines:

- You cannot run mappings to read from or write to SAP HANA Database in the native environment.
- SAP HANA Database field names with the aggregate function do not display correctly in the Designer.
- Do not map the parameter field to the target in the mapping.
  Specify multiple values for a parameter separated by a comma. For example, "Package_Joe/CALC_HANDSON"."EMPNAME" = 'Alex' ,"Package_Joe/CALC_HANDSON"."PARAM_M_INPUT_PARAM_HANDSON" = '5000';
- When you configure a filter, use only the native expression.
- You cannot create dynamic mappings and use the **Create Target** option for SAP HANA Database.

Create Target Option

When you configure a JDBC V2 target in a mapping using the **Create Target** option, consider the following guidelines:

- When you run a mapping enabled to create a new Azure SQL Database target at runtime and the source data contains the Time data type, the Integration Service writes the date time value only until microseconds.
- By default, the Target Schema Strategy is set to **RETAIN - Load into existing table schema**. You must manually select the **Target Schema Strategy** as **CREATE - Create target if it does not exist** in the Advanced properties in a Write operation for the **Create Target** option to work.
- The CREATE TABLE DDL query that generates in the ANSI SQL-92 generic data type format might not run on all databases as the target database might not support the data type or data length. In this case, you must create the table manually and use it as the target in the mapping.
- The option to edit the target metadata is subject to the target database support. When unsupported, you can see an error or warning message in the session logs.
- Ensure to set the correct field precision and scale depending on the type of JDBC V2 database that you want to access.
- When the database subtype used in the JDBC V2 connection is Others, you can write data that contains the BigInt data type only if that database supports the Numeric data type.
- When you write to a PostgreSQL target and the source data contains the Time data type, the mapping runs successfully but the time value in microseconds is truncated to milliseconds.
- When you write data that contains the Float, Double, Number, and Real data types to an Azure SQL Database target, the mapping fails. You must manually change the data type in the physical data object.
- The Date and Time data type is converted to the Timestamp data type.

CHAPTER 6

PowerExchange for JDBC V2 Dynamic Mappings

This chapter includes the following topics:

- JDBC V2 Dynamic Mapping Overview, 31
- Developing and Running Dynamic Mappings, 32
- JDBC V2 Dynamic Mapping Example, 32
- Rules and Guidelines for JDBC V2 Dynamic Mappings, 33

JDBC V2 Dynamic Mapping Overview

You can use JDBC V2 data objects as dynamic sources and targets in a mapping both in the native and the non-native environment.

Use the JDBC V2 dynamic mapping to accommodate changes to source, target, and transformation logics at run time. You can use a JDBC V2 dynamic mapping to manage frequent schema or metadata changes or to reuse the mapping logic for data sources with different schemas. Configure rules, parameters, and general transformation properties to create the dynamic mapping.

If the data source for a source or target changes, you can configure a mapping to dynamically get metadata changes at runtime. If a source changes, you can configure the Read transformation to accommodate changes. If a target changes, you can configure the Write transformation accommodate target changes.

You do not need to manually synchronize the data object and update each transformation before you run the mapping again. The Data Integration Service dynamically determine transformation ports, transformation logic in the ports, and the port links within the mapping.

There are the two options available to enable a mapping to run dynamically. You can select one of the following options to enable dynamic mapping:

- In the **Data Object** tab of the data object read or write operation, select the **At runtime, get data object columns from data source** option when you create a mapping.
  When you enable the dynamic mapping using this option, you can refresh the source and target schemas at runtime.
- In the **Ports** tab of the data object write operation, select the value of the **Columns defined by** property as **Mapping Flow** when you configure the data object write operation properties.

For information about dynamic mappings, see the *Informatica Developer Mapping Guide*.

Developing and Running Dynamic Mappings

Perform the following tasks to develop and run a dynamic mapping to read or write to JDBC V2. The tasks and the order in which you perform the tasks depend on the mapping scenario and the transformations that you plan to use in the mapping.

1. Create a JDBC V2 mapping and add the JDBC V2 objects.
2. Configure a JDBC V2 dynamic source for the Read or Lookup transformation to get metadata changes from the JDBC V2 source at run time. Select the JDBC V2 source object and perform one of the following tasks based on your requirement:
   - Use a parameter as a source for a dynamic mapping source object.
   - Configure data sources for source objects in mappings to get metadata changes at run time. To dynamically get columns from the data source file at run time, select **At run time, get data object columns from the data source**.
3. Create dynamic ports in transformations and link ports.
4. Define input rules for dynamic ports to determine which generated ports to create.
5. Configure a Write transformation to write to a JDBC V2 dynamic target. Select the JDBC V2 target object and perform one of the following tasks based on your requirement:
   - Use a parameter as the data object for the transformation and then change the parameter at run time.
   - To dynamically get data object columns from the data source at run-time, enable the option **At run time, get data object columns from the data source**.
   - Define target object columns by mapping flow to enable upstream mapping objects to update the incoming ports for the Write transformation.
     To do this, select **Columns defined by: Mapping flow** in the **Ports** tab of the Properties view and then select **Create or replace table at run time** in the **Target Schema Strategy** list.
6. Create and configure a run-time link to determine which ports to link at run time.
7. Validate the mapping.
8. Compile and run the dynamic mapping.

JDBC V2 Dynamic Mapping Example

Your organization has a large amount of data that keeps changing. Your organization needs to incorporate all the updated data in a short span of time. Create a dynamic mapping, where you can refresh the source schema dynamically to fetch the updated data. Add all the dynamic ports to the target to override the metadata of the existing ports.

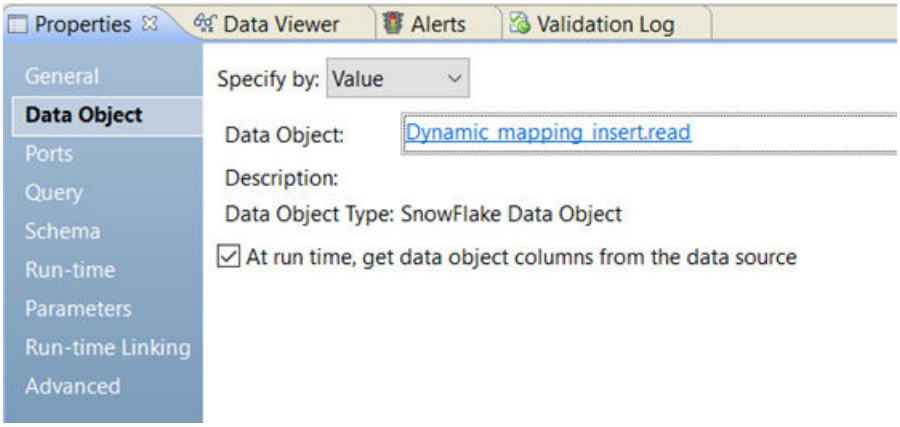
1. Import the JDBC V2 read and write data objects.
2. Select a project or folder in the **Object Explorer** view.
3. Click **File** > **New** > **Mapping**.

   The **Mapping** dialog box appears.
4. Enter the name of the mapping in the **Name** field.
5. Click **Finish**.
6. Drag the data object into a mapping.

The **JDBC V2 Data Object Access** dialog box appears.

7. Select the **Read** option and click **OK**.
8. In the **Data Object** tab, select the **At runtime, get data object columns from the data source** check box.

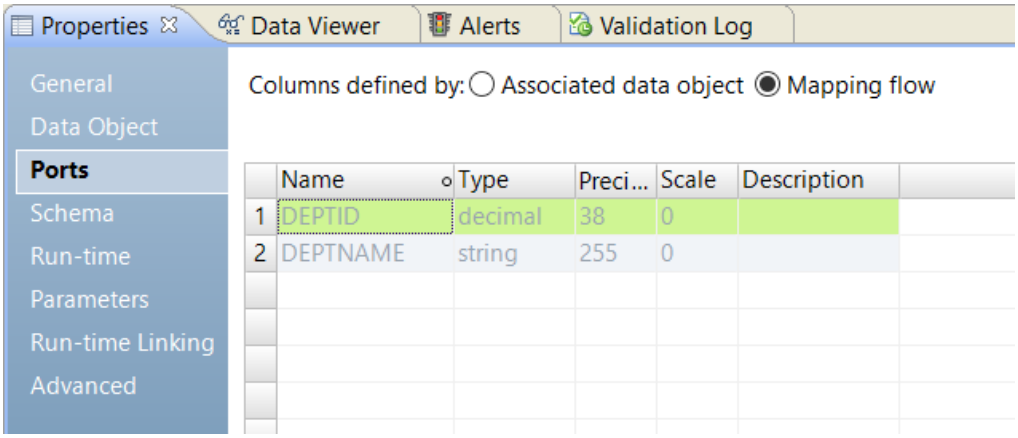
   The following image shows the **Data Object** tab:



9. Drag the data object into a mapping.

   The **JDBC V2 Data Object Acess** dialog box appears.
10. Select the **Write** option and click **OK**.
11. In the **Ports** tab, select the value of the **Columns defined by** as **Mapping Flow**.

    The following image shows the **Ports** tab:



12. Select all the source incoming ports and add the ports to the target.
13. Save and run the mapping.

Rules and Guidelines for JDBC V2 Dynamic Mappings

Consider the following rules and guidelines when you configure dynamic mappings:

- When the data types contain more than 4K characters and the default precision is set to 4K, the mapping fails. You cannot change the precision at the object level. You can set the precision when you create the table at the backend.

CHAPTER 7

JDBC V2 Lookup

This chapter includes the following topics:

- PowerExchange for JDBC V2 Lookup Overview, 34
- General Properties, 35
- Ports Properties, 35
- Run-time Properties, 36
- Lookup Properties, 36
- Advanced Properties, 37
- Adding a JDBC V2 Data Object Operation as a Lookup in a Mapping, 37

PowerExchange for JDBC V2 Lookup Overview

You can use a JDBC data object read operation to look up data in a JDBC V2 table.

You can add a JDBC V2 data object read operation as a lookup in a mapping. You can then configure a lookup condition to look up data from the JDBC V2 table.

You can configure a cached lookup operation to cache the lookup data in a mapping that runs on the Spark engine.

When you enable lookup caching, the Data Integration Service caches the lookup values. The Data Integration Service queries the lookup source once, caches the values, and looks up values in the cache. Caching the lookup values can increase performance on large lookup tables.

When you disable caching, the Data Integration Service does not cache the lookup values. The Data Integration Service queries the lookup source instead of building and querying the lookup cache. Each time a row passes, the Data Integration Service issues a SELECT statement to the lookup source for lookup values.

You can set cached lookup in the run-time properties of the lookup operation in a mapping.

For more information about the cached lookup, see "Lookup Transformation" in the *Developer Transformation Guide*.

General Properties

The general properties display the name and description of the JDBC V2 lookup.

The following table describes the general properties that you can view and edit for a JDBC V2 lookup:

Property	Description
Name	Name of the JDBC V2 lookup.
Description	Description of the JDBC V2 lookup.
Physical Data Object	Name of the JDBC V2 data object read operation.
On multiple matches	Determines which row the JDBC V2 lookup returns when it finds multiple rows that match the lookup condition. You can select one of the following options: - Return first row - Return last row - Return any row - Return all rows - Report error

Ports Properties

The ports properties display the input ports from the source in the mapping to the JDBC V2 lookup. You can specify the ports to be available as output ports from the JDBC V2 lookup. The ports properties display the data types, precision, and scale of the source port.

The following table describes the ports properties:

Property	Description
Name	Name of the source port.
Type	Data type of the source port.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point of numeric values.
Output	Specify the ports that must be available as output ports from the JDBC V2 lookup.
Description	Description of the port.
Input Rules	A set of rules that filter the ports to include or exclude in the transformation based on port names or data type. Configure input rules when you define dynamic ports.

Run-time Properties

Set the run-time properties to configure a cached lookup in a mapping.

The following table describes the run-time properties for a JDBC V2 data object lookup operation in the Run-time view:

Property	Description
Lookup caching enabled	Indicates whether the Data Integration Service caches lookup values. By default, the **Lookup caching enabled** check box is selected. When you disable caching, each time a row passes into the transformation, the Integration Service issues a select statement to the lookup source for lookup values. Applicable in the native environment.

Lookup Properties

Specify the lookup properties to look up a JDBC V2 table. You can configure a lookup condition to look up data from the JDBC V2 table.

There are two types of option that you must select in the **Specify by** property to configure a lookup condition:

- **Value**: Select this option if you want to configure a lookup condition using the column name.
- **Parameter**: Select this option if you want to parameterize the lookup condition.

The following table describes the lookup properties that you can specify for a JDBC V2 lookup if you select the **Value** option:

Property	Description
Lookup Column	The name of the columns that you want to look up.
Operator	Operators that you can use to filter records. You can select one of the following operators: `=, !=, <=, >=, and`
Input Port	The input source port.

The following table describes the lookup properties that you can specify for a JDBC V2 lookup if you select the **Parameter** option:

Property	Description
Parameter	The name of the parameter that you want to use to look up. You can also create a new parameter. Click **New** to create a new parameter. Enter the parameter name and specify an expression in the **New Parameter** dialog box. Click **Validate** to check if the expression that you specified is valid or not.

Advanced Properties

The Developer tool displays advanced properties for the JDBC V2 data object lookup operation in the Advanced view. You can specify the tracing level.
The following table describes the advanced properties for a JDBC V2 data object read operation:

Property	Description
Pre SQL	The SQL query that the Data Integration Service runs before reading data from the source.
Post SQL	The SQL query that the Data Integration Service runs after reading data from the source. Applicable only in the native environment.
Fetch Size	The number of rows that the Data Integration Service fetches from the database in a single call.
Table Name	Overrides the table name used in the metadata import with the table name you specify.
Schema Name	Overrides the schema name of the source object. If you specify the schema name both in the connection and the source read operation properties, the Data Integration Service considers the schema name specified in the source properties.

Adding a JDBC V2 Data Object Operation as a Lookup in a Mapping

Use a JDBC V2 lookup to look up data in a JDBC V2 data object.

1. Open a mapping from the **Object Explorer** view.
2. From the **Object Explorer** view, drag a JDBC V2 data object read operation to the editor.

   The **Add to Mapping** dialog box appears.
3. Select **Lookup** to add the data object read operation as a lookup in the mapping.
4. Click inside the JDBC V2 object operation and connect the lookup input ports and the lookup output ports.
5. In the **Properties** view, configure the following parameters:
   a. On the **General** tab, select the option that you want the Data Integration Service to return when it finds multiple rows that match the lookup condition.
   b. On the **Ports** tab, configure the output ports and input rules.
   c. On the **Run-time** tab, select **Lookup caching enabled**.
   d. On the **Lookup** tab, enter the lookup condition properties.
6. When the mapping is valid, click **File** > **Save** to save the mapping to the Model repository.

CHAPTER 8

JDBC V2 Run-Time Processing

This chapter includes the following topics:

- JDBC V2 Run-Time Processing Overview, 38
- Join Expression, 38
- Filter Expression, 38
- Parameterization for JDBC V2 Sources and Targets, 39
- Partitioning, 40

JDBC V2 Run-Time Processing Overview

When you create a JDBC V2 data object read operation, you define properties that determine how the Data Integration Service reads data from a database that supports the Type 4 JDBC driver.

You can configure a filter expression from the query properties. You can also configure partitioning and parameterization in the run-time properties.

Join Expression

When you have two or more tables in a data object read operation, you can specify a join condition to join data from multiple tables.

You can configure the Join expression type as Native. You can also parameterize the Join condition.

Filter Expression

To read specific data from JDBC V2 table, you can configure a filter condition to query the database. You can use the Native or Platform expression to query specific columns from the database.

Native Expression

You can specify a native expression that uses AND, OR, or nested conditions. The expression that you enter becomes the WHERE clause in the query used to retrieve records from the source.

You can configure a filter that contains one or more Boolean expressions. Select **Native Expression** as the **Expression Type** and select the column on which you want to apply the filter condition.

The Boolean expressions use the following format:

```
<port> <Operator> <Value>
```

For example,

```
employeeID >= 100
```

If you use logical operators, add the operators as a prefix to the expression list. Default is blank.

To filter records from a JDBC V2 source, set the native expression in the data object read operation.

Platform Expression

You can use the platform expression to select specific records from a JDBC V2 table based on the filter condition you specify.

The following table describes the properties you specify when you use the platform expression to filter records from a JDBC V2 table:

Property	Description
Expression Type	The type of filter expression that you want to use to filter records. Select **Platform Expression**.
Left field	The column on which you want to apply the filter condition. The column names appear in the following format: `<reportName>.<datasetName_ColumnName>` For example, `User.ga_newUsers`
Operator	Simple operators you can use to filter records. You can select one of the following operators: `=, !=, <, <=, >, and >=`
Right field	The value you specify to filter the column.

Parameterization for JDBC V2 Sources and Targets

You can parameterize the JDBC V2 connection, JDBC V2 data object read operation properties, and JDBC V2 data object write operation properties to override the mapping properties at run time.

Partitioning

When you configure a JDBC V2 mapping to read data from the database that supports the Type 4 JDBC driver, you can configure partitioning to optimize the mapping performance at run time. The partition type controls how the Data Integration Service distributes data among partitions at partition points.

The following table summarizes the partition type options that you can configure for a read or write operation on the native and non-native environments:

Operation Type	Supported Partition Type in Native Environment	Supported Partition Type in Non-Native Environment
Read	You can select from the following partition types: - **None**. By default, the Data Integration Service creates a single partition. - **Key Range**. The Data Integration Service distributes rows of data based on a port or set of ports that you define as the partition key. The default number of partitions is 2.	You can select from the following partition types: - **None**. By default, the Data Integration Service creates a single partition. - **Fixed**. Define partition key for at least one column.
Write	- **None**. By default, the Data Integration Service creates a single partition. - **Dynamic**. By default, the Data Integration Service creates the same number of partitions for the target based on the number of partitions you specified for the source.	- **None**. By default, the Data Integration Service creates a single partition. - **Fixed**. Define partition key for at least one column. - **Dynamic**. By default, the Data Integration Service creates the same number of partitions for the target based on the number of partitions you specified for the source.

Key Range Partitioning

You can configure key range partitioning for a source operation that runs in the native environment.

The Data Integration Service distributes rows of data based on a port or set of ports that you define as the partition key. You define a range of values for each port. The Integration Service uses the key and ranges to send rows to the appropriate partition.

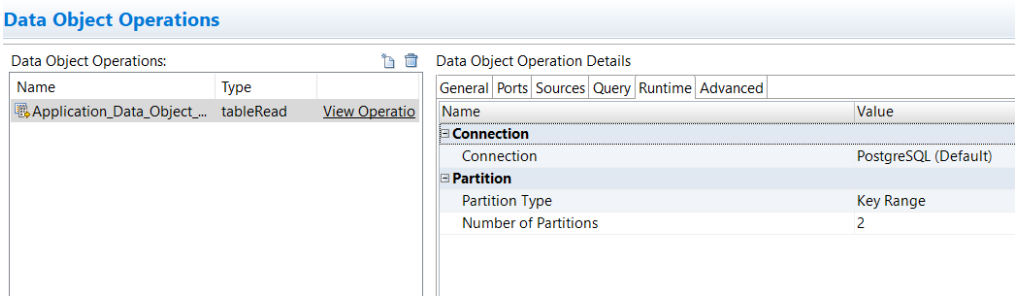
For example, if you select key range partitioning, the Data Integration Service uses the key and ranges to create the WHERE clause when it selects data from the source. Therefore, you can have the Data Integration Service pass all rows that contain a specified value to one partition and all other rows to another partition.

Configure Key Range Partitioning

You can configure key range partitioning for a mapping to read data from databases with Type 4 JDBC driver.

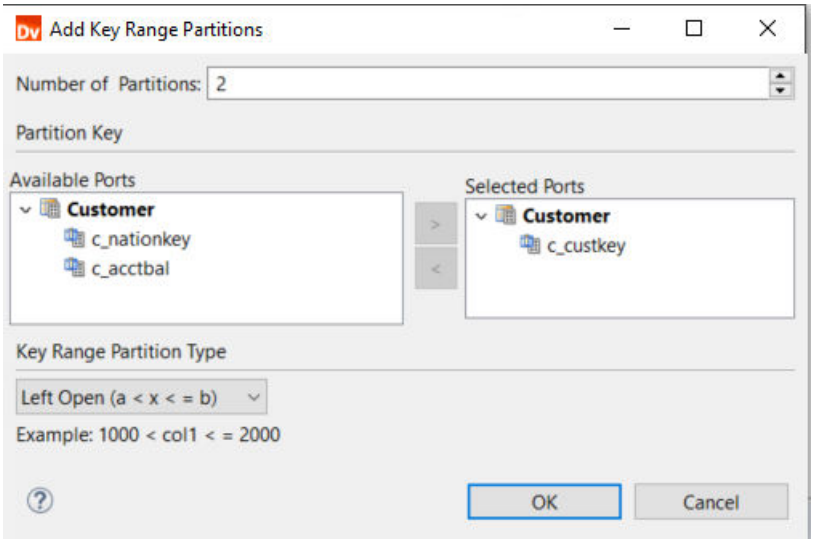
1. Open the JDBC V2 data object read operation for which you want to configure partitioning.

2. On the **Run-time** tab in Data Object Operation Details, select **Key Range** as the **Partition Type**.

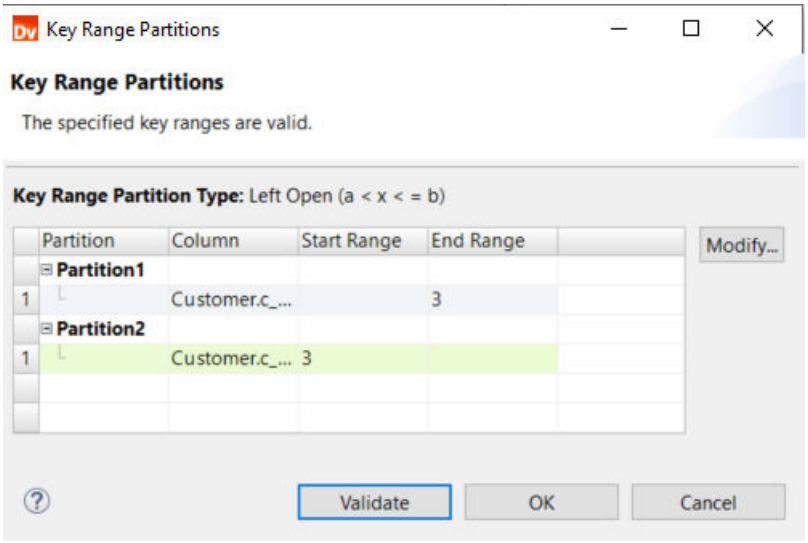


The default number of partitions created for key-range partitioning is 2.

3. To configure the key range partitioning, in the **Number of Partitions** field, click the **Open** button ( ).
4. In the Key Range Partitions dialog box, click **Add**.
5. Select the ports that you want to specify as the partition key, and click **OK**.



6. Specify the start and end ranges for each of the partitions and validate to verify if the specified key ranges are valid.



7. Click **OK**.

Rules and Guidelines for Specifying Key Ranges

Use the following rules and guidelines when you create key ranges:

- You can leave the start and end range blank for a partition. When you leave the start range blank, the Integration Service uses the minimum data value as the start range. When you leave the end range blank, the Integration Service uses the maximum data value as the end range.
- Specify the key range for the partitions in a continuous sequence.
  For example, if you configure three partitions, where 1-3 is the range for the first partition, partition 2 must start with the end range of the first partition and partition 3 must start with the end range of the second partition:

  partition 1: 1-3

  partition 2: 3-5

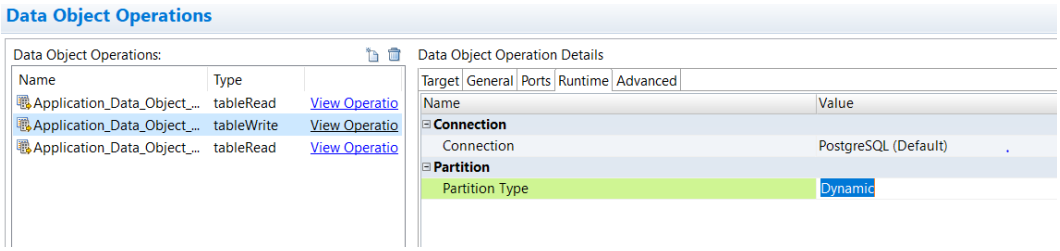
  partition 3: 5-7
- When you enable key range partitioning for a mapping that runs in the native environment, change the **Maximum Parallelism** attribute in the Data Integration Service properties from the default value to an appropriate value, as required. Else, the mapping fails with a java.lang.Exception error.

Dynamic Partitioning

When you configure a write operation on the native or non-native environment, you can configure dynamic partitioning to optimize the mapping performance.

When you select dynamic partitioning, the Data Integration Service determines the number of partitions to create at runtime based on the number of partitions specified for the source.

The following image shows the value of the partitioning type set to dynamic:



Fixed Partitioning

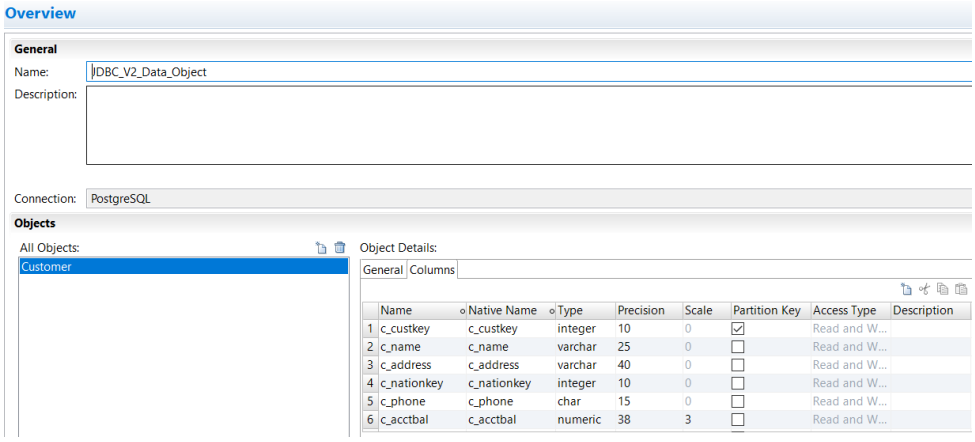
You can configure fixed partitioning for a source or target operation in the non-native environment.

To configure fixed partitioning, you must specify the number of partitions that the Data Integration Service must create at runtime. You must select the columns that you want to define as the partition key.

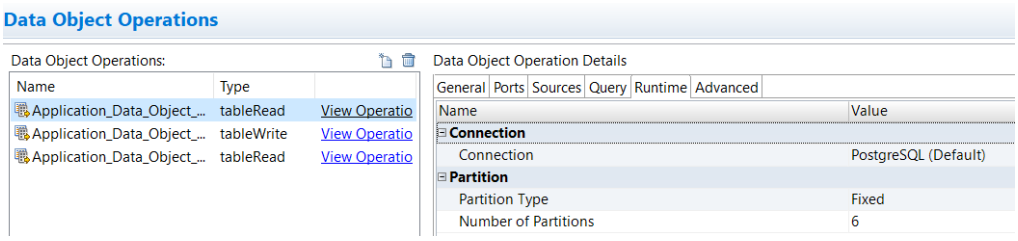
Configure Fixed Partitioning

To configure fixed partitioning for a read or write operation, you must select define at least one column as the partition key.

1. Select the columns in the table that you want to define as partition keys. Perform the following tasks:
   a. Select the **Overview** tab for the read or write operation.
   b. Under **Object Details**, select the **Columns** tab.
   c. Select the name of the column that you want to assign as the partition key.



2. Open the JDBC V2 data object read or write operation for which you want to configure fixed partitioning.
3. On the **Run-time** tab in Data Object Operation Details, select **Fixed** as the **Partition Type**.



4. Specify the number of partitions that the Data Integration Service must create at runtime.

Audits

To validate the consistency and accuracy of data processed in a mapping for a read operation, you can create an audit for the mapping.

An audit is composed of rules and conditions. Use a rule to compute an aggregated value for a single column of data. Use a condition to make comparisons between multiple rules or between a rule and constant values.

You can run audits with mappings that run on the Data Integration Service or the Spark engine.

For more information, see the *Data Engineering Integration 10.5 User Guide*.

CHAPTER 9

JDBC V2 Datatype Reference

This chapter includes the following topics:

- Datatype Reference Overview, 45
- JDBC V2 and transformation data types, 45

Datatype Reference Overview

Informatica Developer uses the following data types in JDBC V2 mappings:

- JDBC V2 native data types. JDBC V2 data types appear in the physical data object column properties.
- 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 Data Integration Service uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

When PowerExchange for JDBC V2 reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When PowerExchange for JDBC V2 writes to a target, it converts the transformation data types to the comparable native data types.

JDBC V2 and transformation data types

The following table lists the Aurora PostgreSQL data types that the Data Integration Service supports and the corresponding transformation data types:

Aurora PostgreSQL Data Type	Transformation Data Type	Description
integer	integer	-2147483648 to 2147483647, precision 10, scale 0
smallint	integer	-32768 to 32767, precision 10, scale 0
bigint	bigint	-9223372036854775808 to 9223372036854775807, precision 19, scale 0
real	double	Precision 15

Aurora PostgreSQL Data Type	Transformation Data Type	Description
double precision	double	Precision 15
decimal	decimal	Precision 1 to 38, scale 0 to 28
numeric	decimal	Precision 1 to 38, scale 0 to 28
boolean	string	Precision 6
char	string	Fixed length, blank padded, precision 1 to 4000
varchar	string	Variable length with limit, precision 1 to 4000
date	date/time	January 1, 0001, through December 31, 9999
time	date/time	00:00:00.000 through 23:59:59.999
timestamp	date/time	Date range: January 1, 0001, through December 31, 9999 time range: 00:00:00 through 23:59:59.997
text	string	Variable unlimited length, precision 4000

The following table lists the Azure SQL Database data types that the Data Integration Service supports and the corresponding transformation data types:

Azure SQL Database Data Type	Transformation Data Type	Description
char	string	Precision 1 to 4000
varchar	string	Precision 1 to 4000
text	text	Maximum string length of 2^31-1 (2,147,483,647), precision 4000
nchar	string	Precision 1 to 4000
nvarchar	string	Precision 1 to 4000
ntext	string	Maximum string length of 2^30 - 1 (1,073,741,823), precision 4000
bigint	bigint	-2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807), precision 19, scale 0
date	date/time	October 15, 1582 CE through December 31, 9999 CE (1582-10-15 through 9999-12-31)
datetime	date/time	Date range: January 1, 1753, through December 31, 9999, time range: 00:00:00 through 23:59:59.997

Azure SQL Database Data Type	Transformation Data Type	Description
datetime2	date/time	Date range: 0001-01-01 through 9999-12-31, time range: 00:00:00 through 23:59:59.9999999
datetimeoffset	string	Date range: 0001-01-01 to 9999-12-31, time range: 00:00:00 to 23:59:59.9999999, time zone offset range: -14:00 through +14:00
decimal	decimal	-922,337,203,685,477.58 to 922,337,203,685,477.58
float	double	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38
int	integer	0 to 255
money	decimal	-922,337,203,685,477.58 to 922,337,203,685,477.58
numeric	decimal	-922,337,203,685,477.58 to 922,337,203,685,477.58
real	double	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38
smalldatetime	date/time	00:00:00.000 through 23:59:59.999
smallint	integer	-32,768 to 32,767
smallmoney	decimal	- 214,748.3648 to 214,748.3647
time	date/time	00:00:00.000 through 23:59:59.999
tinyint	integer	0 to 255

The following table lists the SAP HANA data types that the Data Integration Service supports and the corresponding transformation data types:

SAP HANA Datatype	Transformation Datatype	Description
Alphanum	Nstring	1 to 104,857,600 characters
Bigint	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Binary	Binary	1 to 104,857,600 bytes
Bintext	String	Precision 32000
Boolean	Integer	Boolean (True/False) values, precision 10
Date	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)

SAP HANA Datatype	Transformation Datatype	Description
Decimal (precision, scale) or Dec (p, s)	Decimal	Precision 1 to 28, scale 0 to 28
Double	Double	Precision 15
Float	Double	Precision 15
Integer	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Nvarchar	String	1 to 104,857,600 characters
Real	Double	Precision 7, scale 0
Seconddate	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Shorttext	String	1 to 104,857,600 characters
Smalldecimal	Decimal	Precision 1 to 28, scale 0 to 28
Smallint	Integer	Precision 5, scale 0
Text	String	1 to 104,857,600 characters
Time	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Tinyint	Integer	Precision 5, scale 0
Varchar	String	1 to 104,857,600 characters
Varbinary	Binary	1 to 104,857,600 bytes

INDEX

A

advanced properties
  target 19

C

column properties 15
configure a mapping
  native 26
  non-native 26
create
  JDBC V2 connection 12
create target
  JDBC V2 21
creating
  JDBC V2 data object read operation 23

D

data object read operation
  creating 23
dynamic mappings
  developing and running 32

G

general properties
  input 18
General properties 16

I

importing
  JDBC V2 data object 21
input properties 18

J

JDBC
  lookup overview 34
JDBC data types
  comparing with transformation data types 45
JDBC lookup
  general properties 35
JDBC V2
  dynamic mapping 31
  importing a data object 21
JDBC V2 connection
  create 12
  overview 10
JDBC V2 connection *(continued)*
  properties 10
JDBC V2 data object
  overview 13
JDBC V2 data object read operation
  creating 23
JDBC V2 data types
  overview 45
JDBC V2 dynamic mapping
  example 32
JDBC V2 lookup
  advanced properties 37
  creating 37
  lookup properties 36
  ports properties 35
JDBC V2 parameterization
  for sources 39
  for targets 39
JDBC V2 run-time environment
  description 25
JDBC V2 run-time processing
  parameterization 39
JDBC V2 validation environment
  description 25
JDBC V2 write operation
  properties 18

M

mapping example
  JDBC V2 27

O

Output properties 14
overview
  connection 10

P

Ports properties 16
PowerExchange for JDBC V2
  overview 7
  prerequisites 8

Q

query
  native expression 38, 39
  platform expression 38, 39

R

run-time properties
non-native mapping 25

S

source properties 14

V

validation environments
non-native mapping 25