How to Generate a Mapping from an SQL Query in Informatica Developer
Abstract

You can generate a mapping from an SQL query in the Developer tool. You can create or paste an SQL statement, or load a text file that contains an SQL query. Optionally, you can define or redefine the source of the query table. The Developer tool validates the SQL query and generates a mapping.

Supported Versions

- Informatica Data Quality 10.1
- Informatica Big Data Management 10.1

Table of Contents

Overview ........................................... 2
Example of Generated Mapping from an SQL Query. ............................................ 3
SQL Syntax to Generate a Mapping. ................................................................. 4
Correlated Subqueries. ................................................................. 4
Function Support in Queries that Generate a Mapping. ........................................ 4
Generate a Mapping from an SQL Query with an Unsupported Function. ..................... 5
INSERT, UPDATE and DELETE Syntax .......................................................... 6
Rules and Guidelines for INSERT, UPDATE, and DELETE Statements. ......................... 6
Generate a Mapping from an SQL Statement. ................................................. 6
Create an SQL Statement. ................................................................. 6
Paste or Import the SQL Statement to the Developer Tool. ..................................... 7
Complete Mapping Development. .............................................................. 8

Overview

A mapping is a collection of metadata which allows you to extract data from one or more data sources, perform calculations and transformations on the data, and write the result to the same or another data source. The Developer tool gives you several ways to create a mapping. One method is to generate a mapping from an SQL query. You can generate a mapping from an SQL query that you write from scratch in the Developer tool, or by referring to a text file that contains the query.

You can also generate a logical data object from an SQL query that contains only SELECT statements.

The Developer tool validates the SQL before you run the mapping.

After you generate the mapping, you can edit the mapping in the Developer tool, and you can include it in an application that you deploy on the Data Integration Service.

The following image shows the process of generating a mapping from an SQL query:
Example of Generated Mapping from an SQL Query

You have a table of employees, and you want a list of employee salaries for employees that were hired after January 1, 2001.

You create the following SQL statement:

```
SELECT LastName, Salary from emp1 where HireDate > 01/01/2001
```

The following image shows the mapping created from the SQL statement:
SQL Syntax to Generate a Mapping

You can use an ANSI-compliant SQL statement to generate a mapping in the Developer tool. The Developer tool can generate a mapping from a standard SELECT query. For example:

```sql
SELECT column_list FROM table-name
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
```

If the SELECT SQL statement contains a correlated subquery, the query is valid if it is possible to flatten or rewrite the query as a single standard query.

Correlated Subqueries

A correlated subquery is a subquery that uses values from the outer query in its WHERE clause. The Data Integration Service flattens the correlated subqueries before it runs the query.

The following table shows the results of a correlated subquery that the Data Integration Service flattened:

<table>
<thead>
<tr>
<th>Type</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-flattened</td>
<td><code>SELECT huge.* FROM huge WHERE c1 IN (SELECT c1 FROM tiny)</code></td>
</tr>
<tr>
<td>Flattened</td>
<td><code>SELECT huge.* FROM huge, tiny WHERE huge.c1 = tiny.c1</code></td>
</tr>
</tbody>
</table>

The Data Integration Service can flatten a correlated subquery when it meets the following requirements:

- The type is IN or a quantified comparison.
- It is not within an OR operator or part of a SELECT list.
- It does not contain the LIMIT keyword.
- It does not contain a GROUP BY clause, aggregates in a SELECT list, or an EXIST or NOT IN logical operator.
- It generates unique results. One column in the correlated subquery is a primary key. For example, if `r_regionkey` column is a primary key for the `vs.nation` virtual table, you can issue the following query:
  ```sql
  SELECT * FROM vs.nation WHERE n_regionkey IN (SELECT b.r_regionkey FROM vs.region b WHERE b.r_regionkey = n_regionkey).
  ```
- If it contains a FROM list, each table in the FROM list is a virtual table in the SQL data service.

Function Support in Queries that Generate a Mapping

Informatica supports functions that meet the ANSI SQL-92 standard.

In addition, some functions have specific syntax requirements.
The following table lists the functions and supported syntax:

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE( )</td>
<td>To specify the format of a date:</td>
</tr>
<tr>
<td></td>
<td>DATE(format '&lt;format&gt;')</td>
</tr>
<tr>
<td></td>
<td>where &lt;format&gt; is a standard date format.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>SELECT DATE(format 'dd-mm-yyyy') from table</td>
</tr>
<tr>
<td>POSITION( )</td>
<td>To determine the position of a substring in a literal string:</td>
</tr>
<tr>
<td></td>
<td>POSITION('&lt;substring&gt;', '&lt;string&gt;')</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>POSITION('MA', 'James Martin')</td>
</tr>
<tr>
<td></td>
<td>To determine the position of a substring in a table column:</td>
</tr>
<tr>
<td></td>
<td>POSITION('&lt;substring&gt;', &lt;column_name&gt;)</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>POSITION('MA', FULL_NAME)</td>
</tr>
</tbody>
</table>

**Generate a Mapping from an SQL Query with an Unsupported Function**

When the Developer tool generates a mapping from SQL, it validates the functions in the query. Use ANSI-compliant SQL to ensure valid mapping generation.

If the Developer tool encounters an unknown function in a valid SQL statement, it might generate a mapping that contains a transformation labeled FIX_ME or an expression labeled FIX_EXPR. Edit these objects to fix the mapping and get valid results. Unknown functions appear as a warning message in the mapping log file.

For example, you use the following SQL statement to generate a mapping:

```
SELECT unknownFunctionABC(c_custkey,c_comment) from customer
```

The following image shows how the mapping generated from this SQL statement includes an Expression transformation that requires fixing:

Notice that the Expression transformation is marked with an error icon. Use the Ports tab to edit the erroneous expression. The mapping is not valid until you correct the error.
**INSERT, UPDATE and DELETE Syntax**

Use the following syntax to create valid INSERT, UPDATE and DELETE statements:

- Use the following syntax for an INSERT statement:
  
  ```sql
  INSERT INTO <TABLENAME> [ <list> ]
  <select query>
  ```

- Use the following syntax for an UPDATE statement:
  
  ```sql
  UPDATE [schema .] [ table | view ] [ alias ]
  SET column = { expr | subquery } [...]
  [WHERE condition]
  ```

- Use the following syntax for a DELETE statement:
  
  ```sql
  DELETE FROM <Table> [ [AS] <ALIAS> ] [WHERE condition]
  ```

**Rules and Guidelines for INSERT, UPDATE, and DELETE Statements**

Consider the following rules and guidelines for INSERT, UPDATE, and DELETE statements:

- An INSERT, UPDATE, or DELETE statement creates source and target objects in the mapping that are logical data objects.

- Only one INSERT, UPDATE, or DELETE statement is valid. For example, a statement that contains an INSERT and a nested UPDATE statement is not valid.

- When the INSERT, UPDATE, or DELETE SQL statement contains a correlated subquery, the Developer tool cannot generate a mapping.

- An UPDATE or DELETE statement creates an Update Strategy transformation in a mapping. Because an Update Strategy transaction requires a primary key, the data target must contain a primary key. After mapping generation, verify the primary keys.

**Generate a Mapping from an SQL Statement**

To generate a mapping from an SQL statement, perform the following tasks:

1. Create an SQL statement.
2. Paste or import the SQL statement to the Developer tool, validate the SQL statement, and generate a mapping.
3. Complete mapping development. Follow these steps:
   a. Test and iteratively develop the mapping until it meets requirements.
   b. Deploy the mapping to the Data Integration Service.

**Create an SQL Statement**

To use an SQL statement to generate a mapping, create an SQL statement.

You can use an SQL query tool or write an SQL statement from scratch to create an SQL statement. Follow the syntax guidelines in this article.

**Note:** Some non-Informatica functions are supported. Others can be used in a valid query that generates a mapping with results that are not valid. For more information about function support in SQL statements, contact Informatica Global Customer Support.
Paste or Import the SQL Statement to the Developer Tool

1. Locate the SQL file that contains the SQL statement to import, or copy the entire statement to the clipboard.

2. In the Developer tool, click **File > New > Mapping from SQL Query**
   
The **Generate Mapping or Logical Data Object from SQL Query** dialog box opens.

3. Import the query to the dialog box. Choose one of these methods:
   - Select **Enter an SQL Query** and paste the query from the clipboard to the editor.
   - Select **Select an SQL File** and browse to and select the file.

4. Click **Validate**.
   
   Developer tool validates the SQL statement. Correct any errors.

5. If you want to generate a logical data object instead of a mapping, select **Generate Logical Data Object from the SQL Query**.
   
   If you select this option, perform the following steps:
   1. Optionally rename the logical data object to create.
   2. Optionally click **Browse** to select a location for the Logical Data Object, or accept the default location.
   3. Click **Validate**.
      
      Developer tool validates the SQL statement. Correct any errors.

6. Click **Next**.
   
   The **Select the data source for the table** dialog box opens.

7. To select a data source for the table, click in the Data Source column, and then click **Browse**.
   
   The following image shows where to click **Browse** to select a data source:
8. Click Finish.

The Developer tool generates a mapping from the SQL query and opens the mapping in an editor.

**Complete Mapping Development**

After you create a mapping, perform the following steps to complete development of the mapping:

1. Preview mapping results.
2. Iteratively edit and preview the mapping until it meets requirements.
3. Deploy and run the mapping on the Data Integration Service.

   You can deploy the mapping by itself, or include it in an application that you deploy. If you deploy the mapping by itself, the Data Integration Service creates an application to contain it.

For more information about mappings, applications, and deployment, see the *Informatica 10.1 Developer Tool Guide*, or read the How-to Library (H2L) article, "How to Create, Deploy, and Update an Application," on the Informatica Network.

**Author**

Mark Pritchard
Principal Technical Writer