Database Import Component

The component is designed for importing data from DBMSs. The node based on this component imports database tables, views, or custom SQL query results.

In an SQL query, control variables can be used as parameters or macro substitutions. This allows you dynamically construct the request before it is sent to the server.

> **Important:** To use the node, first create a database connection, and link it to the node's **Connection** input port. (See Integration with Databases.)

Ports

Input ports

- **Connection**: Accepts database connection parameters.
- **Control variables**: An optional port which allows you to use variables for setting configuration parameters.

Output ports

- **Dataset**: The table with imported data.
- **Execution status**: This output port's variables provide the error message, if any, and the execution status code:
  - **0**: node executed without errors
  - **1**: node execution error
  - **2**: execution timed out

Config wizard

The node configuration wizard includes several steps:

Step 1. Import from database

This page provides fields for the following parameters:

- **Connection**: Displays the data source connection string. This field is uneditable.
- **Choose table**: Select this option if you need import data from a database table. To specify the table or view, use the **Table/View** dropdown list below. If the connection is inactive, click the **Activate** button on the top left.
- **SQL query**: Select to create a custom query.
- **Ignore errors**: If you mark this checkbox, the node will run even if errors appear during activation.
- **Import timeout (s)**: This setting defines the maximum time allowed for executing queries and receiving data, though it excludes the initial database server connection time. Note that this limit does not apply to data previews within the wizard. If the timeout expires, the system will stop the active stage: either the query execution or the appending of the output dataset. Use caution when **Ignore errors** is enabled, as reaching the timeout may result in a partially appended dataset rather than a complete cancellation.

The following fields depend on the the selected query type (table or SQL).

- When **Choose table** is selected:
  - **Table/view**: The dropdown to select the data source from. You can also type the table name in the field. To display the list, the connection must be activated using the **Activate** button.
  - **Tables/views**: The **Tables/views** pane on the left displays all available database tables; double-click a table name to populate its specific fields in the list on the right. Once the fields appear, you can quickly select all of them by clicking the **Extract all** button (above the field list) or manualy choose specific fields by checking their individual boxes.
  - **Generate field captions from descriptions**: Select this checkbox to use database field descriptions as output captions. This option is only available if the database supports field descriptions and is disabled by default.
- When **SQL query** is selected:
  - **Tables/views** pane (left): contains available database tables. Click the ... **Show** button at the bottom to reveal the variables pane (hidden by default).
  - **SQL editor** (right): Use this field to write your SQL query query. You can add objects or variables to the query body using these methods:
    - *Drag and drop*: moves the specific object name (from the table list) or a parameter (from the variables pane).
    - CTRL + *drag-and-drop*: moves the full name of a table object or a macro substitution from the variables pane.
    - *Double click*: quickly inserts the clicked object into the editor.
    - *Right click*: opens a context menu with such options as **Insert name to SQL code editor** or **Insert full name to SQL code editor**.
- **Preview**: Click the Preview button to verify your connection configuration. This displays the first 100 rows of the resulting output table.

Step 2. Data filter

This wizard page appears only if **Choose table** mode was selected in the previous step. Use it to configure filters for your imported data. The process is similar to the configuration of the standard Row Filter node. However, filtering import data has the following limitations:

- **Row numbers:** Filtering by row number is not supported.
- **Unique values:** Receiving the list of unique values for a field is unavailable.
- **Case sensitivity:** For `string` datatype fields, the **Case-sensitive** checkbox is only available if supported by the connected DBMS. (For example, MS Access does not support case-sensitive string comparisons.)

The list of fields available for filtering is independent of the output columns selected in Step 1. When the import node is executed, these filters are converted to `WHERE` clauses in the resulting SQL query. Consequently, just as in standard SQL, the columns used for filtering do not need to be included in the `SELECT` statement.

When you navigate to the data filter step, the platform retrieves the information about the columns based on the database connection status:

- If the database connection is inactive:
  - The system automatically uses fields from a table imported earlier in the current session.
  - If no previous table data is available, the system retrieves column information from any existing filter configurations.
- If the database connection is active:
  - The available columns are loaded directly from the selected table after the connection query is executed.
  - If an error occurs while running the query, the field list remains empty.

The filtering step is optional. You can proceed to the next page without configuring filters, and no errors or warnings will be generated.

Using variables in queries

You can use variables in your SQL queries in two ways:

- **Parameters**: to pass specific values into a query. Parameters are prefixed with a colon (e.g., `:var1`). In this mode, the system automatically respects the variable's data type (for example, string values are automatically wrapped in quotation marks). Parameters can only be used within the `WHERE` clause.
- **Macro substitutions**: to insert raw SQL code or query fragments. These must be enclosed in percent signs, (e.g., `%condition%`). To use this method, select it from the variable's context menu.

> **Note:** If the substitution construction is placed inside quotation marks, the quotation marks are prioritized, and the substitution may be treated as a literal string rather than executable code.

▼ Example:

```
SELECT * FROM %tables%
WHERE
   %cond1% :TitlePart
   AND Books.year > :MinYear
   AND Publisher.Name LIKE '%Media%'
```

where

variable name	type	example value
tables	string	Books JOIN Publisher ON Books.PubId = Pub.Id
cond1	string	title LIKE
TitlePart	string	Adventures%
MinYear	integer	2005
Media	string	Astra Info

In this case, `'%Media%'` is treated as a literal string. Because it is enclosed in quotation marks, the system does not recognize it as a macro substitution and will not replace it with the variable's value.

The resulting query will look like this:

```
SELECT * FROM Books JOIN Publisher ON Books.PubId = Pub.Id
WHERE
   title LIKE 'Adventures%'
   AND Books.year > 2005
   AND Publisher.Name LIKE '%Media%'
```

Read on: Text File Import