

# Using DbVisualizer Variables

DbVisualizer variables are used to build parameterized SQL statements and let DbVisualizer prompt you for the values when the SQL is executed. This is handy if you are executing the same SQL repetitively, just wanting to pass new data in the same SQL statement.

- [Variable Syntax](#)
- [Pre-defined Variables](#)
- [Variables for Java System Properties and OS Environment Variables](#)
- [Using "now" in values for Time, Date, and Timestamp](#)
- [Variable Substitution in SQL statements](#)

A DbVisualizer variable that **doesn't specify a data type** will always be replaced with the **value as a literal**. This allows use of variables anywhere in an SQL statement. If a **data type is specified**, the prompted value will be bound with the SQL and variables can in this context **only be used where supported** by the target database.

DbVisualizer Variables	
<code>\${variable    value    type    options}\$</code>	<p>This is the most flexible syntax as it supports setting a name, default value, data type, and other options. Check the <a href="#">DbVisualizer Variables</a> section for details.</p> <p>A DbVisualizer variable can be used anywhere in the SQL as the specified value replaces the variable definition as a literal (unless a data type is specified; with a data type, its behavior is exactly the same as for <b>Named Parameter Markers</b>).</p> <div style="border: 1px solid #ccc; padding: 5px;"><p><b>Example</b></p><pre>select * from   EMPLOYEE where  FIRST_NAME like '\${First Name}  Phil}\$' and    AGE &gt; \${Age}  20}\$</pre></div> <p>The variable identifiers, <code>\${...}\$</code> can be modified in <b>Tools-&gt;Tool Properties</b> and in the <b>General / Variables</b> category.</p>

## Variable Syntax

The variable format supports setting a default value, data type and a few options as in the following example:

```
${FullName||Andersson||String||where pk}$
```

This is the complete syntax for a DbVisualizer variable:

```
${name || value || type || options}$
```

Part	Default	Description
<b>name</b>	<b>Required</b>	Required. This is the name that appear in the prompt window. If multiple variables in a script have the same name, the substitution dialog shows only one and the entered value will be applied to all variables with that name
<b>value</b>	null	The default value for the variable
<b>type</b>	none (= literal)	The type of variable: String, Boolean, Integer, Float, Long, Double, BigDecimal, Date, Time and Timestamp. In addition DbVisualizer defines: BinaryData and TextData (for CLOB). This is used to determine how the data should be passed between DbVisualizer and the database server. If no type is specified, it is treated as a literal

<b>options</b>	none	<p>The options part is used to express certain conditions. Separate these with a whitespace</p> <ul style="list-style-type: none"> <li>• <b>pk</b> Indicates that the variable is part of the primary key in the final SQL. Represented with a symbol in the prompt window</li> <li>• <b>where</b> Defines that the variable is part of the WHERE clause. A symbol indicate this condition in the prompt window</li> <li>• <b>noshow</b> This option define that the variable should not appear in the prompt window. A value must be set when using this option, unless it is an output variable (see <b>dir</b> below)</li> <li>• <b>nobind</b> Used in combination with when a <b>type</b> is set and defines that the variable should be replaced as a literal in the SQL rather than being bound as a parameter marker</li> <li>• <b>dir=in   out   inout</b> The direction for a variable used with the <b>@call</b> command (it is ignored for other uses). A variable assigned the return value for a function must be declared as <b>dir=out</b>, and a variable used for a procedure parameter must use a dir type matching the procedure parameter direction declaration. <b>in</b> is the default</li> <li>• <b>scope=post</b> The scope must be specified as <b>post</b> when using variables representing columns when exporting BLOB/CLOB values to separate files named based on column values.</li> </ul>
----------------	------	--

## Pre-defined Variables

A few pre-defined DbVisualizer variables can be used anywhere in the SQL. These are replaced with actual values just before the SQL is sent to the DB server.

Note that none of the pre-defined variables below will show in the prompt window.

```

${dbvis-date}$
${dbvis-time}$
${dbvis-timestamp}$
${dbvis-connection}$
${dbvis-database-type}$

```

By default, date/time variable values are formatted as defined in **Tool Properties->Data Formats**, but you can also specify a custom format for a single use of the variable, e.g.

```

${dbvis-date|||||format=[yyyymmdd]}$

```

The following variables can be used only when monitoring a SQL statement that produce a result set and the **Allowed Row Count** for the monitor is > 0. The output format is seconds and milliseconds. Ex: 2.018

```

${dbvis-exec-time}$
${dbvis-fetch-time}$

```

The following variable holds the absolute path to the current directory, e.g. set by the **@cd** command:

```

${dbvis-pwd}$

```

In an sql script, the name on the result set produced by the next SELECT statement can be set with the **@set resultset name** command (see [Using Client-Side Commands](#)). This result set name is accessible through the variable

```

${dbvis-resultset-name}$

```

## Variables for Java System Properties and OS Environment Variables

You can use DbVisualizer variables to access the value of a Java system property by prefixing the property name with **java.** (java.<property>). Please note that some java properties include "java" in the property name!

Examples:

```

@echo ${java.user.home}$
@echo ${java.java.io.tmpdir}$

```

To access the value of an operating system environment variable, prefix the variable name with **env.** (env.<variable>).

Examples:

```

@echo ${env.USER}$
@echo ${env.TMP}$

```

This may be used in many cases, for instance when importing or exporting files. The export example below shows how to specify the DbVisualizer Bookmarks folder without explicitly giving the hardcoded folder path.

```

@export set filename="${java.dbvis.prefsdir}$/Bookmarks/myscript.sql" Format="SQL" AppendFile="false";

```

A list of Java properties can be found in the **Java Properties** tab accessible from the **HelpAbout...** menu

## Using "now" in values for Time, Date, and Timestamp

For variables with the type set to `Time`, `Date`, and `Timestamp`, the value may be set to the literal `now`. The value is then converted to the specified type with the format defined in **Tool Properties** and **Data Formats** category.

Variable	Format	Sample
<code>\${myDate   now   Date}\$</code>	yyyy-MM-dd	2017-07-17
<code>\${myDate   now   Time}\$</code>	HH:mm:ss	09:02:50
<code>\${myDate   now   Timestamp}\$</code>	yyyy-MM-dd HH:mm:ss	2017-07-17 09:03:11

For these types it is also possible to specify the value `now` in the variable prompting window.

## Variable Substitution in SQL statements

For variable processing to work in the SQL Commander, make sure the **SQL Commander Options->Parameterized SQL** is checked in the **SQL Commander** main menu.

A simple variable may look like this:

```
${FullName}$
```

A variable is identified by the start and end sequences, `${...}$`. (These can be [re-defined](#) in **Tool Properties**). During execution, the SQL Commander searches for variables and displays the prompt window with the name of each variable and an input (value) field. Enter the value for each variable and then press **Execute**. This will then replace the variable with the value as a literal and finally let the database execute the statement.

Consider the following SQL statement with variables. It is the simplest use of variables since it only contains the variable names. In this case it is also necessary to enclose text values with quotes since the prompt window cannot determine the actual data type for the variables.

```
INSERT
INTO
    EMPLOYEES
(
    EMPLOYEE_ID,
    FIRST_NAME,
    LAST_NAME,
    EMAIL,
    PHONE_NUMBER,
    HIRE_DATE,
    JOB_ID,
    SALARY,
    COMMISSION_PCT,
    MANAGER_ID,
    DEPARTMENT_ID
)
VALUES
(
    ${EMPLOYEE_ID}$,
    ${FIRST_NAME}$,
    ${LAST_NAME}$,
    ${EMAIL}$,
    ${PHONE_NUMBER}$,
    ${HIRE_DATE}$,
    ${JOB_ID}$,
    ${SALARY}$,
    ${COMMISSION_PCT}$,
    ${MANAGER_ID}$,
    ${DEPARTMENT_ID}$
)
```

Executing the above SQL will result in the following prompt window:

Key	Name	Value	Type
	EMPLOYEE_ID		Literal
	FIRST_NAME		Literal
	LAST_NAME		Literal
	EMAIL		Literal
	PHONE_NUMBER		Literal
	HIRE_DATE		Literal
	JOB_ID		Literal
	SALARY		Literal
	COMMISSION_PCT		Literal
	MANAGER_ID		Literal
	DEPARTMENT_ID		Literal

1: EMPLOYEE\_ID  
Not NULL  
JDBC: N/A, Java: Literal

Show SQL

Format: Text

Continue Cancel

Using variables with no data type defined shows these as **Literal**. This means that the specified value will replace the variable as-is in the SQL statement.

The prompt window has the same look and functionality as the Form Data Editor, i.e. you can sort, filter, insert pre-defined data, copy, paste and edit cells in the multi line editor, plus a lot of other things. In addition the prompt window adds two new commands (leftmost in the toolbar and in the form right-click menu).

<b>Set Default Values</b>	This will set each value to the default value for the variable. If a default value was not specified in the variable, <b>(null)</b> will shown
<b>Set Previously Used Values</b>	Set the value for each variable to the values (matched by name) that was used in the previous run (if there are no values from a previous run, this button is disabled)

The **SQL Preview** area shows the statement with all variables replaced with the values.

Here is an example of a more complex use of variables utilizing default value, data type and options:

```

INSERT
INTO
  EMPLOYEES
(
  EMPLOYEE_ID,
  FIRST_NAME,
  LAST_NAME,
  EMAIL,
  PHONE_NUMBER,
  HIRE_DATE,
  JOB_ID,
  SALARY,
  COMMISSION_PCT,
  MANAGER_ID,
  DEPARTMENT_ID
)
VALUES
(
  ${EMPLOYEE_ID}|105|BigDecimal|pk ds=7 dt=NUMERIC}$,
  ${FIRST_NAME}|David|String|nullable ds=20 dt=VARCHAR}$,
  ${LAST_NAME}|Austin|String|ds=25 dt=VARCHAR}$,
  ${EMAIL}|DAUSTIN|String|ds=25 dt=VARCHAR}$,
  ${PHONE_NUMBER}|590.423.4569|String|nullable ds=20 dt=VARCHAR}$,
  ${HIRE_DATE}|2005-06-25 00:00:00|Timestamp|ds=7 dt=TIMESTAMP}$,
  ${JOB_ID}|IT_PROG|String|ds=10 dt=VARCHAR}$,
  ${SALARY}|4800|BigDecimal|nullable ds=10 dt=NUMERIC}$,
  ${COMMISSION_PCT}|(null)|BigDecimal|nullable ds=4 dt=NUMERIC}$,
  ${MANAGER_ID}|103|BigDecimal|nullable ds=7 dt=NUMERIC}$,
  ${DEPARTMENT_ID}|60|BigDecimal|nullable ds=5 dt=NUMERIC}$
)

```

This example use the full capabilities of variables. This example is generated by the **Script to SQL Commander->INSERT COPY INTO TABLE** right click menu choice in the **Data** tab grid. By default it generates variables representing the actual values and the characteristics of the columns.

Key	Name	Value	Type
	EMPLOYEE_ID	105	BigDecimal
	FIRST_NAME	David	String
	LAST_NAME	Austin	String
	EMAIL	DAUSTIN	String
	PHONE_NUMBER	590.423.4569	String
	HIRE_DATE	2005-06-25 00:00:00	Timestamp
	JOB_ID	IT_PROG	String
	SALARY	4800	BigDecimal
	COMMISSION_PCT		BigDecimal
	MANAGER_ID	103	BigDecimal
	DEPARTMENT_ID	60	BigDecimal

1: **EMPLOYEE\_ID** NUMERIC Format: Unformatted  
 Not NULL, Key Column  
 JDBC: N/A, Java: BigDecimal

Show SQL Continue Cancel

To highlight that a variable is part of the **WHERE** clause in the final SQL, it is represented with a green symbol in front of the name.

When executing an SQL statement that consist of variables, DbVisualizer replaces each variable with either the value as a literal or as a parameter marker. Using parameter markers to pass data with a statement is more reliable than literals. DbVisualizer will automatically generate a parameter marker if the variable has the data type set and if there is no **nobind** option specified.

The following will be replaced with a parameter marker:

```
`${Name}|rolle|String}`
```

These will be replaced with the value as a literal in the final SQL:

```
`${Name}|rolle}`  
`${Name}|rolle|String|nobind}`
```

Variables in DbVisualizer may be used anywhere in a statement as long as there is no data type specified.

## Changing the Delimiter Characters

You can change which identifiers should be used as the prefix, suffix and part delimiter in a variable expression in **Tools->Tool Properties**, in the **General / Variables** category.