Using Parameter Markers

Parameter markers (also referred as **bind/host variables** or **place holders**) are commonly used in database applications where the SQL is composed of static text combined with values represented as markers instead of actual values. These markers are processed during the preparation of the SQL statement and values are then bound with the markers. Each database has its recommendations for how and when to use parameter markers so this is not further discussed here.

DbVisualizer supports the most common syntaxes for parameter markers to comply with the supported databases. Parameter markers are categorized as either **named** or **unnamed** markers. The following sections explains their respective syntaxes.

It is not possible to mix DbVisualizer variables and parameter markers, or named and unnamed parameter markers, in the same script. If you do, you will only be prompted for values for one type and the execution will fail.

Named Parameter Markers

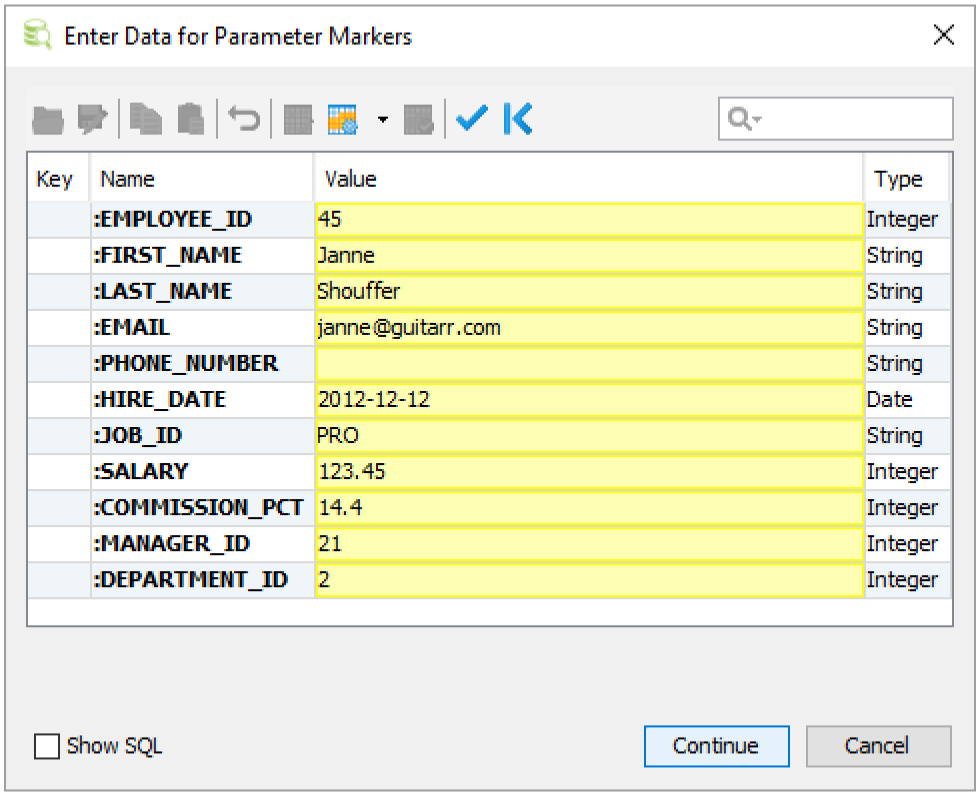
Named Parameter Markers	

`&name`
`:name`
`:{name}`
`:'name'`
	These syntaxes are supported natively by a few databases. This format allows only a name for the parameter and no other settings, such as type or default value. The parameter name is the name DbVisualizer shows in the prompt window.
	
	Named parameter values are bound at runtime with the markers in the SQL. Some JDBC drivers/databases requires that the proper data type is set while some are more relaxed. For named (and unnamed) parameter markers, you may chose data type in the prompt window.
	
	Named parameter markers should only be used in contexts supported by the actual database, usually for column values. For example, as opposed to a DbVisualizer variable, a parameter marker cannot be used for a table or column name.
	
	The only difference between **&name**, **:name**, **:{name}** and **:'name'** is that the latter two, **:{name}** and **:'name'**, allow white spaces in the name.
	
	Example
	
	`insert into EMPLOYEE (ID, FIRST_NAME, LAST_NAME, ADDRESS, AGE)`
	`values (null, &FirstName, &LastName, &Address, &Age);`
	
	`insert into EMPLOYEE (ID, FIRST_NAME, LAST_NAME, ADDRESS, AGE)`
	`values (null, :FirstName, :LastName, :Address, :Age);`
	
	`insert into EMPLOYEE (ID, FIRST_NAME, LAST_NAME, ADDRESS, AGE)`
	`values (null, :{FirstName}, :{LastName}, :{Address}, :{Age});`
	
	Read more about named parameter markers.

The following is a sample SQL executed in the SQL Commander:

```
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);
```

The prompt window will show the markers with their respective names:



Key	Name	Value	Type
	:EMPLOYEE_ID	45	Integer
	:FIRST_NAME	Janne	String
	:LAST_NAME	Shouffer	String
	:EMAIL	janne@guitarr.com	String
	:PHONE_NUMBER		String
	:HIRE_DATE	2012-12-12	Date
	:JOB_ID	PRO	String
	:SALARY	123.45	Integer
	:COMMISSION_PCT	14.4	Integer
	:MANAGER_ID	21	Integer
	:DEPARTMENT_ID	2	Integer

☐ Show SQL

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

To apply the values, close the window and continue with the execution, use key binding **Ctrl+Enter** (**Command+Enter** on macOS).

Unnamed Parameter Markers

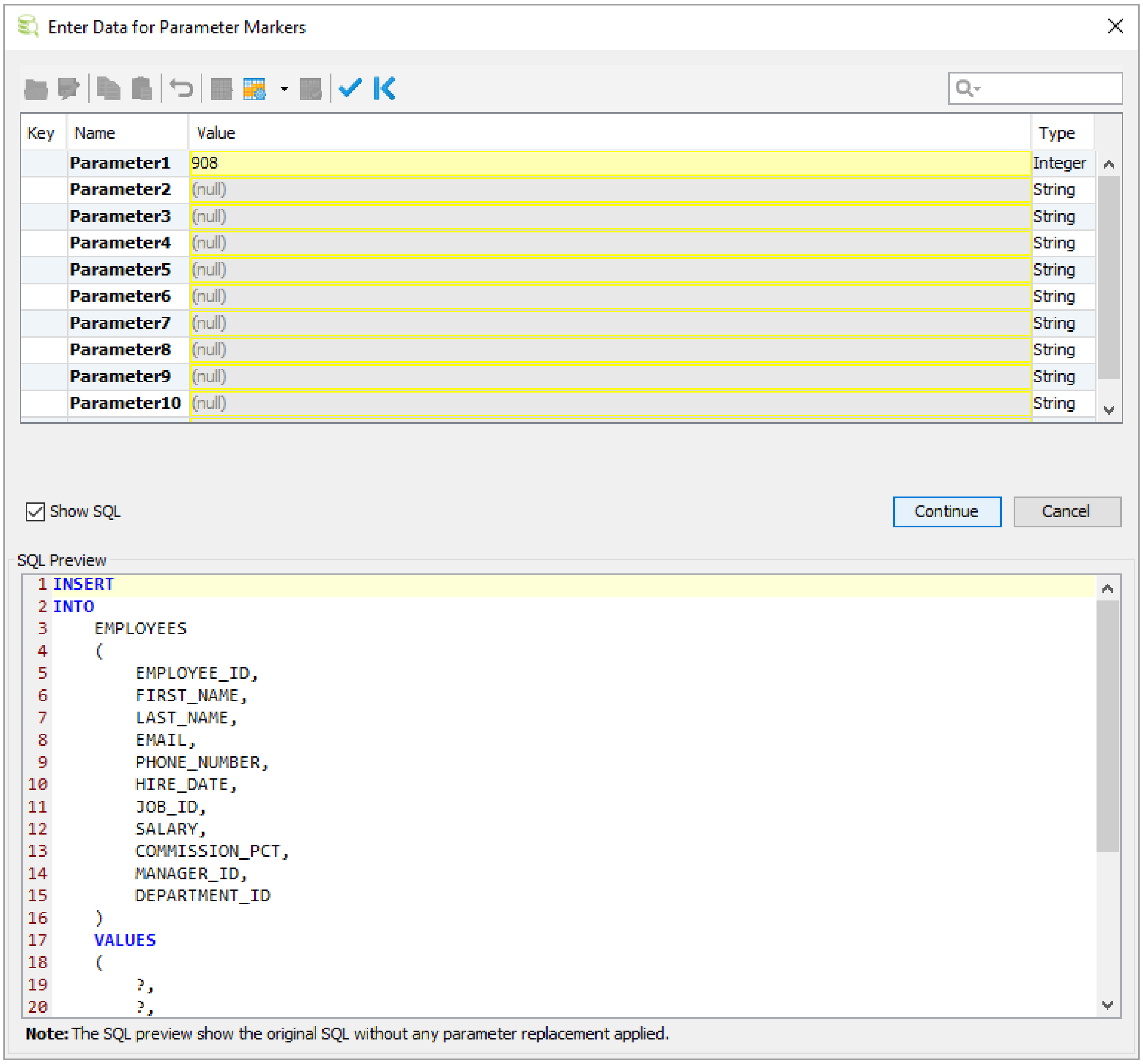
Unnamed Parameter Markers	

?	The question marker symbol is probably the most supported parameter marker among the supported databases. It is also the most unintuitive marker since the user has to remember the order of question marks and the corresponding values.
	
	Since there is no name associated with it, DbVisualizer shows these as **Parameter 1**, **Parameter 2** and so on in the prompt window.
	
	There is no technical difference between how unnamed and named parameter markers are handled internally in DbVisualizer or when processed by the database. All are bound with a prepared SQL statement.
	
	Example
	
	`insert into EMPLOYEE (ID, FIRST_NAME, LAST_NAME, ADDRESS, AGE)`
	`values (null, ?, ?, ?, ?)`
	
	Use named in favor of unnamed parameter markers if there is support in the target database based on the easier reading of named markers. Read more about unnamed parameter markers.

This is the same SQL as used in the Named Parameter Marker section but here question marks are used as markers:

```
INSERT INTO EMPLOYEES
  (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE,
   JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, DEPARTMENT_ID)
VALUES
  (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

Since we're using the unnamed marker, **?**, the name of each parameter is displayed as **Parameter 1**, **Parameter 2** and so on:


Enter Data for Parameter Markers

Key	Name	Value	Type
	Parameter1	908	Integer
	Parameter2	(null)	String
	Parameter3	(null)	String
	Parameter4	(null)	String
	Parameter5	(null)	String
	Parameter6	(null)	String
	Parameter7	(null)	String
	Parameter8	(null)	String
	Parameter9	(null)	String
	Parameter10	(null)	String

☑ Show SQL

SQL Preview

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

Note: The SQL preview show the original SQL without any parameter replacement applied.

Due to the use of unnamed markers it is not very intuitive what parameter correspond to which part in the statement. The **SQL Preview** may be handy to get an idea. The **Type** field is automatically adjusted based on what data is entered for a value. The data type may be manually set by left-click on the type and choose another type from the drop-down.

To apply the values, close the window and continue with the execution, use key binding **Ctrl+Enter** (**Command+Enter** on macOS).

Get Parameter Types via JDBC

The processing of named and unnamed parameter markers is managed by DbVisualizer. By default there is no data type detection of the target columns identified by the markers and DbVisualizer will initially present these as **String** in the prompt window. When changing the value for a parameter in the prompt window, a data analyzer is triggered which will automatically detect the type and update the **Type** field accordingly.

Some drivers (far from all) have the capability to detect the real data type for the referenced columns in the SQL statement. To enable this processing, select the **Get Parameter Types via JDBC** action in the **SQL Commander** menu. DbVisualizer will then show the correct types in the prompt window.

Having **Get Parameter Types via JDBC** enabled while executing may decrease performance substantially as each SQL statement in the script is then pre-processed with the database before the prompt window is displayed.

Database Support/Driver Support

The support for parameter markers may differ between databases. Please consult the documentation for the database to see what syntax it supports.

In some situations the database and DbVisualizer support for named parameters might be incompatible. An example is when using the same parameter name in multiple places in the SQL. When preparing a statement towards such a database, the database may report that the parameter is only used once. In these cases, DBVisualizer will trust the driver and revert to generating the names visible in the form as Parameter 1 , Parameter 2 and so on.