


Command Line Interface

Only in DbVisualizer Pro

 This feature is only available in the DbVisualizer Pro edition.

In addition to the DbVisualizer GUI tool, there is also a pure command line interface for running scripts. We recommend that you use this interface for tasks that you schedule via the operating system's scheduling tool, or when you need to include database tasks in a command script for a larger job. It is also the right tool for execution of large scripts, such as a script generated by the DbVisualizer Export Schema feature.

Don't forget to check all [client-side commands](#) that can also be used in the scripts executed with the command-line interface.

- [Command Line Options](#)
- [Examples](#)
 - [Executing single statements](#)
 - [Executing scripts](#)
 - [Controlling the output](#)
 - [Using variables - prompting for values](#)
 - [Combining OS scripts, the command line interface and DbVisualizer variables](#)
- [Setting up the connection properties on the command line](#)
- [Exit codes from dbviscmd](#)
- [Generating a Command From SQL Commander](#)

On Windows and Linux/Unix, you find this command as a BAT file (*dbviscmd.bat*) or a shell script (*dbviscmd.sh*) in the DbVisualizer installation directory. For macOS, the shell script is located in */Application/DbVisualizer-<Version>.app/Contents/Resources/app*.

Command Line Options

The command line interface supports the following options:

```

Usage: dbviscmd (-connection <name> |
  -url <URL> -drivername <name> |
  -url <URL> -driverclass <name> -driverpath <p1:p2...> )
  [-userid <userid>] [-password <password>]
  [-masterpw <password>]
  (-sql <statements> |
  -sqlfile <filename> [-encoding <encoding>] )
  [-catalog <catalog>] [-schema <schema>]
  [-maxrows <max>] [-maxchars <max>]
  [-stoponerror] [-stoponsqlwarning] [-stoponnorows]
  [-stripcomments (true | false)]
  [-processvariables]
  [-emptypromptvalue <string>]
  [-listconnections]
  [-output (all | none | log | result)] [-outputfile <filename>]
  [-debug [-debugfile <filename>]]
  [-erroridir <directory>]
  [-prefsdir <directory>]
  [-help] [-version]

```

Options:

-connection <name>	Database connection name (created with the GUI)
-url <URL>	Database URL
-drivername <name>	Database driver name (created with the GUI)
-driverclass <name>	Full name of the JDBC Driver class name
-driverpath <p1:p2...>	Paths to the jar files constituting the JDBC driver. Each path separated by a ":" on Linux/macOS and ";" on Windows.
-userid <userid>	Userid to connect as.
-password <password>	Password for userid.
-masterpw <password>	Master Password for encrypted database passwords
-sql <statements>	One or more delimited SQL statements
-sqlfile <filename>	SQL script file to execute
-encoding <encoding>	Encoding for the SQL script file
-catalog <catalog>	Catalog to use for unqualified identifiers
-schema <schema>	Schema to use for unqualified identifiers
-maxrows <max>	Maximum number of rows to display for a result set
-maxchars <max>	Maximum number of characters to display for a column
-stoponerror	Stop execution when getting an error
-stoponsqlwarning	Stop execution when getting an SQL warning
-stoponnorows	Stop execution when empty result set or no affected rows
-stripcomments <true/false>	Strip comments before sending to database. Default is the setting made in the GUI
-output <out>	"all" (default), output both log msgs and result sets "none", suppress both log messages and result sets "log", output only log messages "result", output only result sets
-outputfile <filename>	Script execution output file. Default is stdout
-processvariables	Process variables
-emptypromptvalue <string>	String to use when entering an empty value when prompted for variable(s)
-listconnections	Lists all database connections
-debug	Write debug messages
-debugfile <filename>	File for debug messages. Default is stderr
-erroridir <directory>	Use an alternate location for error logs
-prefsdir <directory>	Use an alternate user preferences directory
-help	Display this help
-version	Show version info

There are two options to specify which database to connect to:

- Using a connection already defined by the DbVisualizer tool. This is done by using the `-connection` parameter. If you have forgot the connection name, use the `-listconnections` option to get a list of all existing names.
- Specifying the connection properties by using the parameter `-url`. The `-url` parameter specifies the JDBC URL for the database to connect to. For information about the JDBC url see [Setting Up a Connection Manually](#). There are also some examples below showing how to specify connection properties using the `-url` parameter

Examples

Executing single statements

You can use the command line interface to execute a single SQL statement:

```
> dbviscmd.bat -connection "Oracle" -sql "select * from hr.countries"
14:34:48 START Executing Command Line, Database Connection: Oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
14:34:48 INFO Physical database connection acquired for: Oracle
COUNTRY_ID COUNTRY_NAME REGION_ID
-----
AR Argentina 2
AU Australia 3
BE Belgium 1
BR Brazil 2
CA Canada 2
CH Switzerland 1
CN China 3
DE Germany 1
DK Denmark 1
EG Egypt 4
FR France 1
HK HongKong 3
IL Israel 4
IN India 3
IT Italy 1
JP Japan 3
KW Kuwait 4
MX Mexico 2
NG Nigeria 4
NL Netherlands 1
SG Singapore 3
UK United Kingdom 1
US United States of America 2
ZM Zambia 4
ZW Zimbabwe 4
14:34:48 SUCCESS [SELECT - 25 rows, 0.007 secs] Result set fetched
select * from hr.countries;
14:34:48 END Execution 1 statement(s) executed, 25 row(s) affected, exec/fetch time: 0.007/0.005 secs [1
successful, 0 errors]
```

If you like to execute just a few statements, you can pass in a list of statements:

```

> dbviscmd.bat -connection "Oracle" -sql "select * from hr.countries; select * from hr.regions"
14:42:21 START Executing Command Line, Database Connection: Oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
14:42:21 INFO Physical database connection acquired for: Oracle
COUNTRY_ID COUNTRY_NAME REGION_ID
-----
AR Argentina 2
AU Australia 3
BE Belgium 1
BR Brazil 2
CA Canada 2
CH Switzerland 1
CN China 3
DE Germany 1
DK Denmark 1
EG Egypt 4
FR France 1
HK HongKong 3
IL Israel 4
IN India 3
IT Italy 1
JP Japan 3
KW Kuwait 4
MX Mexico 2
NG Nigeria 4
NL Netherlands 1
SG Singapore 3
UK United Kingdom 1
US United States of America 2
ZM Zambia 4
ZW Zimbabwe 4
14:42:21 SUCCESS [SELECT - 25 rows, 0.004 secs] Result set fetched
select * from hr.countries;
REGION_ID REGION_NAME
-----
5 Australia
6 South America
1 Europe
2 Americas
3 Asia
4 Middle East and Africa
14:42:21 SUCCESS [SELECT - 6 rows, 0.003 secs] Result set fetched
select * from hr.regions;
14:42:21 END Execution 2 statement(s) executed, 31 row(s) affected, exec/fetch time: 0.007/0.002 secs [2
successful, 0 errors]

```

Executing scripts

If you frequently want to execute a number of statements, it's best to put them into a script file. Here's how to execute a script that contains the two statements from the example above:

```

> dbviscmd.bat -connection "Oracle" -sqlfile "myscript.sql"

14:45:11 START Executing Command Line, Database Connection: Oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
14:45:11 INFO Physical database connection acquired for: Oracle
COUNTRY_ID COUNTRY_NAME REGION_ID
-----
AR Argentina 2
AU Australia 3
BE Belgium 1
BR Brazil 2
CA Canada 2
CH Switzerland 1
CN China 3
DE Germany 1
DK Denmark 1
EG Egypt 4
FR France 1
HK HongKong 3
IL Israel 4
IN India 3
IT Italy 1
JP Japan 3
KW Kuwait 4
MX Mexico 2
NG Nigeria 4
NL Netherlands 1
SG Singapore 3
UK United Kingdom 1
US United States of America 2
ZM Zambia 4
ZW Zimbabwe 4
14:45:11 SUCCESS [SELECT - 25 rows, 0.004 secs] Result set fetched
select * from hr.countries;
REGION_ID REGION_NAME
-----
5 Australia
6 South America
1 Europe
2 Americas
3 Asia
4 Middle East and Africa
14:45:11 SUCCESS [SELECT - 6 rows, 0.003 secs] Result set fetched
select * from hr.regions;
14:45:11 END Execution 2 statement(s) executed, 31 row(s) affected, exec/fetch time: 0.007/0.002 secs [2
successful, 0 errors]

```

Controlling the output

You can use options to control how much output to generate. If you only want to see the results, use the `-output` option with the result keyword:

```
> dbviscmd.bat -connection "Oracle" -sqlfile "myscript.sql" -output result
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2
CH	Switzerland	1
CN	China	3
DE	Germany	1
DK	Denmark	1
EG	Egypt	4
FR	France	1
HK	HongKong	3
IL	Israel	4
IN	India	3
IT	Italy	1
JP	Japan	3
KW	Kuwait	4
MX	Mexico	2
NG	Nigeria	4
NL	Netherlands	1
SG	Singapore	3
UK	United Kingdom	1
US	United States of America	2
ZM	Zambia	4
ZW	Zimbabwe	4

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

For other scripts, for instance a script containing INSERT statements, you may only want to see the log messages:

```
> dbviscmd.bat -connection "Oracle" -sqlfile "myscript.sql" -output log
14:25:29 START Executing Command Line, Database Connection: Oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
14:25:30 INFO Physical database connection acquired for: Oracle
14:25:30 SUCCESS [SELECT - 25 rows, 0.012 secs] Result set fetched
select * from hr.countries;
14:25:30 SUCCESS [SELECT - 4 rows, 0.009 secs] Result set fetched
select * from hr.regions;
14:25:30 END Execution 2 statement(s) executed, 29 row(s) affected, exec/fetch time: 0.021/0.004 secs [2
successful, 0 errors]
```

Using variables - prompting for values

The DbVisualizer command line execution supports the DbVisualizer Variables as described in [Using DbVisualizer Variables](#). To enable this you will need to use the option `-processvariables`.

```

> dbviscmd.bat -connection "Oracle" -sql "SELECT FIRST_NAME FROM HR.EMPLOYEES where FIRST_NAME LIKE ${Name}|A%
||String||};" -processvariables
dbviscmd: Valid inputs: Enter '_B_' for empty, '(null)' for null
dbviscmd: Variable 'Name' (Literal) [A%]: B%
11:25:26 START Executing Command Line for: 'Oracle' [Oracle], Schema: SYSTEM
11:25:26 INFO Physical database connection acquired for: Oracle
FIRST_NAME
-----
Bruce
Britney
11:25:26 SUCCESS [SELECT - 2 rows, 0.047 secs] Result set fetched
SELECT FIRST_NAME, LAST_NAME, PHONE_NUMBER FROM HR.EMPLOYEES where EMPLOYEES.FIRST_NAME LIKE B%;
11:25:26 END Execution 1 statement(s) executed, 2 row(s) affected, exec/fetch time: 0.047/0.000 secs [1
successful, 0 errors]

```

Combining OS scripts, the command line interface and DbVisualizer variables

For more complex tasks, you can call the command line interface from a shell script, for instance a Bourne shell script on Unix or a BAT file on Windows. You can also use DbVisualizer variables to pass information between the shell script and the SQL script. In this example, we have a simple SQL script (*cmdtest.sql*) that contains a SELECT statement with a variable in place for the table name:

cmdtest.sql

```
select * from ${table}$
```

A text file (*tables.txt*) contains the table names we want to execute the SQL script with:

tables.txt

```
hr.countries
hr.regions
```

In a command shell (Bourne or Bash), we can then execute the script using the table names from the text file:

```

for name in `cat tables.txt`;
do ./dbviscmd.sh -connection "oracle" -sql "@run cmdtest.sql \${table}||\${name}|||nobind}\$";
done

15:01:19 START Executing Command Line, Database Connection: oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
15:01:20 INFO Physical database connection acquired for: oracle
15:01:20 RUNNING [@run ...ntries|||nobind}$ - - secs]
@run cmdtest.sql \${table}||hr.countries|||nobind}$;
COUNTRY_ID COUNTRY_NAME REGION_ID
-----
AR Argentina 2
AU Australia 3
BE Belgium 1
BR Brazil 2
CA Canada 2
CH Switzerland 1
CN China 3
DE Germany 1
DK Denmark 1
EG Egypt 4
FR France 1
HK HongKong 3
IL Israel 4
IN India 3
IT Italy 1
JP Japan 3
KW Kuwait 4
MX Mexico 2
NG Nigeria 4
NL Netherlands 1
SG Singapore 3
UK United Kingdom 1
US United States of America 2
ZM Zambia 4
ZW Zimbabwe 4
15:01:20 SUCCESS [SELECT - 25 rows, 0.016 secs] Result set fetched
select * from hr.countries;
15:01:20 SUCCESS [@run ...ntries|||nobind}$ - 0.016 secs] Script processed
@run cmdtest.sql \${table}||hr.countries|||nobind}$;
15:01:20 END Execution 1 statement(s) executed, 25 row(s) affected, exec/fetch time: 0.016/0.012 secs [1
successful, 0 errors]
java -cp /Users/ulf/work/github/dbvis/trunk/pureit/apps/dbvis/classes:/Users/ulf/work/github/dbvis/trunk/pureit
/apps/dbvis/resources:/Users/ulf/work/github/dbvis/trunk/pureit/apps/dbvis/external/* -Xmx512M -Djava.awt.
headless=true -Ddbvis.home=/Users/ulf/work/github/dbvis/trunk/pureit/apps/dbvis com.onseven.dbvis.
DbVisualizerCmd -masterpw stairway -connection oracle -sql @run cmdtest.sql \${table}||hr.regions|||nobind}$;
15:01:23 START Executing Command Line, Database Connection: oracle Database Type: ORACLE Catalog: null Schema:
SYSTEM
15:01:23 INFO Physical database connection acquired for: oracle
15:01:23 RUNNING [@run ...egions|||nobind}$ - - secs]
@run cmdtest.sql \${table}||hr.regions|||nobind}$;
REGION_ID REGION_NAME
-----
5 Australia
6 South America
1 Europe
2 Americas
3 Asia
4 Middle East and Africa
15:01:23 SUCCESS [SELECT - 6 rows, 0.001 secs] Result set fetched
select * from hr.regions;
15:01:23 SUCCESS [@run ...egions|||nobind}$ - 0.001 secs] Script processed
@run cmdtest.sql \${table}||hr.regions|||nobind}$;
15:01:23 END Execution 1 statement(s) executed, 6 row(s) affected, exec/fetch time: 0.001/0.000 secs [1
successful, 0 errors]

```

The command line interface is called with the `-sql` option, specifying the [client-side command @run](#). A [DbVisualizer variable](#) is passed to the `@run` command with the value taken from the shell variable. This DbVisualizer variable value is then available to the script executed by the `@run` command.

Note that you may need to escape certain characters that the shell would otherwise interpret, like the dollar signs that are part of the DbVisualizer variable delimiters.

Setting up the connection properties on the command line

As an alternative to using a connection already set-up through the DbVisualizer tool you may use the `-url` parameter. In combination with the parameters `-drivername`, `-driverclass`, and `-driverpath` these parameters enables you connect without prior specification using the DbVisualizer tool.

Following are some examples.

Executing SQL towards a MySQL instance running on `localhost` port `3306`. The parameter `"-drivername MYSQL"` specifies that we are using a JDBC driver specified in the DbVisualizer tool named `MYSQL`. For listing of the existing drivers use the **Tools->Driver Manager** in the DbVisualizer tool. Read more in [Installing a JDBC Driver](#).

Using `-url` and `-drivername` parameters

```
./dbviscmd.sh -url jdbc:mysql://localhost:3306/
  -drivername MYSQL
  -sql "select * from sakila.actor"
  -userid root
```

An alternative to use the `-drivername` you may use the parameters `-driverclass` and `-driverpath` to specify the JDBC driver.

Using `-driverclass` and `-driverpath` parameters

```
./dbviscmd.sh -url jdbc:oracle:thin:@localhost:1521/ORCL
  -driverclass oracle.jdbc.OracleDriver
  -driverpath "ojdbc6.jar:orai18n.jar:xdb.jar:xmlparserv2.jar"
  -sql "select * from HR.COUNTRIES"
  -userid system
  -password oracle
```

The above example connects to an Oracle instance on `localhost` and port `1521`. Note that the separator character `:"` between the different jar files is platform dependant. On Windows-based desktop platforms, the value of this field is the semicolon `;"`.

Exit codes from `dbviscmd`

These are the exit codes when running `dbviscmd`

Code	Meaning
0	OK
1	Other error
2	Connect error
3	Script execution resulted in an error. Execution stopped
4	Script execution resulted in errors
5	Script execution failed unexpectedly

Generating a Command From SQL Commander

From time to time, you may want to use some SQL that you have tested in the GUI and run it in the command-line interface. You can generate the string with all parameters and paste that string on the command line or a script. Use menu option **SQL Commander -> Generate Command for dbviscmd** to open a dialog where you can specify the options. Some options are selected and some are disabled depending on the editor contents.

If you generate a command from an anonymous script (a script that has not been saved to file) with more than one line, you will be prompted to save the file or to generate a temporary file. If you generate the command from a file that is modified, you will be prompted to save the file before generating the command.

Besides the [Command Line Options](#), there are a few settings in the dialog:

- You can use a predefined (named) or an anonymous connection; this will enable and disable connection options accordingly.

- You can choose whether or not to **Include SQL Commander Options**; this is a convenient way to enable or disable all corresponding options. You can still enable or disable individual options as desired.
- You can choose whether or not to **Include Output Options**; this is a convenient way to enable or disable all corresponding options. You can still enable or disable individual options as desired.

Generate Command for dbviscmd [2: Sakila.customerList.sql]
✕

Generate a command line string for use with the command-line interface, **dbviscmd** (see [DbVisualizer Users Guide](#) for details). Select relevant options and **copy** to system clipboard when done.

Use Predefined Connection Use Anonymous Connection
 Include SQL Commander Options Include Output Options

Options

Include	Option	Value	Description
<input checked="" type="checkbox"/>	-connection <name>	Sakila H2 (dbvis)	Database connection name (created with the GUI)
<input type="checkbox"/>	-url <URL>	jdbc:h2:D:\code\git.root\DbVis\dbvisualizer\pureit\apps\proto>re\main\java\com\enseven\dbvis\db\h2\sakila\sakila-dbvis	Database URL
<input type="checkbox"/>	-drivername <name>	H2-embedded	Database driver name (created with the GUI)
<input type="checkbox"/>	-driverclass <name>	org.h2.Driver	Full name of the JDBC Driver class name
<input type="checkbox"/>	-driverpath <p1;p2...>	D:\Program Files\DbVisualizer\jdbc\h2\h2.jar	Paths to the jar files constituting the JDBC driver. Each path separated by a ";"
<input type="checkbox"/>	-userid <userid>	<userid>	UserId to connect as
<input type="checkbox"/>	-password <password>	#UNDEFINED#	Password for userid
<input type="checkbox"/>	-masterpw <password>	<password>	Master Password for encrypted database passwords
<input type="checkbox"/>	-sql <statements>	N/A (multi-line statement)	One or more delimited SQL statements
<input checked="" type="checkbox"/>	-sqlfile <filename>	C:\Users\wti user\,dbvis\Bookmarks\QB\Sakila.customerList.sql	SQL script file to execute
<input checked="" type="checkbox"/>	-encoding <encoding>	UTF-8	Encoding for the SQL script file
<input type="checkbox"/>	-catalog <catalog>	<catalog>	Catalog to use for unqualified identifiers
<input type="checkbox"/>	-schema <schema>	SAKILA	Schema to use for unqualified identifiers
<input type="checkbox"/>	-maxrows <max>	-1	Maximum number of rows to display for a result set
<input type="checkbox"/>	-maxchars <max>	-1	Maximum number of characters to display for a column
<input type="checkbox"/>	-stoponerror		Stop execution when getting an error
<input type="checkbox"/>	-stoponsqlwarning		Stop execution when getting an SQL warning
<input type="checkbox"/>	-stoponnorows		Stop execution when empty result set or no affected rows
<input type="checkbox"/>	-stripcomments <true/false>	false	Strip comments before sending to database. Default is the setting made in the GUI
<input type="checkbox"/>	-processvariables		Process variables
<input type="checkbox"/>	-emptypromptvalue <string>	<string>	String to use when entering an empty value when prompted for variable(s)
<input type="checkbox"/>	-output <out>	<out>	"all" (default), output both log msgs and result sets "none", suppress both log messages and result sets "log", output only log messages "result", output only result sets
<input type="checkbox"/>	-outputfile <filename>	<filename>	Script execution output file. Default is stdout
<input type="checkbox"/>	-debug		Write debug messages
<input type="checkbox"/>	-debugfile <filename>	<filename>	File for debug messages. Default is stderr
<input type="checkbox"/>	-erroridir <directory>	<directory>	Use an alternate location for error logs

Preview

Wrap Lines

```

D:\code\git.root\DbVis\dbvisualizer\pureit\apps\dbvis\bin\dbviscmd
-connection "Sakila H2 (dbvis)"
-sqlfile "C:\Users\wti user\,dbvis\Bookmarks\QB\Sakila.customerList.sql"
-encoding "UTF-8"
  
```