SQL SERVER – USING THE SQLCMD COMMAND LINE UTILITY

The **sqlcmd** command line utility enables us to interact with **SQL Server** from the command line. It can:

- Define connection setting information
- Execute **Transact-SQL** (**T-SQL**) statements
- Call external scripts
- Use environment variables
- Store the output results of executed queries in a specified text file

sqlcmd gives us the functionality (and more) of the ***osql*** and ***isql*** command line utitlities that were present in **SQL 2000** but ***have now been deprecated***. It can also connect to previous versions of SQL Server such as **SQL Server 2000**.

The options for executing sqlcmd are flexible, it can be executed:

- In a **.bat** file;
- Run interactively from the command line;
- Called directly from the command line using **SQLCMD** with command line arguments;

Syntax

```
Sqlcmd            [-U login id]          [-P password]
  [-S server]            [-H hostname]          [-E trusted connection]
  [-d use database name] [-l login timeout]     [-t query timeout]
  [-h headers]           [-s colseparator]      [-w screen width]
  [-a packetsize]        [-e echo input]        [-I Enable Quoted Identifiers]
  [-c cmdend]            [-L[c] list servers[clean output]]
  [-q "cmdline query"]   [-Q "cmdline query" and exit]
  [-m errorlevel]        [-V severitylevel]     [-W remove trailing spaces]
  [-u unicode output]    [-r[0|1] msgs to stderr]
  [-i inputfile]         [-o outputfile]        [-z new password]
  [-f  | i:[,o:]] [-Z new password and exit]
  [-k[1|2] remove[replace] control characters]
  [-y variable length type display width]
  [-Y fixed length type display width]
  [-p[1] print statistics[colon format]]
  [-R use client regional setting]
  [-b On error batch abort]
  [-v var = "value"...]  [-A dedicated admin connection]
  [-X[1] disable commands, startup script, enviroment variables [and exit]]
  [-x disable variable substitution]
  [-? show syntax summary]
```

Examples of sqlcmd in Use

Running interactively from the command prompt

On your machine that runs **SQL Server** go to the windows **Start Menu** and select **Run**. Type in**cmd** and click **OK**. A command prompt window will now open.

First we type in **sqlcmd** along with the database server] connection settings. By default ***sqlcmd*** uses windows authentication for authenticating to **SQL Server** so we can omit the connection parameters to connect using windows authentication to the default instance of SQL Server on the machine. Alternatively we can use the **-S** parameter to specify the Server name and instance name we want to connect to.

Once we have typed **sqlcmd** and pressed ***'Enter'*** we will then be given a line number prompt: **>1**.

Now we can enter a number of **T-SQL** statemetents (pressing ***'Enter'*** after each one to give a new line number prompt). These T-SQL statements are not executed until the ***'GO'*** command is entered on a new line and the ***'Enter'*** key is pressed. This enables batches of statements to be executed just as in standard **SQL**.

The following example connects to a server called ***MYSERVER*** and to an instance of the database engine on that server called ***SQL2005*** and then runs a batch of SQL statements:

```
C:\Windows>sqlcmd -S MYSERVER\SQL2005
1> USE MyTestDB
2> SELECT Name FROM Customers
3> SELECT TOP 1 Telephone FROM Customers
4> GO
Changed database context to 'MyTestDB'.
Name
--------------------------------------------------
Joe Bloggs
Bob Marley
Jimi Hendrix

(3 rows affected)
Telephone
------------------------------
020 2999999
(1 rows affected)
1>
```

Writing the results output to a text file

From the example above we can see that the results of the queries are displayed in the command window. We can output these results to a text file by using the optional **-o** parameter. If we also want to write the queries to the output results we can use the **-e** parameter which echoes the queries to the output results.

For the above example, if we wanted to write the queries and their results to a text file located at **C:\MyResults.txt** we would just amend the first line to read:

```
C:\Windows>sqlcmd -S MYSERVER\SQL2005 -o C:\MyResults.txt -e
```

Using a SQL Server username and password

If you want to connect to the database engine using a SQL Server login rather than using windows authentication you can use the **-U** parameter for the login ID and the **-P** parameter for the password:

```
C:\Windows>sqlcmd -S MYSERVER\SQL2005  -U sa -P MyPassword
```

Specifying an external script to run

Rather than typing out all our script commands in the command prompt we can use the **-i**parameter to tell sqlcmd to use an input file which contains script that may include T-SQL statements along with sqlcmd non T-SQL commands. This is especially useful when the script length gets long as it can save us typing out into a command prompt. It also enables us to code the script in the user friendly graphical environment of **SQL Server Management Studio** (*SSMS*) where we can take advantage of colour coding and syntax checking.

When writing sqlcmd scripts in *SSMS* you must enable the **SQLCMD Mode** icon in the query toolbar. You will then get the support for the non **T-SQL sqlcmd** commands, color coding and syntax highlighting

For our example say we have the following script located at **C:\MyScript.sql**

```
USE MYTestDB
SELECT TOP 1 * FROM Customers ORDER BY CustomerID DESC
INSERT INTO Customers VALUES ('Bill Gates', '020 222222')
SELECT @@IDENTITY As 'Last ID Entered'
```

The following example runs the **C:\MyScript.sql** script from the command line and outputs the queries and results to a text file located at C:\MyResults.txt.

```
C:\Windows>sqlcmd -S NOTTINTRA3\SQL2005 -i C:\MyScript.sql -o
C:\MyResults.txt -e
```

We can now see one of the great benefits of sqlcmd in that by putting our sqlcmd commands in a .bat file we can easily automate the execution of one or more scripts by running this one batch file.

Using scripting variables with sqlcmd

We can use variables in our sqlcmd input files or scripts. These scripting variables can be declared in the script itself or passed into the command as a parameter.

To declare and assign a value to a variable in the script

Use the **:setvar** *VariableName Value* syntax.

To reference the variable in the script use the following syntax: **$(VariableName)** where***VariableName*** is the name of your variable.

The following example declares a variable called ***SelectTable*** and assigns it the value 'Customers' and then uses this variable in a **SELECT** statement.

```
:SETVAR SelectTable Customers
SELECT * FROM $(SelectTable)
```

The value held in the ***SelectTable*** variable replaces the $(SelectTable) reference so that the SELECT statement above would execute as ***SELECT * FROM Customers***.

To declare and assign a value to a variable with a command line parameter

The **-v** parameter is used to assign variables when using the command line. The following example passes a variable called ***SelectTable*** with a value of ***Customers***:

```
C:\Windows>sqlcmd -i MyScript.sql -v SelectTable = Customers
```

If you want to pass multiple variables using the command line then simply list the variables with no delimiter between them in following fashion (the example is split onto two lines but should actually be on one line:

```
C:\Windows>sqlcmd -i MyScript.sql -v SelectTable = Customers Var2 = 3 Var3 = No
```

SQLCMD extended (non T-TSQL) commands

As well as being able to run **T-SQL** statements, sqlcmd also has its own extended set of commands that can be used in scripts that it executes.

To obtain the list of commands run sqlcmd interactively from the command line and issue the **:Help** command as follows:

```
C:\WINDOWS>sqlcmd
1> :Help
:!! []
  - Executes a command in the Windows command shell.
:connect server[\instance] [-l timeout] [-U user [-P password]]
  - Connects to a SQL Server instance.
:ed
  - Edits the current or last executed statement cache.
:error
```

```
  - Redirects error output to a file, stderr, or stdout.
:exit
  - Quits sqlcmd immediately.
:exit()
  - Execute statement cache; quit with no return value.
:exit()
  - Execute the specified query; returns numeric result.
go []
  - Executes the statement cache (n times).
:help
  - Shows this list of commands.
:list
  - Prints the content of the statement cache.
:listvar
  - Lists the set sqlcmd scripting variables.
:on error [exit|ignore]
  - Action for batch or sqlcmd command errors.
:out |stderr|stdout
  - Redirects query output to a file, stderr, or stdout.
:perftrace |stderr|stdout
  - Redirects timing output to a file, stderr, or stdout.
:quit
  - Quits sqlcmd immediately.
:r
  - Append file contents to the statement cache.
:reset
  - Discards the statement cache.
:serverlist
  - Lists local and SQL Servers on the network.
:setvar {variable}
  - Removes a sqlcmd scripting variable.
:setvar
  - Sets a sqlcmd scripting variable.
1>
```