Working with the SQL Server command line (sqlcmd)

October 18, 2017 by Daniel Calbimonte

Introduction

Sqlcmd allows executing queries, T-SQL sentences and SQL Server scripts using the command line.

In the previous article How to work with the command line and Azure to automate tasks, we worked with the sqlcmd in Azure.

In this new chapter, we will show the following examples in a local SQL Server using sqlcmd:

1. Working with sqlcmd interactive mode including how to
   a. connect to SQL Server
   b. check the current database
   c. list databases
   d. check if the SQL Server is case sensitive
   e. check the SQL Server edition
   f. check the SQL Server Authentication
   g. list the variables set
2. Running sqlcmd in command mode including how to
   a. back up a database
   b. run a T-SQL script and receive the output in a file
   c. work with variables
   d. list the table names of a database
   e. list the column names of a database
   f. check all the commands
   g. exit if the command fails
   h. display error messages according to the error level
   i. accept user input
3. Working in SSMS in sqlcmd mode including how to
   a. run sqlcmd in SSMS
   b. set the sqlcmd mode by default in SSMS

4. Working with PowerShell including how to
    1. invoke sqlcmd using PowerShell
    2. run scripts in SQL PowerShell (check table fragmentation)
    3. output verbose results
5. DAC
    a. How to work with a Dedicated Administrator Connection (DAC)
6. When to use sqlcmd mode, interactive mode, DAC, SSMS, PowerShell

Requirements

1. Sqlcmd installed in a Windows Machine (Linux supports sqlcmd, but it is slightly different).

Getting Started

1. Working with sqlcmd interactive mode

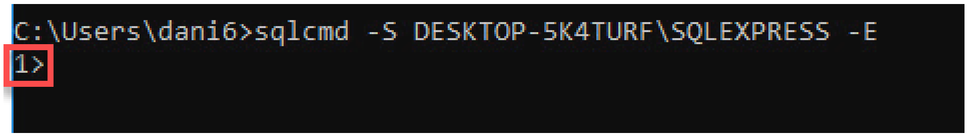
   In interactive mode, you can write the input and interact using the command line.

    a. How to connect to SQL Server using sqlcmd
    To connect to your local machine, specify the SQL Instance name and the credentials:

    sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -E

    The –S value is to specify the SQL Server name of the instance and -E is to specify a trusted connection. If you do not specify the SQL Server name, it will try to connect to the local machine.

    When you connect, you will see the number **1>**:



```
C:\Users\dani6>sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -E
1>
```

    The number 1> means that it is connected and ready to receive sentences to execute.

    If you enabled SQL Server Authentication, you will need to specify a user name and a user password (I am assuming that the user is already created). Note that you will need to **EXIT** of sqlcmd to login with this credential.

    sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -U jsmith

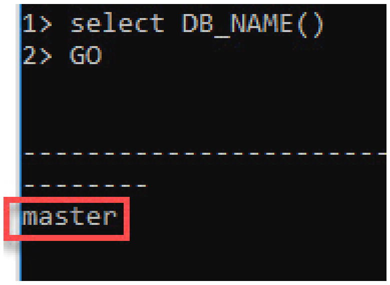
The command line will ask you the password. You can optionally specify the password (not recommended, but sometimes is the only way to work):

sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -U jsmith -P Mypwd$%1234565

b. How to check the current database in sqlcmd

When a SQL Server Login is created, you can define the default database you want to log in. If it is not specified, the master database is the default one.

```
select DB_NAME()
GO
```



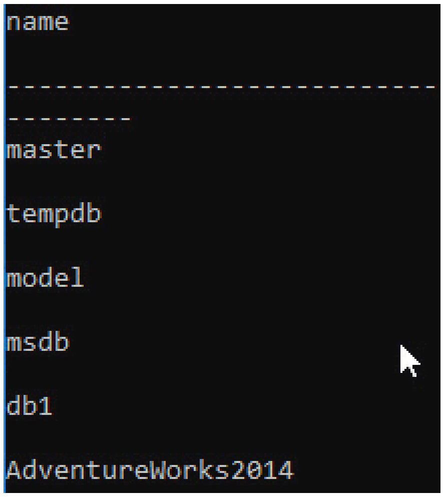
```
1> select DB_NAME()
2> GO

------------------------
--------
master
```

c. How to list the databases in sqlcmd

The following sentences will list the databases in the SQL Instance:

```
select name from sys.databases
go
```

In the sys.databases table, you have all the database information:



```
name

----------------------------
--------
master

tempdb

model

msdb

db1

AdventureWorks2014
```

You can also use the sp_databases stored procedure:

```
Sp_databases
Go
```

d. How to check if the SQL Server is case sensitive in sqlcmd

The following T-SQL Sentences are used to detect the collation information including if the machine is case sensitive or not:

```
SELECT SERVERPROPERTY('COLLATION')
GO
```

The information displayed will be as follows:

Modern_spanish is the collation, CI means case insensitive and CS is case sensitive. AS means Accent Sensitive and AI is Accent Insensitive.

You can also check the information, with the sp_helpsort procedure:

```
sp_helpsort
go
```

The information displayed is the following:

Modern-Spanish, case-insensitive, accent-sensitive, kanatype-insensitive, width-insensitive

e. How to check the SQL Server edition in SQL

You can check the SQL Server Edition, using the following T-SQL sentences:

```
SELECT SERVERPROPERTY('EDITION')
GO
```

The result is the following:

f. How to check the SQL Server Authentication in sqlcmd

Before Azure, there were two options to Authenticate to SQL Server:

i. Windows Authentication where you can use an Active directory account or a local Windows account.
ii. Windows Authentication and SQL Authentication where you can also authenticate using an account created in SQL Server.

To detect the authentication, you can use the following sentences:

```
SELECT SERVERPROPERTY('IsIntegratedSecurityOnly')
GO
```

The result displayed is the following:

If the result is 0, it means that both authentications are enabled. If it is 1, only Windows Authentication is enabled.

g. How to list the variables set

In order to list all the variables set, run the following command in sqlcmd:

:ListVar

It will show all the variables set:

2. Running sqlcmd in command mode

You can run sqlcmd as commands. You can run scripts in command mode.

a. How to run a T-SQL script and receive the output in a file in sqlcmd

In the next example, we will show how to run a script using sqlcmd and show the results in another file.

We will first create a script file named columns.sql with the following sentences:

```
select * from adventureworks2014.information_schema.columns
```

In the cmd, run the following command to invoke sqlcmd:

```
sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -E -i c:\sql\columns.sql -o c:\sql\exit.txt
```

-i is used to specify the input. You specify the script file with the queries.
-o is used to show the results of the input in a file.

The exit.txt file will be created:

If we open the file, we will see the output results:

b. How to back up in sqlcmd

We will first create a script to back up the database named backup.sql:

```
BACKUP DATABASE [AdventureWorks2014] TO  DISK = N'C:\SQL\backup.bak'
GO
```

In the cmd run the following command:

```
sqlcmd -S DESKTOP-5K4TURF\SQLEXPRESS -E -i c:\sql\backup.sql -o c:\sql\output.txt
```

The output will be similar to this one:

The commands will create a backup in a file named backup.sql in the c:\sql folder:

c. How to work with variables in sqlcmd

You can work with variables in sqlcmd. The following example will set the variable DATABASENAME with the value adventureworks2014 and then we change the context to the database specified:

```
:SETVAR DATABASENAME "adventureworks2014"
USE $(DATABASENAME);
```

```
GO
```

The result displayed is the following:

As you can see, SETVAR is used to specify the value of the variable. Then you need to use $() for the variable.

Another example is to set the variable CONTACTTYPEID to 3 and use it in the where clause to find a contact type ID according to the value of the variable:

```
:SETVAR CONTACTTYPEID 3
SELECT [ContactTypeID]
      ,[Name]
      ,[ModifiedDate]
  FROM [Person].[ContactType]
where contacttypeid=$(CONTACTTYPEID)
GO
```

The result displayed is the following:

d. How to list the table names of a database in sqlcmd

You can list the tables of the database using the information_schema.tables view. We will first create a script named tables.sql. This script contains the tables and views:

```
--Script name: tables.sql
select table_name from adventureworks2014.information_schema.tables
GO
```

Next, we will invoke sqlcmd to execute the script.

sqlcmd -E -i c:\sql\tables.sql -o c:\sql\output.txt -S DESKTOP-5K4TURF\SQLEXPRESS

The result displayed are the following in the output.txt file:

e. How to list the column names of a database in sqlcmd

The following sentences will list the table names and the column names of a database in a

The following sentences will list the table names and the column names of a database in a script named columns.sql:

```
--Filename columns.sql
select table_name, column_name from adventureworks2014.information_sch
ema.columns
GO
```

In the cmd run this command:

sqlcmd -E -i c:\sql\columns.sql -o c:\sql\output.txt -S DESKTOP-5K4TURF\SQLEXPRESS

The result of the output.txt is the following:

f. How to check all the commands

You can check all the sqlcmd commands using this command:

Sqlcmd -?

This command will list all the commands available:

g. How to exit if the command fails

The following command will exit if it fails using the –b parameter:

sqlcmd -E -q “create table adventureworks” –b -S DESKTOP-5K4TURF\SQLEXPRESS

The command will exit if there is an error:

h. How to display error messages according to the error level

If there is an error, the error is displayed. However, according to the error level, you can stop this behavior by default using the -m option.

Here it is an example about this:

The following command shows an error message:

However, if you add the –m 16, the error will no longer be displayed because the error has the level of 15:

sqlcmd -E -q "create table adventureworks" -m 16 -S DESKTOP-5K4TURF\SQLEXPRESS

-m 16 will show only the errors higher than 16. As you can see the error message is no longer displayed

i. How to accept user input

The following example will run a SQL script with one variable. The example will create a database specified by the user.

We will first create a script named createdb.sql with the following content:

```
--file createdb.sql
CREATE DATABASE $(DATABASENAME);
GO
```

Next, in the cmd we will run the database specifying the database name:

sqlcmd -E -v DATABASENAME="Userinput" -i c:\sql\createdb.sql

The command will create a database named Userinput.

In sqlcmd you can run the sp_databases stored procedure:

```
Sp_databases
GO
```

And you will be able to see the database created:

3. Working in SSMS in sqlcmd mode

a. How to run sqlcmd in SSMS

Yes, in SSMS, click on your query and select Query>SQLCMD Mode:

The following example will create a database named sales in SSMS.

```
:SETVAR DATABASENAME "sales"
create database $(DATABASENAME);
GO
```

If everything is OK, a database named sales will be created:

b. How can we set the sqlcmd mode by default in SSMS?

Yes, to do this, go to **Tools>Options** in SSMS and check the **By default, open new queries in SQLCMD mode.**

4. Working with PowerShell

a. How to invoke sqlcmd using PowerShell

PowerShell can be used to invoke sqlcmd. To open PowerShell for SQL Server, go to the Windows Search and write sqlps:

In sqlps, write these cmdlets to run the sp_who stored procedure:

invoke-sqlcmd -query “sp_who”

Note that if you have SSMS 17 or later, SQL PowerShell is installed separately. For more information about installing SQL PowerShell, refer to our link:

- What is new in SSMS 17; PowerShell and DAX

b. How to run scripts in SQL PowerShell (check table fragmentation)

It is possible to run SQL Server scripts with PowerShell. The following example will show the fragmentation of the table of the table Person.Address in the Adventureworks database.

We will first create a script named fragmentation.sql:

```
DECLARE @db_id SMALLINT=DB_ID('AdventureWorks');
DECLARE @object_id INT=OBJECT_ID(N'AdventureWorks.Person.Address');

SELECT * FROM sys.dm_db_index_physical_stats(@db_id,
@object_id, NULL, NULL , 'LIMITED');
GO
```

In PowerShell for SQL Server, run this script:

Invoke-sqlcmd –inputfile "c: \sql\fragmentation.sql" | Out-File -filePath "C:\sql\outps.txt"

The output of the outps.txt file will be the following:

c. How to use verbose output

Verbose is used to display information that is not displayed by default. For example, the command print is not displayed by default. Let's take a look to an example.

In sqlps, run this cmdlet:

Invoke-Sqlcmd -Query "PRINT 'HELLO SQLSHACK'"

The cmdlet will not return any value. However, if you run with the parameter verbose, the output can be displayed:

Invoke-Sqlcmd -Query "PRINT 'HELLO SQLSHACK'" –verbose

5. DAC

a. How to work with a Dedicated Administrator Connection (DAC) in sqlcmd

If SQL Server fails to connect in SSMS or other tools, it is possible to try a DAC connection. This connection is connection allows to diagnostic and verify the problems of the

This connection is connection allows to diagnostic and verify the problems of the Database Server. When the SQL Server is corrupt and it is not possible to connect to it, DAC connection usually works.

The following example shows how to connect to a SQL Server database:

```
sqlcmd -S DESKTOP-5K4TURF -E -A -d master
```

-A is used to specify a DAC connection and -d is used to specify the database to connect.

A DAC connection requires the SQL Browser service to be started and enabled. To enable the SQL Browser service, if it is disabled, you can use the following commands:

```
sc config sqlbrowser start=demand
```

If it is enabled, the message will be the following:

To start the service, you can use the following commands:

```
net start sqlbrowser
```

6. When to use sqlcmd mode, interactive mode, DAC, SSMS, PowerShell

Use interactive mode when you need to run multiple queries and administrative tasks. The sqlcmd command line mode is used when you have specific tasks like a backup. Use it when you have a specific task in mind. DAC is used for disaster recovery (for example when the master database is damaged and you cannot access to SQL Server using SSMS or other conventional tools). SSMS in sqlcmd mode can be used to create scripts. It is great to debug and program large scripts to be used later in the command line mode.

Use PowerShell if you have other PowerShell scripts and you need to integrate some sqlcmd invocations to it.

Conclusion

Sqlcmd is a very powerful feature that can help us to automate tasks in SQL Server. You can run scripts and save the results of your queries in a text file.

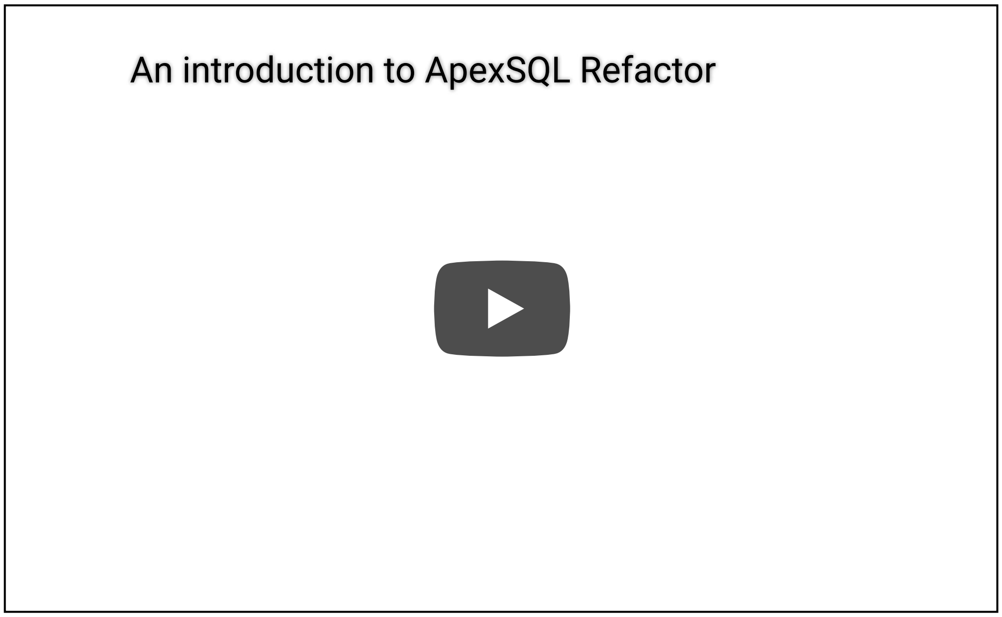
Previous article in this series:

- How to work with the command line and Azure to automate tasks

How to work with the command line and Azure to automate tasks

See more

To boost SQL coding productivity, check out these free SQL tools for SSMS and Visual Studio including T-SQL formatting, refactoring, auto-complete, text and data search, snippets and auto-replacements, SQL code and object comparison, multi-db script comparison, object decryption and more



Daniel Calbimonte

Daniel Calbimonte is a Microsoft Most Valuable Professional, Microsoft Certified Trainer and Microsoft Certified IT Professional for SQL Server. He is an accomplished SSIS author, teacher at IT Academies and has over 13 years of experience working with different databases.

He has worked for the government, oil companies, web sites, magazines and universities around the world. Daniel also regularly speaks at SQL Servers conferences and blogs. He is also a writer for SQL Server training material for certification exams.

View all posts by Daniel Calbimonte

Related Posts:

1. Overview of the SQLCMD utility in SQL Server
2. Introduction to mssql-cli, a new interactive command line tool for SQL Server
3. How to work with the command line and Azure to automate tasks
4. How to work migrate backups, files and scripts to/from the cloud using the command line
5. Working with PowerShell's Invoke-SqlCmd

SQL commands, SQLCMD

76,340 Views

0 Comments **SQL Shack** **John Paul Lim Ga…**

♡ **Recommend** 12 Tweet Share **Sort by Best**

Start the discussion…

Be the first to comment.

ALSO ON **SQL SHACK**

Creating a SQL Server audit using SQL Server Extended Events and Triggers

2 comments • 5 months ago

Avatar **Heisenberg** — Event Notifications are also a good way to automate auditing.

SQL Server in Azure Kubernetes Service (AKS)

2 comments • a month ago

Avatar **Sudheer** — Great article. Steps are in detail

Three Standard SQL Unit Tests you can write against any Stored Procedure

1 comment • 19 days ago

Avatar **Raja Imran Khan** — Very well written article. Your knowledge and application is admirable. Keep sharing your expertise.

SQL Union vs Union All in SQL Server

1 comment • 2 months ago

Avatar **Kannan** — Excellent article.