**Automate Out of Date SQL Server Statistics with PowerShell**

By: [Tim Smith](https://www.mssqltips.com/sqlserverauthor/92/tim-smith/)

FROM: <https://www.mssqltips.com/sqlservertip/4192/automate-out-of-date-sql-server-statistics-with-powershell/>

**Problem**

We use some maintenance tools which tell us when tables need their statistics updated and we wanted to know if we could build some tools to automate updating statistics on tables outside of maintenance schedules, on the basis of when we receive the alerts from these tools. For an example, we might receive a text or email alert telling us that table dbo.Example has statistics that are out-of-date or that the data has heavily fluctuated, and we'd like to automate updating statistics on this object automatically. Our environment allows us to perform updates when they are needed for most of our servers where we monitor the statistics fluctuation.

**Solution**

The answer will vary by tool, since it's being built around a tool:

* Some tools may store the information in a database that allows developers to pull the server name, database name, and table name, allowing developers to run a re-indexing script for all the objects listed, which will minimize the amount of work to update the statistics.
* If a tool does not provide the ability to retrieve this information, developers may look at patterns on the server or database level and run any update statistic maintenance around that time, or build a structure that allows them to save the data from the tool.
* If the tool issues email alerts, developers might consider extracting the email messages and updating statistics based on emails. If the tool saves these alerts to a file, using an ETL function to import the data through the file also is an alternative here.

There are some other alternatives to using tools, which may not allow for automating these tasks, even though automation of these tasks is an easy next step. One popular alternative is to update statistics on a maintenance schedule and this might be an effective approach, relative to the data change and user need. Another alternative for heavy ETL environments is to update statistics based on a specified sample size toward the end of a load (generally, in bigger data environments). For OLTP tables, developers might consider frequent updates on statistics; for an example, on two particular tables, I saw increased performance doing statistics' updates every two hours because of the frequent changes in the table throughout the day. A tool for situations like this may create more problems than it's worth because this is expected behavior based on the environment (heavy ETL or heavy OLTP).

CREATE TABLE tb\_UpdateStatistics(

ServerName VARCHAR(50),

DatabaseName VARCHAR(100),

TableName VARCHAR(250),

Options VARCHAR(2000)

)

We'll use the columns from this table to perform the statistics update:

Function Get-StatParameters {

Param(

[ValidateLength(3,50)][string]$readserver

)

Process

{

$scon = New-Object System.Data.SqlClient.SqlConnection

$scon.ConnectionString = "Data Source=$readserver;Initial Catalog=master;Integrated Security=true;Connection Timeout=45;"

$cmd = New-Object System.Data.SqlClient.SqlCommand

$cmd.Connection = $scon

$cmd.CommandText = "SELECT ServerName, DatabaseName, TableName, Options FROM tb\_UpdateStatistics"

$cmd.CommandTimeout = 45

try

{

$scon.Open()

$sqlread = $cmd.ExecuteReader()

while ($sqlread.Read())

{

[string]$read\_servername = $sqlread["ServerName"]

[string]$read\_databasename = $sqlread["DatabaseName"]

[string]$read\_tablename = $sqlread["TableName"]

[string]$read\_options = $sqlread["Options"]

[string]$command = "UPDATE STATISTICS $read\_tablename $read\_options"

### Either Invoke-SqlCmd or Execute-Sql (build the command)

Execute-Sql -server $read\_servername -database -$read\_databasename -command $command

}

}

catch [Exception]

{

Write-Warning "Get-StatParameters"

Write-Warning $\_.Exception.Message

}

finally

{

$cmd.Dispose()

$scon.Dispose()

}

}

}

For the above, the **$read\_options** allows you to specify options in your update, if the application doesn't just warn about statistics (for instance, if it also makes a suggestion which can be saved - like **WITH SAMPLE 50 PERCENT**). If there is no option specified, this will simply be blank. We can also update the statistics using PowerShell:

Function Update-SpecificStatistics {

Param(

[ValidateLength(4,30)][string]$server

, [ValidateLength(1,45)][string]$database

, [ValidateLength(4,250)][string]$table

, [ValidateScript({Test-Path $\_})][string]$smolibrary

)

Process

{

$nl = [Environment]::NewLine

Add-Type -Path $smolibrary

$srv = New-Object Microsoft.SqlServer.Management.SMO.Server($server)

try

{

Write-Host "Updating statistics for $table ..."

$srv.Databases["$database"].Tables["$table"].UpdateStatistics("All","Fullscan")

}

catch [Exception]

{

Write-Warning "Update-SpecificStatistics"

Write-Warning $\_.Exception.Message

}

}

}

According to Microsoft, instead of "All" as the first parameter, you can enter *Column* or *Index*, depending on your target. If you don't to perform a full scan, you can look at other options [Microsoft offers](https://msdn.microsoft.com/library/microsoft.sqlserver.management.smo.statisticsscantype.aspx). We can also run T-SQL with the appropriate update statistics script, using the column **Options** on our table to specify how the command will be run. From there, we can get the command and then either Invoke-SqlCmd or Execute-Sql against the appropriate server and database.

Updating statistics heavily varies by environment, load as well as data. There are situations I've seen where frequently updating statistics multiple times an hour helps, while in other environments, doing so rarely helps. Also, be careful about overlooking small tables that may be key pieces of your application layer - a small table that grows with inaccurate statistics can sometimes be the cause of the headache.

**Next Steps**

* Determine how to retrieve the object information from the tool and then build a plan to update statistics based on your needs.
* Check out these tips:
  + [Execute UPDATE STATISTICS for all SQL Server Databases](https://www.mssqltips.com/sqlservertip/1606/execute-update-statistics-for-all-sql-server-databases/)
  + [Reduce Time for SQL Server Index Rebuilds and Update Statistics](https://www.mssqltips.com/sqlservertip/3100/reduce-time-for-sql-server-index-rebuilds-and-update-statistics/)
  + [SQL Server Auto Update and Auto Create Statistics Options](https://www.mssqltips.com/sqlservertip/2766/sql-server-auto-update-and-auto-create-statistics-options/)
  + Review all [SQL Server Maintenance](https://www.mssqltips.com/sql-server-tip-category/25/maintenance/) tips
  + Review all [SQL Server Indexing](https://www.mssqltips.com/sql-server-tip-category/38/indexing/) tips

Last Update: 3/9/2016