**Automate Removal of Saved SQL Server Objects During Deployments**

FROM: <https://www.mssqltips.com/sqlservertip/4259/automate-removal-of-saved-sql-server-objects-during-deployments/>

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

**Problem**

When we roll out our SQL Server deployments, we like to keep a backup copy of objects (tables, stored procedures, views, functions, etc.) that we change before running our change scripts. Unfortunately, we tend to forget about these objects until we experience low space issues and want to know if we can automate a solution with PowerShell to remove all of these older objects. In most of our deployments, we don't need to restore a full backup, so we prefer to save tables and procedures before deployment, if they don't exceed a certain amount of space (in some cases, a full restore is required).

**Solution**

One solution is to migrate backup objects by a schema designed for dropping objects after a certain period of time (such as after thirty days, or two weeks). The process flow is to create the backup object on the "remove" schema (whether through a copy or script and create) and run a script that later drops all the objects on that schema if their create date exceeds a specified date.

Create a schema with an unusual name to store the objects

Let's make sure that the name we choose for this schema is not something that another department will need later. For instance, a schema named "Remove" may be used by another team later on a different time table than we choose. We should choose a name that no one else will. For this example, I'll use the schema *dbarem*:

CREATE SCHEMA dbarem

Copy backup objects to the new schema

One of the fastest ways to do this for tables, which contain data, is:

SELECT \*

INTO dbarem.tblOurTable

FROM dbo.tblOurTable

If we are only updating a set of values for one or two columns in a large table, consider storing only the necessary values to avoid using too many resources. For an example:

SELECT

pkidcolumn, -- For the UPDATE with JOIN

, UpdateColumnOne -- Example names

, UpdateColumnTwo -- Example names

INTO dbarem.tblOurTable

FROM dbo.tblOurTable

In the above example, I can store a minimal amount of data for a backup (if the table was very large) and later perform and update using a join with the primary key column. This may or may not be a useful work-around if the environment is already limited on space.

In the case of stored procedures, views or functions, we can script them as CREATE and add the *dbarem* schema. An example:

CREATE PROCEDURE [dbo].[stp\_AddValues]

AS

BEGIN

INSERT INTO tblOurTable

VALUES (1,2,3,4)

END

Would change to:

CREATE PROCEDURE [dbarem].[stp\_AddValues]

AS

BEGIN

INSERT INTO tblOurTable

VALUES (1,2,3,4)

END

Schedule the drop

In the below function call, I drop the objects immediately - with 0 days as the parameter. If I wanted to wait two weeks, I would enter **-14** as the parameter for days. Even though I comment out the email line, this may be something to consider when testing. Note that the range of acceptable integer values for days is -31 to 0. If you want to keep objects for more than 31 days, you will want to make sure that the first value reflects this; such as -90 for a choice to retain objects for three months.

Function Drop-TablesBySchema {

Param(

[ValidateSet("2008R2","2012","2014","2016")][string]$version

,[ValidateLength(0,50)][string]$observer

, [ValidateLength(0,100)][string]$obdatabase

, [ValidateLength(0,25)][string]$obschema

, [ValidateRange(-31,0)][int]$days

)

Process

{

$obnl = [Environment]::NewLine

[string]$warning = ""

switch ($version)

{

"2008R2" {

Write-Host "Adding SMO library for version $version"

Add-Type -Path "C:\Program Files (x86)\Microsoft SQL Server\100\SDK\Assemblies\Microsoft.SqlServer.Smo.dll"

}

"2012" {

Write-Host "Adding SMO library for version $version"

Add-Type -Path "C:\Program Files (x86)\Microsoft SQL Server\110\SDK\Assemblies\Microsoft.SqlServer.Smo.dll"

}

"2014" {

Write-Host "Adding SMO library for version $version"

Add-Type -Path "C:\Program Files (x86)\Microsoft SQL Server\120\SDK\Assemblies\Microsoft.SqlServer.Smo.dll"

}

"2016" {

Write-Host "Adding SMO library for version $version"

Add-Type -Path "C:\Program Files (x86)\Microsoft SQL Server\130\SDK\Assemblies\Microsoft.SqlServer.Smo.dll"

}

}

$obsrv = New-Object Microsoft.SqlServer.Management.Smo.Server($observer)

### If Triggers and UserDefinedFunctions apply, add them to this array

$objectloop = @("Tables","StoredProcedures","Views")

foreach ($dropobject in $objectloop)

{

foreach ($dropit in $obsrv.Databases["$obdatabase"].$dropobject | Where-Object {($\_.Schema -eq $obschema) -and ($\_.CreateDate -lt (Get-Date).AddDays($days))})

{

$warning += "Dropped $dropobject [$observer].[$obdatabase].[$obschema].[" + $dropit.Name + "]" + $obnl

$dropit.Drop()

}

}

Write-Host $warning

### For multi-server environments, I like a confirmation email and the warning can be the body of the email.

#Send-MailMessage -From "" -To "" -SmtpServer "" -Subject "Cleared Off Old Objects" -Body $warning

}

}

Drop-TablesBySchema -version 2014 -tblserver "INSTANCE\SERVER" -tbldatabase "tblOurTable" -tblschema "dbarem" -days 0

From here, we can [use a configuration table where all the servers we manage are stored](https://www.mssqltips.com/sqlservertip/3544/using-powershell-with-configuration-tables-in-sql-server/), or set a job or task to automatically run the script daily.

**Next Steps**

* Verify the schema and its naming convention for migrating backup objects.
* If a deployment changes a few objects that would make it simpler to roll back by object instead of a full restore, update the save objects to use the schema for later removal.
* If extensive verification is required for deployments, you may be able to run the script at the end of the deployment and remove the objects.

Last Update: 5/30/2016