**Auto Rename SQL Server Tables and Their References with PowerShell**

|  |  |
| --- | --- |
|  | By: [Tim Smith](http://www.mssqltips.com/sqlserverauthor/92/tim-smith/)  FROM: <http://www.mssqltips.com/sqlservertip/3274/auto-rename-sql-server-tables-and-their-references-with-powershell/?utm_source=dailynewsletter&utm_medium=email&utm_content=text&utm_campaign=20150114> |

**Problem**

A developer recently asked me if we had an automated approach to changing table names when ad hoc requests arise, often from individuals who don't understand object dependencies. For instance, in this case, thirty-two tables needed to be re-named, yet hundreds of procedures and views referred to these tables, which meant that his changes had to be applied to more objects than the original tables.

**Solution**

Before we cover a solution, let's focus on preventing this problem. In this developer's example, the data end users hadn't seen the in-progress product. Some developers want to show the final result, which has advantages, but can come with disadvantages such as, "Hey, can we completely change these thirty two table names?" Receiving input while building projects will often minimize the amount of ad-hoc requirements; it is always better to over-communicate than to assume.

We can use PowerShell to automate this task of re-naming objects if the following conditions are true:

1. Our process to retrieve, update, remove and add data originate from the database (or other databases), and not from within an application.
2. We avoid using interfering object names; as an example, if we name a table "HumanResource" we don't also have columns, schemas, views, or other objects with that same name. If we use overlapping names, this script will warn us and not perform any action. In other words, how do you delineate specific objects, like tables, procedures and views - if you have no answer, don't use this script.
3. We know all the dependencies; in this example, we know that views and procedures refer to these tables. If we have other dependencies, we will need to expand the scope of this function.
4. If another database references these objects, it doesn't have a conflicting object name. This is something to keep in mind when creating objects - each object in a database doesn't match another object in another database by name and there are advantages to this approach (granted, some exceptions do apply).

We will use an example of two tables, referenced by both views and stored procedures. To shed light on the above points regarding naming convention, imagine if **tb\_Currency** was named to **Currency** - see the problems it would create by trying to automate a re-name? We receive requirements that **CurrencyRatingsInfo** should be renamed to **tb\_Ratings\_Currency** and **tb\_Currency** should be renamed to **tb\_Currencies\_All**.

CREATE TABLE CurrencyRatingsInfo(

CurrencyID INT,

Rating VARCHAR(4),

RatingDate DATE DEFAULT GETDATE()

)

CREATE TABLE tb\_Currency(

CurrencyID INT IDENTITY(1,1) PRIMARY KEY,

Currency VARCHAR(5)

)

CREATE PROCEDURE stp\_AddCurrData

AS

BEGIN

INSERT INTO CurrencyRatingsInfo (CurrencyID,Rating)

SELECT c.CurrencyID, st.Rating

FROM stage\_FreshCurrencyData st

INNER JOIN tb\_Currency c ON c.CurrencyID = st.CurrencyID

END

CREATE VIEW view\_AllCurrData

AS

SELECT c.Currency, r.Rating, r.RatingDate

FROM CurrencyRatingsInfo r

INNER JOIN tb\_Currency c ON c.CurrencyID = r.CurrencyID

We will start our PowerShell function by passing in four variables: the server (**$server**), the database (**$database**), the object we're searching for (**$search**), and the object name we want to rename $search to (**$new**). While in this example, only one procedure and one view (as well as the table name) are changed, this starts to make things easier when we have hundreds of both. After we pass in those parameters, we will use the SMO library to connect to the server. Note that we can add $smolibrary as a parameter if we have environments that may point to different dlls.

Function AlterObjectName ($server, $database, $search, $new)

{

## We pass in the SMO library dll to connect

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

## We pass in the server and instance to connect

$serv = New-Object Microsoft.SqlServer.Management.Smo.Server("$server")

## We point to the database we want to use on the server

$db = $serv.Databases["$database"]

}

Since we will change the name of our tables for this example, we want to find the table by the name that we pass in (**$search**). We can do this by looking through each table in the database with the name we passed to the function. If it's found, we then want to verify that there are no name conflicts. Therefore, we loop through every table in the database searching for this similarity that could create conflicts:

$table = $db.Tables["$search"]

$cnt = 0

foreach ($tab in $db.Tables)

{

$t = $tab.Name

if ($t -like "\*$search\*")

{

$cnt++

}

}

First, we want to verify that our table exists in our function. If it doesn't, we want PowerShell to warn us, "Hey, I didn't find that table." The below if statement checks if the table exists, by verifying that the object is not null:

if ($table.Name -eq $null)

{

Write-Host "Table not found."

}

Next, we want to change the table name in our else if statement because the table has been found, and we've verified that it doesn't have a conflict with another table, if it reaches this point:

elseif ($cnt -eq 1)

{

## We found the table, now let's re-name it

$table.Rename("$new")

}

At this point, our table has been re-named. The problem is that this will cause issues with both our view and procedure because they now point to an object that doesn't exist. We will now update the procedure and view. Note that the logic is the same, outside of looking at the view and procedure. First, we will loop through all the procedures that are not system procedures (**Where-Object {$\_.IsSystemObject -eq $false}**)), second we will replace every instance of **$search** we find with **$new** in the procedure's or view's text body, and finally alter the procedure or view to reflect this change.

## Return to use only user procedures and views

foreach ($proc in $db.StoredProcedures | Where-Object {$\_.IsSystemObject -eq $false})

{

## Replace search in the procedure's text body with new

$proc.TextBody = $proc.TextBody.Replace("$search", "$new")

## Now alter the stored procedure

$proc.Alter()

}

foreach ($view in $db.Views | Where-Object {$\_.IsSystemObject -eq $false})

{

$view.TextBody = $view.TextBody.Replace("$search", "$new")

$view.Alter()

}

Now our stored procedures and views will reflect the new table's name. Finally, we'll log an else statement at the end in case we have multiple objects come up. Consider if we were renaming a table called Currency, and we had other tables called CurrencyRating, CurrencySouth, CurrencyWest, etc. We don't want to update those names in our procedure, so if we've identified these conflicts, we want to log it so that we know ahead of time.

else

{

Write-Host "Warning: multiple tables with a similar name/naming convention."

}

Let's call this function, re-naming **CurrencyRatingsInfo** to **tb\_CurrencyRatings**.

Function AlterTableName ($server, $database, $search, $new)

{

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

$serv = New-Object Microsoft.SqlServer.Management.Smo.Server("$server")

$db = $serv.Databases["$database"]

$table = $db.Tables["$search"]

$cnt = 0

foreach ($tab in $db.Tables)

{

$t = $tab.Name

if ($t -like "\*$search\*")

{

$cnt++

}

}

if ($table.Name -eq $null)

{

Write-Host "Table not found."

}

elseif ($cnt -eq 1)

{

$table.Rename("$new")

foreach ($proc in $db.StoredProcedures | Where-Object {$\_.IsSystemObject -eq $false})

{

$proc.TextBody = $proc.TextBody.Replace("$search", "$new")

$proc.Alter()

}

foreach ($view in $db.Views | Where-Object {$\_.IsSystemObject -eq $false})

{

$view.TextBody = $view.TextBody.Replace("$search", "$new")

$view.Alter()

}

}

else

{

Write-Host "Warning: multiple tables with a similar name/naming convention."

}

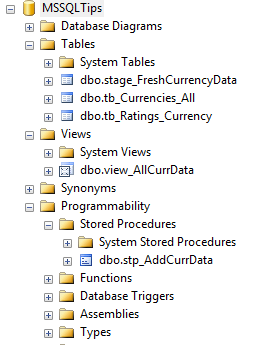
}

AlterTableName -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -search "CurrencyRatingsInfo" -new "tb\_Ratings\_Currency"

## These are called independently

AlterTableName -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -search "tb\_Currency" -new "tb\_Currencies\_All"

The below image shows the table re-named, and the below code shows both the updated script for the procedure and view (to view this, we can right click on each and select the option to either modify or alter:



-- Procedure

ALTER PROCEDURE [dbo].[stp\_AddCurrData]

AS

BEGIN

INSERT INTO tb\_Ratings\_Currency (CurrencyID,Rating)

SELECT c.CurrencyID, st.Rating

FROM stage\_FreshCurrencyData st

INNER JOIN tb\_Currencies\_All c ON c.CurrencyID = st.CurrencyID

END

-- View

ALTER VIEW [dbo].[view\_AllCurrData]

AS

SELECT c.Currency, r.Rating, r.RatingDate

FROM tb\_Ratings\_Currency r

INNER JOIN tb\_Currencies\_All c ON c.CurrencyID = r.CurrencyID

If we didn't know that only views and procedures called this table, our first step would be to identify dependencies. We also should always ensure that our objects when named (or re-named) are distinct to the degree that they will never interfere with other objects if changes must be made; keep in mind, this function won't update a table's name if it will interfere with another table. Note that in this example, we can re-name the objects to their original name after we made the change:

AlterTableName -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -search "tb\_Ratings\_Currency" -new "CurrencyRatingsInfo"

AlterTableName -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -search "tb\_Currencies\_All" -new "tb\_Currency"

If another database referred to these tables, we could also pass that database and server into the above function, if we know that above assumptions hold true for our environment. Because we want to be extra careful, I would suggest reviewing the changes made before making the changes; the below script will produce the changed text body of views and procedures, but will not update them. This is useful for debugging and verifying:

Function AlterTableName ($server, $database, $search, $new)

{

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

$serv = New-Object Microsoft.SqlServer.Management.Smo.Server("$server")

$db = $serv.Databases["$database"]

$table = $db.Tables["$search"]

$cnt = 0

foreach ($tab in $db.Tables)

{

$t = $tab.Name

if ($t -like "\*$search\*")

{

$cnt++

}

}

if ($table.Name -eq $null)

{

Write-Host "Table not found."

}

elseif ($cnt -eq 1)

{

$table.Rename("$new")

foreach ($proc in $db.StoredProcedures | Where-Object {$\_.IsSystemObject -eq $false})

{

$proc.TextBody = $proc.TextBody.Replace("$search", "$new")

$proc.TextBody

#$proc.Alter()

}

foreach ($view in $db.Views | Where-Object {$\_.IsSystemObject -eq $false})

{

$view.TextBody = $view.TextBody.Replace("$search", "$new")

$view.TextBody

#$view.Alter()

}

}

else

{

Write-Host "Warning: multiple tables with a similar name/naming convention."

}

}

We've commented out the re-name and alter so that we can verify what the text of the procedures and views will change to once we make these changes. We can use the SMO library to re-name other objects, stored procedures, views, databases, etc., but that doesn't mean we can re-name these objects to anything, nor does it mean that it will be re-named in every place it should be. For instance, with stored procedures, applications often call these and the above function doesn't replace the old names with new names for external objects. The below script (similar to the table script) will re-name a procedure, and we should be careful to consider the previous point:

Function AlterProcName ($server, $database, $search, $new)

{

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

$serv = New-Object Microsoft.SqlServer.Management.Smo.Server("$server")

$db = $serv.Databases["$database"]

$procedure = $db.StoredProcedures["$search"]

$cnt = 0

foreach ($proc in $db.StoredProcedures)

{

$p = $proc.Name

if ($p -like "\*$search\*")

{

$cnt++

}

}

if ($procedure.Name -eq $null)

{

Write-Host "Procedure not found."

}

elseif ($cnt -eq 1)

{

$procedure.Rename("$new")

}

else

{

Write-Host "Warning: multiple procedures with a similar name/naming convention."

}

}

AlterProcName -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -search "stp\_AddCurrData" -new "stp\_OurNewProcedure"

This re-naming script often will help in an initial phase, in a well designed environment, or with DBAs who know the ins and outs of their environment. In addition, developers should use caution with re-naming objects if these same objects are referenced outside the database, especially if other databases possess objects of the same name, even though these scripts are designed to warn developers if they detect a conflict.

**Next Steps**

* Test the above scripts, checking for possible conflicts and where these might arise.
* Remember, this won't update C#, Python or Ruby code. This updates SQL Server database values only.

Last Update: 7/18/2014

Bottom of Form

|  |  |
| --- | --- |
| **Saturday, July 19, 2014 - 2:11:11 AM - Sukanta Biswas** | [Read The Tip](http://www.mssqltips.com/sqlservertip/3274/auto-rename-sql-server-tables-and-their-references-with-powershell/) |
| I used SqlServer R2.  my question is that. i want to execute Trigger automatically during a particular date. How its possible?plz solver the problem Sir, | |

|  |  |
| --- | --- |
| **Friday, July 18, 2014 - 1:58:08 PM - jeff\_yao** | [Read The Tip](http://www.mssqltips.com/sqlservertip/3274/auto-rename-sql-server-tables-and-their-references-with-powershell/) |
| Good tip in general, but lacks a little bit of completeness.  For example, there is no consideration about other potential objects that may refer to the table, like UDFs, triggers, synonyms and jobs (whose step may be plain t-sql referring to the table).  So my suggestion here is you really need to rename a db object (not necessarily table, can be SP, View, function), every possible scenario needs to be reviewed. | |