**A General Way to Maintain History Tables**

**By** [**Jefferson Elias**](http://www.sqlservercentral.com/Authors/Articles/Jefferson_Elias/1637604/)**, 2017/06/30 (first published: 2015/09/28)**

**FROM:** [**http://www.sqlservercentral.com/articles/Maintenance/130502/?utm\_source=SSC&utm\_medium=pubemail**](http://www.sqlservercentral.com/articles/Maintenance/130502/?utm_source=SSC&utm_medium=pubemail)

**Introduction**

Have you ever had a monitoring tool taking a snapshot of a DMV, for instance, perfstats? Do you need to delete that data on regular basis? I will present you a way to perform this action with a set of tables ... dynamically.

The principle is pretty simple: insert a row in a given table and execute a single procedure periodically either manually, with a SQL Agent Job, or using a scheduled batch/Powershell script.  We can delete records of a table based on various criteria: a date column value, a "status" column, a condition based on multiple fields of different types, etc. Each deletion criterium will be implemented by a stored procedure. We will refer to those criteria as "cleanup types" and we will call "cleanup methods" the procedures which implements those cleanup types. Each cleanup type needs a set of parameters such as the object name, schema and database. We will refer to them as "cleanup parameters".

This article won't cover every possible cleanup types and methods but will instead introduce the fundamentals to help in their implementation. To illustrate what can be done, we will look closed at a cleanup type called "DATE\_LOWER\_THAN" that will delete all the records based on a date column which has a value older than a given number of days from the execution date.

**The Objects**

As deleting records from history tables can be assimilated to a maintenance operation, I placed the all the objects described in this article inside a database schema I called [maintenance].

Let's now talk about the objects inside that schema starting with the table in which we will store the cleanup parameters. This table is called [MaintenanceSettings]. It has multiple columns which are the cleanup parameters explained above. Those parameters can be either common or specialized.

Here are examples of common parameters:

* the name of the database and schema in which we can find the table to maintain,
* the name of this table itself,
* the primary discriminating column which has to be used for deletion. (name and data type)

We also have specialized parameters which are dynamic but for which there is a writing convention. Indeed, the value specified in aspecialized argument has to look like this: '<DATA\_TYPE>:<VALUE>'. For instance, a specified argument example could be "NUMBER:90" or 'STRING:CLOSED'.

You will find below the code which creates the [MaintenanceSettings] table. In this code, the CleanupParamX columns are specialized parameters while the [CleanupEnabled] column is a flag which, when set to true, enables the cleanup method execution against the table.

CREATE TABLE maintenance.MaintenanceSettings (

DbName varchar(256) NOT NULL,

ObjectOwner varchar(256) NOT NULL,

ObjectName varchar(256) NOT NULL,

ObjectType varchar(256) NOT NULL,

CleanupEnabled BIT NOT NULL,

CleanupColumn varchar(256) NOT NULL,

CleanupType varchar(256) NOT NULL, -- ONLY ''DATE\_LOWER\_THAN'' at the moment is allowed

-- FORMAT: "<TYPE>:<VALUE>". Example : STRING:blabla; NUMBER:

CleanupParam1 varchar(256) ,

CleanupParam2 varchar(256) ,

CleanupParam3 varchar(256) ,

CleanupParam4 varchar(256)

)

With the "DATE\_LOWER\_THAN" method, we will set for each table the specialized parameters as follows:

* CleanupParam1 will keep the date format that has to be used. At the moment only the value "NUMBER:112" has been tested.
* CleanupParam2 will keep the number of DAYS we need to keep in the table. Here is a possible value: "NUMBER:90".
* CleanupParam3 will keep a number of records that have to be performed at a time. That is the number of records that need to be included in a single DELETE operation. Here is a possible value: "NUMBER:30000".
* CleanupParam4 is not used at the moment.

As the [MaintenanceSettings] table could be used for other settings than table maintenance by defining new columns, there is a view called [CleanupSettings] with has exactly the same columns as the table above.

Our cleanup parameters are stored in the database for any cleanup type. We can have a look at the cleanup methods and how they are called. There is a procedure called [TableCleanupUsingParamTbl] that will run the other procedures with the appropriate parameters. It's algorithm is simply as follows:

foreach Table in CleanupSettings {

if Cleanup is disabled for that Table using that column {

CONTINUE

}

"Parse cleanup parameters"

"Determine cleanup type and Run appropriate cleanup method"

}

The action "Parse cleanup parameters" uses a helper procedure called [ParseMaintenanceCleanupParam], which breaks the CleanupParamX column value into its type and value.

As explained previously, we only have the "DATE\_LOWER\_THAN" cleanup type implemented at the moment. Its cleanup method is materialized by the stored procedure called [TableCleanupByDate], which will scan the given table looking for records with a CleanupColumn value older than the retention period. This is obtained by substringing the <VALUE> side of CleanupParam2 value for the given table to the return of GETDATE built-in function. If there are too many records to delete at a time based on CleanupParam3 value, it will use a "divide and conquer" approach with dichotomic recursion as shown in the algorithm below.

DECLARE @cnt BIGINT

DECLARE @minVal DATE

DECLARE @newRetention BIGINT

SELECT @cnt = count(\*), min(DateColumn) from Table where DateColumn < ActualRetention

IF(@cnt > CleanupParam3) {

SET @newRetention = ActualRetention + (ActualRetention - minVal) / 2

"Call TableCleanupByDate with retention = @newRetention"

}

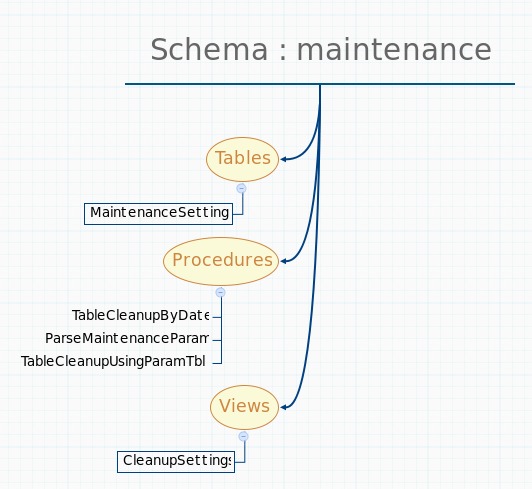
DELETE Table WHERE DateColumn < ActualRetention

At this point we now have reviewed all the objects that are used in this solution.

**Getting started**

I attached a script with this article. Download it, and open it up. You may want to change the name of the target database. To do so, just find and replace all occurences of "DBAToolbox". Next, run the script against your test server.

This script will create a bunch of database objects in the database of your choice. Those objects are explained in previous section. Their names and types are shown in the figure below.



For those who have jumped directly to this section, here is a little explanation about the objects. For a deeper explanation, refer to the previous sections.

* **MaintenanceSettings** table contains miscellaneous parameters for object maintenance (especially the object - DbName, SchemaName, ObjectName - to take care of and some parameters for each part of maintenance).
* **TableCleanupByDate** is the generic procedure that deletes rows in a given table based on the parameters for a the "DATE\_LOWER\_THAN" cleanup type.
* **ParseMaintenanceParam** is just an utility procedure which parses the CleanupParamXX fields and returns the <TYPE> and <VALUE> into different variables.
* **TableCleanupUsingParamTbl** is the central function which takes the parameters set in MaintenanceSettings table and performs the cleanup.
* **CleanupSettings** is a view on MaintenanceSettings table as this table could be used to define settings for other maintenance functions as Index Maintenance or whatever.

**Example Usage**

Let's create a test table, populate it and run the procedure [maintenance].[TableCleanupUsingParamTbl].

PRINT 'Create a test table'

create table dbo.TestDateBasedTable (

DateStamp DATETIME DEFAULT GETDATE() ,

DataD INT

);

GO

PRINT 'Populate the a test table'

DECLARE @i int;

SET @i = 0;

BEGIN TRANSACTION

while @i < 10000

BEGIN

insert into dbo.TestDateBasedTable (

DateStamp,DataD

)

values (

GETDATE() - @i/100 , @i

);

SET @i = @i + 1 ;

END ;

COMMIT;

GO

Let's see what are the minimum and maximum date values stored in our test table.

DECLARE @minDS VARCHAR(128);

DECLARE @maxDS VARCHAR(128);

select @minDS = CONVERT(VARCHAR,min(DateStamp),112), @maxDS = CONVERT(VARCHAR,max(DateStamp),112) from dbo.TestDateBasedTable

PRINT 'Min Date : ' + @minDS

PRINT 'Max Date : ' + @maxDS

--Min Date : 20150521

--Max Date : 20150819

GO

We see that our minimum date is 2015 05 21.

Let's now define the data maintenance on our test table. That done, the [TableCleanupUsingParamTbl] procedure will take it into consideration

insert into [maintenance].[MaintenanceSettings] (

[DbName],[ObjectOwner],[ObjectName],[ObjectType],[CleanupEnabled],[CleanupColumn],[CleanupType],[CleanupParam1],[CleanupParam2],[CleanupParam3],[CleanupParam4]

)

values (

'DBAToolbox','dbo','TestDateBasedTable','TABLE',1,'DateStamp','DATE\_LOWER\_THAN','NUMBER:112','NUMBER:10','NUMBER:30000',null

)

Now let's run the procedure, [TableCleanupUsingParamTbl].

exec [maintenance].[TableCleanupUsingParamTbl] @debug = 1

Let's see if the procedure execution did something to our test table.

-- Just to see the effects

DECLARE @minDS VARCHAR(128);

DECLARE @maxDS VARCHAR(128);

select @minDS = CONVERT(VARCHAR,min(DateStamp),112), @maxDS = CONVERT(VARCHAR,max(DateStamp),112) from dbo.TestDateBasedTable

PRINT 'Min Date : ' + @minDS

PRINT 'Max Date : ' + @maxDS

GO

-- Min Date : 20150809

-- Max Date : 20150819

We had previously a minimum date value of "2015 05 21" and we now have a minimum date value of "2015-08-09". We set the CleanupParam2 value to "NUMBER:10" and it really kept 10 days as the maximum date value is "2015 08 19".

**What's next?**

As you now know how to run the [TableCleanupUsingParamTbl] procedure and set its parameters inside the MaintenanceSettings table, you could automate its execution by creating a SQL Agent Job or a Scheduled Powershell script that will call this procedure and clean things up.

You can also implement your own data cleanup method and call it from the TableCleanupUsingParamTbl procedure. For instance, let's say you have a table with a status column. Assuming that a 'T' value for this column states that the information contained in the row has been transmitted and therefore can be deleted. You can implement a procedure to perform this action, call the CleanupType "STATUSCOL\_EQUALS" and modify the TableCleanupUsingParamTbl stored procedure so that it calls the cleanup method associated to the "STATUSCOL\_EQUALS" cleanup type.

**Future Improvements**

Here are a couple of improvement in the list.

* Creating a method for setting the cleanup inside MaintenanceSettings table as it's not really natural to set the <TYPE>:<VALUE>...
* Implement a DATE\_LOWER\_THAN based on different retention unit, not only days.
* Add new cleanup methods

**Thanks for reading**

I hope this will help some of you in your daily tasks.

I'd appreciate to be notified whenever you experience an issue using this method and script or if you have any comment on this subject.

**Resources:**

[Solution.Maintenance.DataMgmt.sql](http://www.sqlservercentral.com/Files/Solution.Maintenance.DataMgmt.sql/26967.sql)