Temporal Tables Part 2 - Creating and Deleting System-Versioned Tables

In this article I will show you how to create and deleted system-versioned tables and the associated history temporal tables.

Business Situation

Our Management is keen on using the new temporal table support feature introduced in SQL Server 2016 which tracks the history of all the data changes to a table as part of their new RDC Limit Tier functionality we are building. One of the first tables they want to create that will track historical data is a table named *dbo.RDCLimitTier.* This table will contain the RDC Limit Tier of the current users. By using the historical temporal data support in SQL Server 2016, management will be able to track RDC Limit Tier changes for the users over time. By creating this system-versioned table using the new temporal data table the application will be able to track users RDC Limit Tier changes over time.

RDC Limit Tiers

Tier ID	Tier Name	Limit Per Day ($)
0	Tier 0	0
1	Tier 1	1000
2	Tier 2	2500
3	Tier 3	5000
4	Tier 4	10000

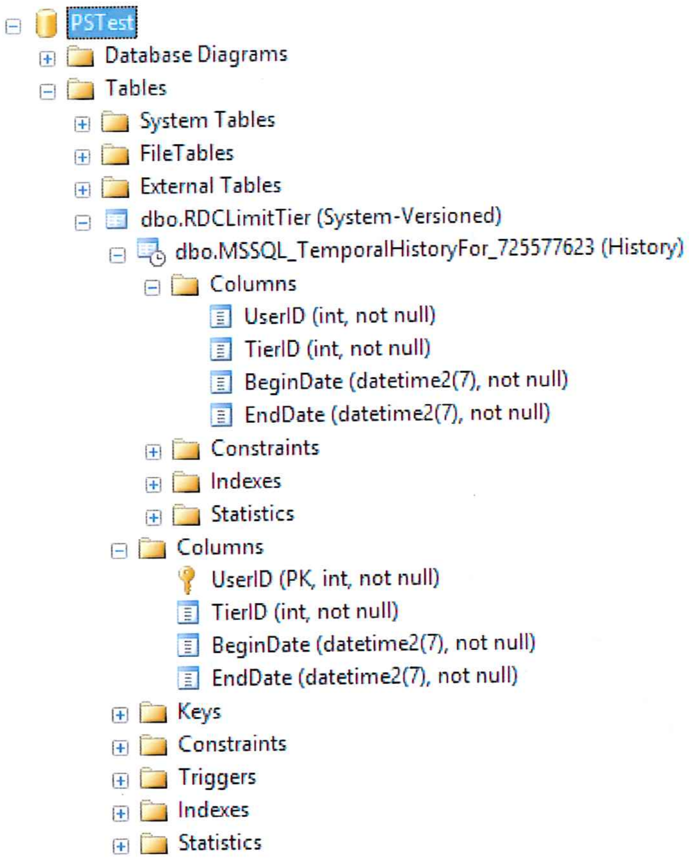
Creating a System-Generated History Table

A system-generated history table is a table that is automatically named by SQL Server when it is created. The following CREATE TABLE script will create a history table with a system generated name:

```
CREATE TABLE [dbo].[RDCLimitTier](
      [UserID] [int] NOT NULL PRIMARY KEY CLUSTERED,
      [TierID] [int] NOT NULL,
         [BeginDate] datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
      [EndDate] datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
      PERIOD FOR SYSTEM_TIME ([BeginDate], [EndDate]))
WITH (SYSTEM_VERSIONING = ON);
```

Here we created a table named *RDCLimitTier*. Note that we didn't specify the name of the history table. All we specified was the "WITH (SYSTEM_VERSIONING = ON)" specification.

If you look at the Object Explorer details in SQL Server Management Studio (SSMS), after creating this table we can see the system generated temporal data history table. Here is an image of the Object Explorer output:



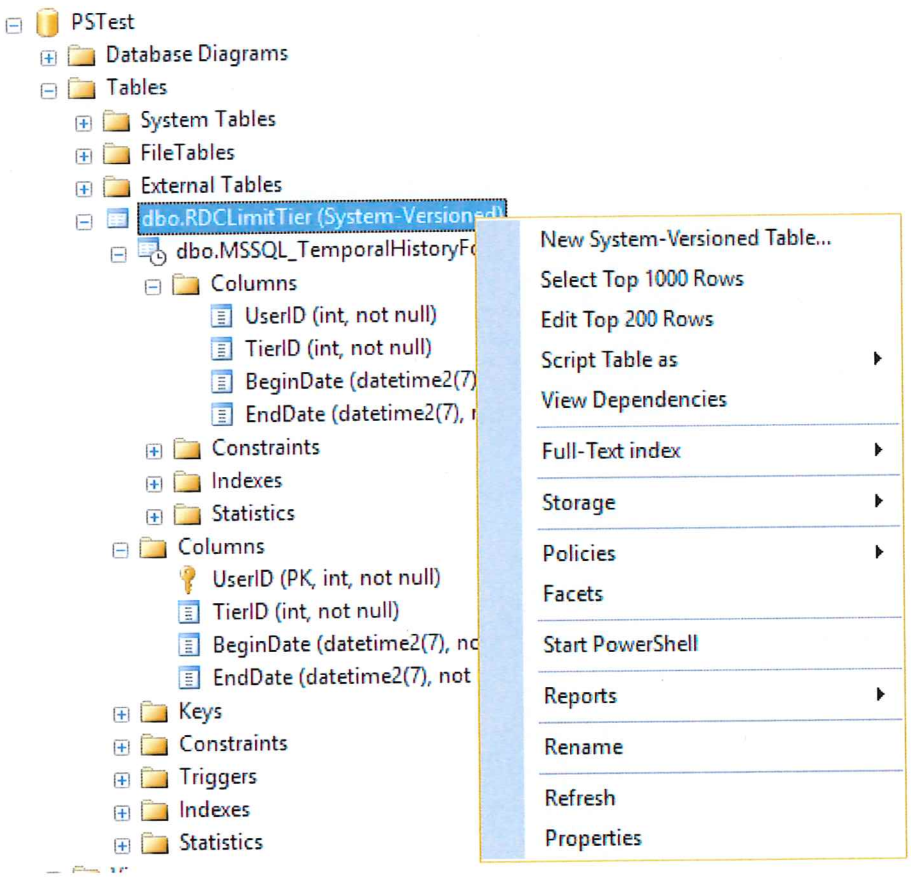
```
PSTest
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    dbo.RDCLimitTier (System-Versioned)
      dbo.MSSQL_TemporalHistoryFor_725577623 (History)
        Columns
          UserID (int, not null)
          TierID (int, not null)
          BeginDate (datetime2(7), not null)
          EndDate (datetime2(7), not null)
        Constraints
        Indexes
        Statistics
      Columns
        UserID (PK, int, not null)
        TierID (int, not null)
        BeginDate (datetime2(7), not null)
        EndDate (datetime2(7), not null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
```

As you can see the table *dbo.RDCLimitTier* is identified as a "System-Versioned" table. This tells us that there is a history table associated with this table. The second line of output in the above screenshot shows the actual history table that was created. For this example that table name is *dbo.MSSQL_TemporalHistoryFor_725577623*.

Dropping a System-Versioned Table and the Associated Historical Table

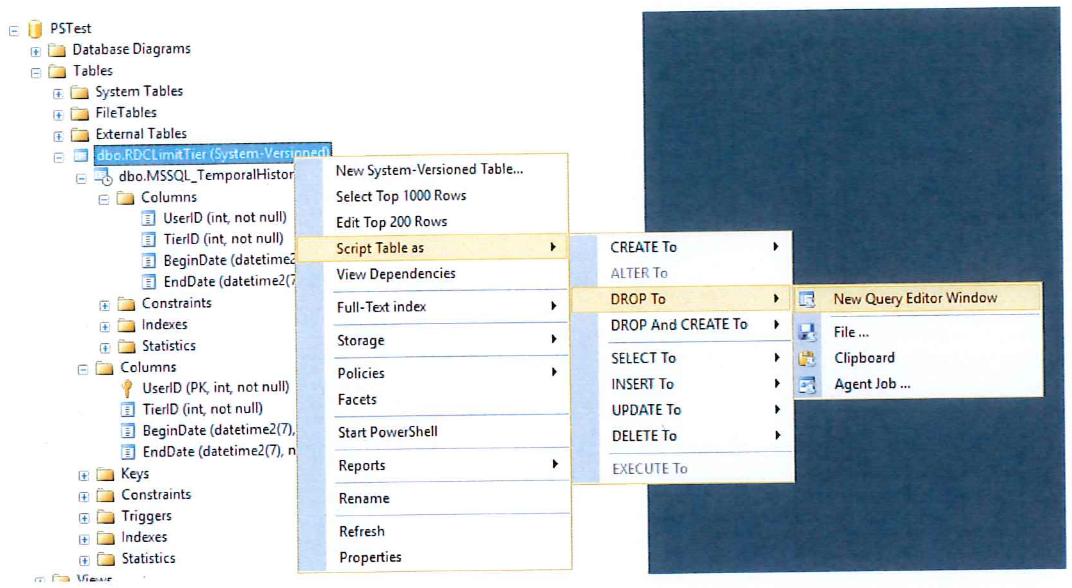
I personally don't like system generated names. As you can see in the prior example above, the historical temporal data table created by the system is not the easiest to remember. Therefore instead of using the system-generated table name as in the above example, I will drop the system generated history table and the associated temporal table.

You might be thinking that you can delete a system-generated history table and its associated table by right-clicking on the system-versioned table name, in Object Explorer and then just right clicking the "Delete" menu item. If you think that you'd be wrong. Here is what you see when you do a right-click on the *dbo.RDCLimitTier* table:



As you can see from the menu above there is no delete option displayed when you clicked on the *dbo.RDCLimitTier* table. So how do you delete a system-versioned table?

It is not a single statement process to delete a system-versioned table. To generate the statements necessary to delete the *dbo.RDCLimitTier* table, right click on the table; select the "DROP to" scripting option to the new query editor window as shown below.



After selecting the “DROP to” option the following script was placed in the new query editor window:

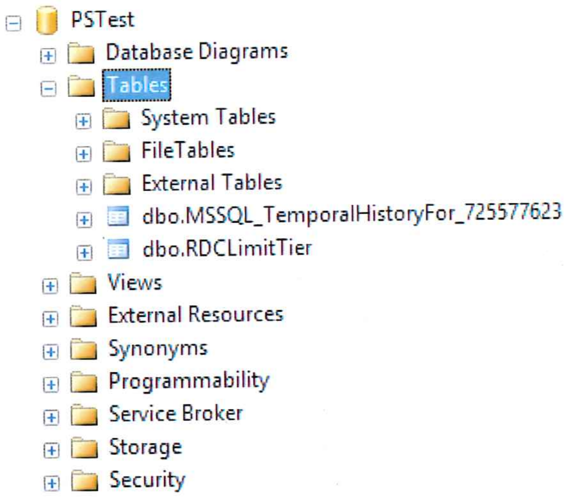
```
USE [PSTest]
GO

/****** Object:  Table [dbo].[RDCLimitTier]    Script Date: 8/10/2016 9:07:45 PM
******/
ALTER TABLE [dbo].[RDCLimitTier] SET ( SYSTEM_VERSIONING = OFF )
GO

/****** Object:  Table [dbo].[RDCLimitTier]    Script Date: 8/10/2016 9:07:45 PM
******/
DROP TABLE [dbo].[RDCLimitTier]
GO

/****** Object:  Table [dbo].[MSSQL_TemporalHistoryFor_725577623]    Script Date:
8/10/2016 9:07:46 PM ******/
DROP TABLE [dbo].[MSSQL_TemporalHistoryFor_725577623]
GO
```

Here you can see that first we have to ALTER the table *dbo.RDCLimitTier* to set the system versioning to off. As soon as we perform that ALTER statement, the *dbo.RDCLimitTier* table is no longer versioned, but the history table *dbo.MSSQL_TemporalHistoryFor_725577623* becomes real table. This can be seen in the following SQL Server Object Explorer window:



To finish up dropping my original *dbo.RDCLimitTier* table and the system generated history table I run the following commands from a SQL Server Query window:

```
USE [PSTest]
GO

/****** Object:  Table [dbo].[RDCLimitTier]    Script Date: 8/10/2016 9:07:45 PM
******/
DROP TABLE [dbo].[RDCLimitTier]
GO

/****** Object:  Table [dbo].[MSSQL_TemporalHistoryFor_725577623]    Script Date:
8/10/2016 9:07:46 PM ******/
DROP TABLE [dbo].[MSSQL_TemporalHistoryFor_725577623]
GO
```

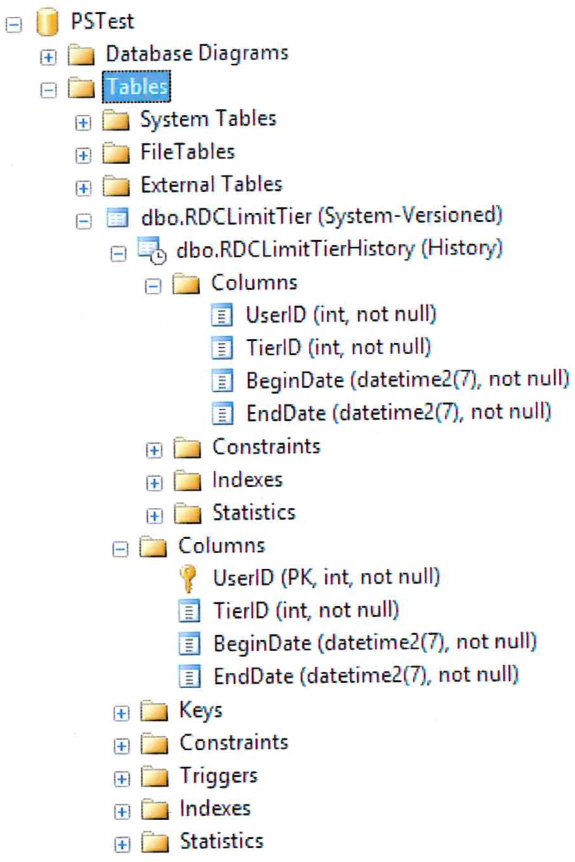
Note: - You can't drop the system generated history table alone when the system versioning is turned on for the *dbo.RDCLimitTier* table.

Creating a Named Historical Temporal table

Like I said earlier I don't like default names for SQL Server objects. Therefore I want to show you how to create a temporal table that has a name that we will define. There are two different ways to create a temporal table that is named. The first method I will show you is to let the SQL Server generate the column definitions based on the base table. Below is the code to create a temporal table that we have named, but lets the system generate the column definitions for the historical temporal table from the definition of the base table:

```
CREATE TABLE [dbo].[RDCLimitTier](
      [UserID] [int] NOT NULL PRIMARY KEY CLUSTERED,
      [TierID] [int] NOT NULL,
         [BeginDate] datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
      [EndDate] datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
      PERIOD FOR SYSTEM_TIME ([BeginDate], [EndDate]))
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.RDCLimitTierHistory));
```

Here you can see that we have created the temporal history table with a name by using the HISTORY_TABLE clause on the CREATE TABLE statement. In this example I didn't specify any column definitions for the history table. The columns defined in the history table are the same as the dbo.*RDCLimitTier* table.



```
PSTest
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    dbo.RDCLimitTier (System-Versioned)
      dbo.RDCLimitTierHistory (History)
        Columns
          UserID (int, not null)
          TierID (int, not null)
          BeginDate (datetime2(7), not null)
          EndDate (datetime2(7), not null)
        Constraints
        Indexes
        Statistics
      Columns
        UserID (PK, int, not null)
        TierID (int, not null)
        BeginDate (datetime2(7), not null)
        EndDate (datetime2(7), not null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
```

Summary

Creating a temporal data history table while you create your table is easy by just adding the "SYTEM_VERSIONING' clause to your create table statement. Keep in mind you can either let SQL Server generate the history table name, or you can specify a history table name with your "CREATE TABLE" statement. Next time you have a business need to track table records changing over time, don't write application code to do this, but instead build a history table, using the temporal data table feature of SQL Server 2016.