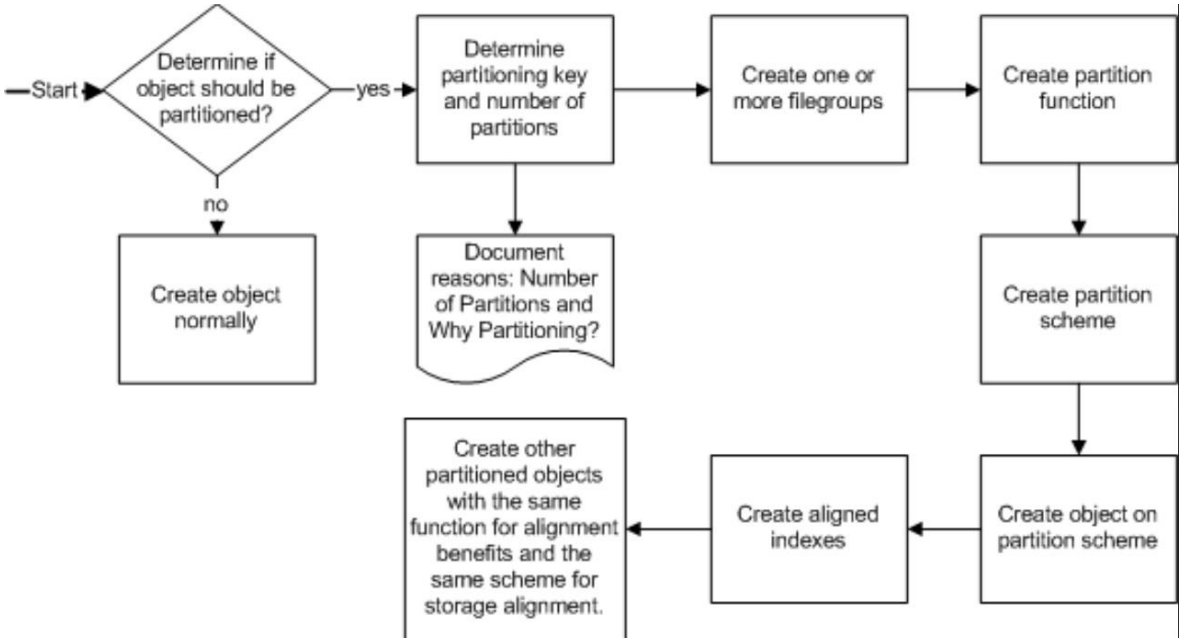
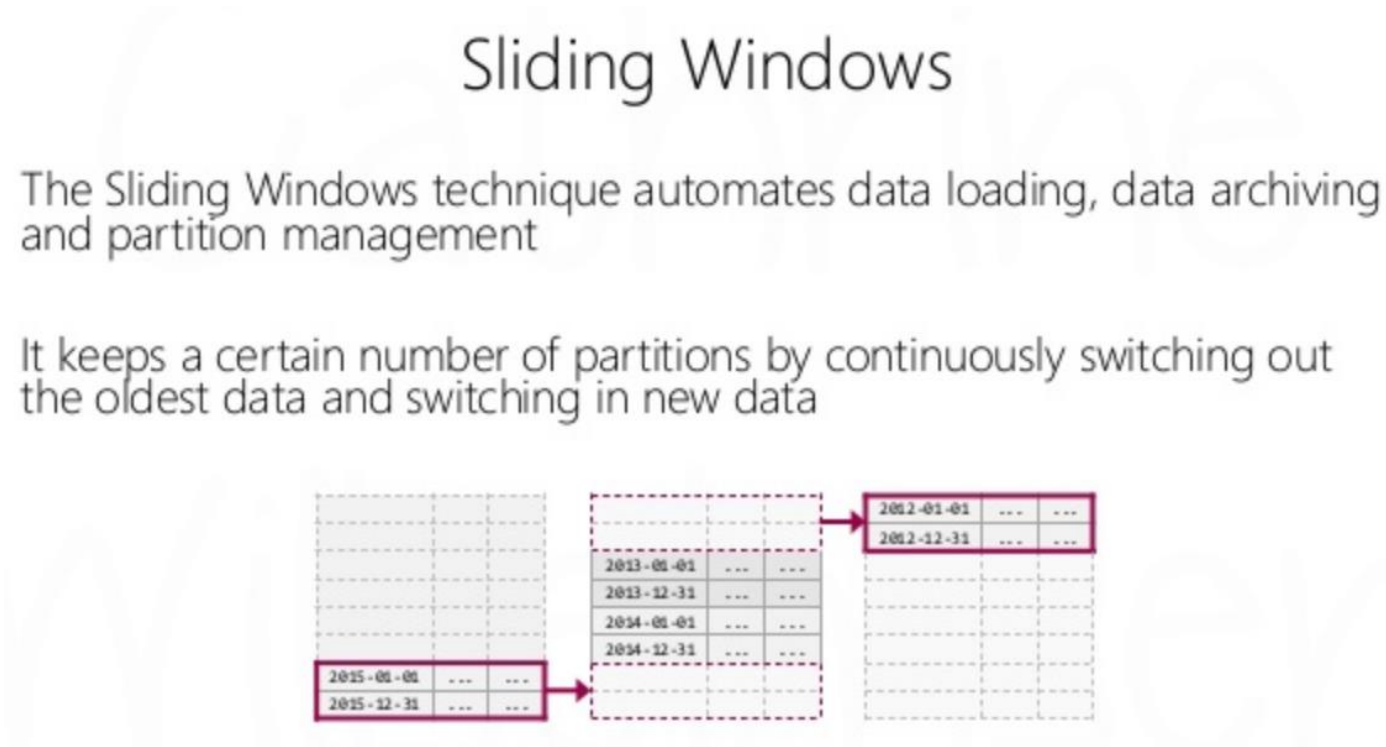
Table Partitioning & Data Warehouses

Discussion



Space



Space

Sliding Window Scenario

The concept of a sliding window scenario is to manage and keep the same number of partitions on a partitioned table over time. When a new period starts, a new partition is created to accommodate the new data and at the same time the oldest partition is taken out from the partitioned table to maintain the same number of partitions. The oldest partition that has been taken out of the partitioned table either can be dropped or archived. The best part of managing a sliding window scenario in SQL Server is its meta data operation, and hence it is significantly faster.

These are the steps you need to follow for setting up a sliding window scenario:

- Create a table, which will act like a staging table and which will have same structure and will reside on the same file group as the oldest partition. Next you need to create matching clustered indexes and optionally non clustered indexes.
- Take out the oldest partition from the partitioned table into the staging table, as created above with use of the SWITCH clause with the ALTER TABLE statement.
- Modify the partition function to remove the boundary value of the oldest partition with the MERGE clause with ALTER PARTITION FUNCTION statement.
- Modify the partition scheme to designate a new file group to be used by the newest partition with the NEXT USED clause of the ALTER PARTITION SCHEME statement.
- Modify the partition function to add the boundary value for the newest partition with the SPLIT clause with the ALTER PARTITION FUNCTION statement.

Related Articles

- The Format() Function in SQL Server 2012
- Sequence Object in SQL Server 2012
- Contained Database Authentication in SQL Server 2012
- Analysis Services PowerShell Provider (SQLAS) in SQL Server 2012

The data in the staging table, which now contains the data from the oldest partition of the partitioned table, can be either deleted/dropped or can be archived to another table or another partitioned archived table.

And now, there are two ways to load data into newest partition. The first one, when you execute the INSERT statement, it moves the record to the appropriate partition and the second one, is to load bulk data using the following steps:

- Create a table, which will act like a staging table and which will have the same structure and will reside on the same file group as the newest partition.
- Load data into this staging table; please note, as this staging table is not associated to your partitioned table the bulk data load operation into this staging table will not have an impact on the partitioned table.

- Next you need to create matching clustered indexes and optionally non clustered indexes and ensure these indexes are aligned.
- Take the staging table into the newest partition of the partitioned table with use of the SWITCH clause with the ALTER TABLE statement.
- Update the index statistics to ensure indexes are utilized appropriately for the queries.

Setting up the Sliding Window Scenario

Scenario 1 – 2016/2014 https://msdn.microsoft.com/en-us/library/cc280599(v=sql.120).aspx

Manage Partition Wizard

Use the Manage Partition Wizard to manage and modify existing partitioned tables through partition switching or the implementation of a sliding window scenario. This wizard can ease the management of your partitions and simplify the regular migration of data in and out of your tables.

To start the Manage Partition Wizard

- In SQL Server Management Studio, select the database, right-click the table on which you want to create partitions, point to **Storage**, and then click **Manage Partition**.

  Note If **Manage Partition** is unavailable, you may have selected a table that does not contain partitions. Click **Create Partition** on the **Storage** submenu and use the **Create Partition Wizard** to create partitions in your table.

For general information about partitions and indexes, see Partitioned Tables and Indexes.

This section provides the information that is required to manage, modify, and implement partitions using the **Manage Partition Wizard**.

The following sections provide help on the pages of the **Manage Partition Wizard**.

Select Partition Action Page

Select Partition Switching-In Options Page

Select Partition Switching-Out Options Page

Select the Staging Table Options Page

Select Output Option Page

New Job Schedule Page

Summary Page

Progress Page

Use the **Select Partition Action** page to choose the action you want to perform on your partition.

Create a Staging Table

Partition switching is a common partitioning task if you have a partitioned table that you migrate data into and out of on a regular basis; for example, you have a partitioned table that stores current quarterly data, and you must move in new data and archive older data at the end of each quarter.

The wizard designs the staging table with the same partitioning column, table and column structure, and indexes, and stores the new table in the filegroup where your source partition is located.

To create a staging table to switch in or switch out partition data, select **Create a staging table for partition switching**.

Sliding Window Scenario

To manage your partitions in a sliding-window scenario, select **Manage partitioned data in a sliding window scenario**.

UIElement List

Create a staging table for partition switching

Creates a staging table for the data you are switching in or switching out of the existing partitioned table.

Switch out partition

Provides options when removing a partition from the table.

Switch in partition

Provides options when adding a partition to the table.

Manage partitioned data in a sliding window scenario

Appends an empty partition to the existing table that can be used for switching in data. The wizard currently supports switching into the last partition and switching out the first partition.

Select Partition Switching-In Options Page

Use the **Select Partition Switching-In options** page to select the staging table you are switching into the partitioned table.

UIElement List

Show All Partitions

Select to show all partitions, including the partitions currently in the partitioned table.

Partition grid

Displays the partition name, **Left boundary**, **Right boundary**, **Filegroup**, and **Row count** of the partitions you selected.

Switch in table

Select the staging table that contains the partition that you want to add to your partitioned table. You must create this staging table before you switch-in partitions with the **Manage PartitionsWizard**.

Select Partition Switching-Out Options Page

Use the **Select Partition Switching-Out options** page to select the partition and the staging table to hold the partitioned data that you are switching out of the partitioned table.

UIElement List

Partition grid

Displays the partition name, **Left boundary**, **Right boundary**, **Filegroup**, and **Row count** of the partitions you selected.

Switch out table

Choose a new table or an existing table to switch-out your data to.

New

Enter a new name for the staging table you want to use for the partition to switch out of the current source table.

Existing

Select an existing staging table you want to use for the partition you want to switch out of the current source table. If the existing table contains data, this data will be overwritten with the data you are switching out.

Select the Staging Table Options Page

Use the **Select the Staging Table Options** page to create the staging table you want to use for switching your partitioned data.

Staging tables must reside in the same filegroup of the selected partition where the source table is located. The staging table must mirror the design of both the source table and the destination table.

You can also create the same indexes in the staging table that exist in the source partition. The staging table automatically contains a constraint based on the elements of the source partition. This constraint is typically generated from the boundary value of the source partition.

UIElement List

Staging table name

Create a name for the staging table or accept the default name displayed in the edit box.

Switch partition

Select the source partition that you want to switch out of the current table.

New boundary value

Select or enter the boundary value you want for the partition in the staging table.

Filegroup

Select a filegroup for the new table.

In This Section

Select Output Option Page

Use the **Select Output Option** page to specify how you want to complete the modifications to your partitions.

When the wizard finishes, it creates a script in Query Editor to modify partitions in the table. Select **Create Script** if you want to review the script, and then execute the script manually.

Script to file

Generate the script to a .sql file. Specify either **Unicode** or **ANSI text**. To specify a name and location for the file, click **Browse**.

Script to Clipboard

Save the script to the Clipboard.

Script to New Query Window

Generate the script to a Query Editor window. If no editor window is open, a new editor window opens as the target for the script.

Run immediately

Have the wizard finish modifications to the partitions when you click **Next** or **Finish**.

Select to modify the table partitions at a scheduled date and time.

Change schedule

Opens the **New Job Schedule** dialog box, where you can select, change, or view the properties of the scheduled job.

New Job Schedule Page

Use the **New Job Schedule** page to view and change the properties of the schedule.

Select the type of schedule you want for the SQL Server Agent job.

Name

Type a new name for the schedule.

Jobs in schedule

View the existing jobs that use the schedule.

Schedule type

Select the type of schedule.

Enabled

Enable or disable the schedule.

Recurring Schedule Types Options

Select the frequency of the scheduled job.

Occurs

Select the interval at which the schedule recurs.

Recurs every

Select the number of days or weeks between recurrences of the schedule. This option is not available for monthly schedules.

Monday

Set the job to occur on a Monday. Only available for weekly schedules.

Tuesday

Set the job to occur on a Tuesday. Only available for weekly schedules.

Wednesday

Set the job to occur on a Wednesday. Only available for weekly schedules.

Thursday

Set the job to occur on a Thursday. Only available for weekly schedules.

Friday

Set the job to occur on a Friday. Only available for weekly schedules.

Saturday

Set the job to occur on a Saturday. Only available for weekly schedules.

Sunday

Set the job to occur on a Sunday. Only available for weekly schedules.

Day

Select the day of the month the schedule occurs. Only available for monthly schedules.

Of every

Select the number of months between occurrences of the schedule. Only available for monthly schedules.

The

Specify a schedule for a specific day of the week on a specific week within the month. Only available for monthly schedules.

Occurs once at

Set the time for a job to occur daily.

Occurs every

Set the number of hours or minutes between occurrences.

Start date

Set the date when this schedule will become effective.

End date

Set the date when the schedule will no longer be effective.

No end date

Specify that the schedule will remain effective indefinitely.

One Time Schedule Types Options

If you schedule a job to run once, you must select a date and time in the future.

Date

Select the date for the job to run.

Time

Select the time for the job to run.

Summary Page

Use the **Summary** page to review the options that you have selected on the previous pages.

UIElement List

Review your selections

Displays the selections you have made for each page of the wizard. Click a node to expand and view your previously selected options.

Progress Page

Use the **Progress** page to monitor status information about the actions of the **Manage Partition Wizard**. Depending on the options that you selected in the wizard, the **Progress** page might contain one or more actions. The top box displays the overall status of the wizard and the number of status, error, and warning messages that the wizard has received.

Details

Provides the action, status, and any messages that are returned from action taken by the wizard.

Action

Specifies the type and name of each action.

Status

Indicates whether the wizard action as a whole returned the value of **Success** or **Failure**.

Message

Provides any error or warning messages that are returned from the process.

Stop

Stop the action of the wizard.

Report

Create a report that contains the results of the **Manage Partition Wizard**. The options are:

- View Report
- Save Report to File
- Copy Report to Clipboard
- Send Report as Email

View Report

Open the **View Report** dialog box. This dialog box contains a text report of the progress of the **Manage Partition Wizard**.

Close the wizard.

Scenario 2 – not tested on SQL 2014/2016

Microsoft Corporation March 2006 - Applies To: SQL Server 2005

Summary: This article shows how to implement an automatic sliding window in a partitioned table on Microsoft SQL Server 2005. (8 printed pages)

Horizontal partitioning is a new feature of SQL Server 2005. When trying to use horizontal partitioning, a common problem is how to make it fully automated without the need of manual intervention.

The sliding sample in the AdventureWorks sample database is hard-coded. The objective of this document is to show how to implement it in a real world environment. The sample has a 60-day sliding window with 60 partitions.

The idea is to create automated stored procedures with no parameters needed so that they can be scheduled to run through SQL Agent on a daily basis. Two stored procedures are needed: one for the right side and

another for the left side. The first one will create a new partition for the next day. The second one will kill the most left partition.

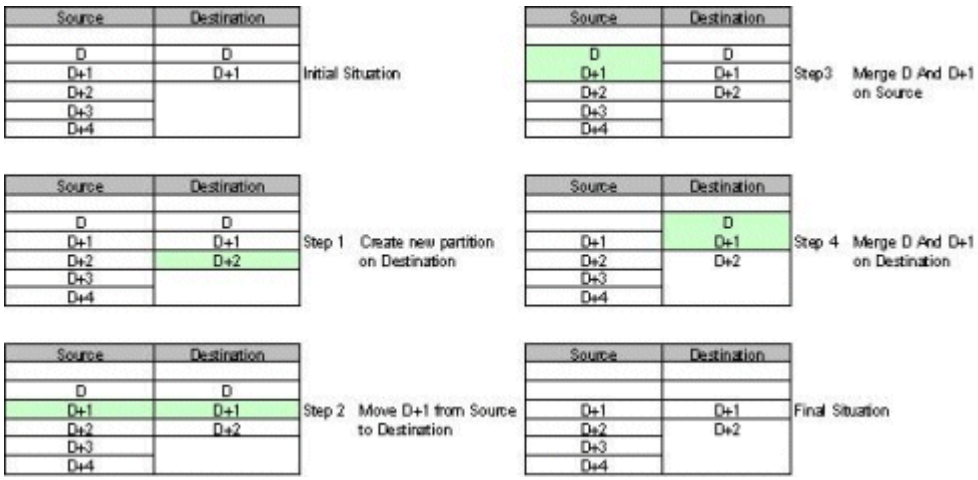
The first problem encountered in transforming the script is the ADD CONSTRAINT statement used to establish a left boundary on the source table. In order to automate the script, it is necessary to have all dates based on the partition 1 date. In earlier versions of SQL Server, this was accomplished by using ALTER TABLE statements; however, this will not work because ALTER TABLE doesn't accept variables but just scalar values. There are two ways to work around this:

- Write a C# assembly that mounts the command and issues the statement in an ad-hoc way.
- Move the second partition and not the first.

This article follows the second approach, keeping the first partition only to set a left boundary.

The second problem is that you can't issue any table command such TRUNCATE TABLE directly on the partition. If you have to copy the data or even delete it, you have to use a second auxiliary table. This table must have the same structure as the original one.

The following schema shows how the left part works.



The following code does the process.

Copy

```
--
**************************************************************************
**************
--
-- Summary: Range partition tables CDR and CDR_AUX
--      - Creates partition function pfDaily on the End_Date column of
the CDR table, so that each partition contains one day of data.
--      - Creates partition scheme pfDaily to map the partitions to
filegroups. All partitions reside on the same filegroup (Primary).
--      - Drops and re-creates the CDR table specifying the partition
scheme pfDaily as the location for the table.
--      - Creates partition function pfDaily_Aux on the End_Date column
of the CDR_AUX table.
--      - Creates partition scheme pfDaily_Aux to map the partitions to
filegroups. All partitions reside on the same filegroup (Primary).
--      - Drops and re-creates the CDR_AUX table specifying the
partition scheme pfDaily_Aux as the location for the table.
--
--
-- SQL Server Version: 9.00
--
--
**************************************************************************
**************

use AdventureWorks
```

```
go

--To drop all to re[eat the test remove the following lines
--truncate table cdr
--drop table cdr
--drop pARTITION SCHEME pfDaily
--drop partition function pfDaily;
--truncate table cdr_aux
--drop table cdr_aux
--drop pARTITION SCHEME pfDaily_aux
--drop partition function pfDaily_aux;

create partition function pfDaily (datetime)
as RANGE RIGHT for values(
'2005-05-07', '2005-05-08', '2005-05-09', '2005-05-10', '2005-05-11',
'2005-05-12', '2005-05-13', '2005-05-14',
'2005-05-15', '2005-05-16', '2005-05-17', '2005-05-18', '2005-05-19',
'2005-05-20', '2005-05-21', '2005-05-22',
'2005-05-23', '2005-05-24', '2005-05-25', '2005-05-26', '2005-05-27',
'2005-05-28', '2005-05-29', '2005-05-30',
'2005-05-31', '2005-06-01', '2005-06-02', '2005-06-03', '2005-06-04',
'2005-06-05', '2005-06-06', '2005-06-07',
'2005-06-08', '2005-06-09', '2005-06-10', '2005-06-11', '2005-06-12',
'2005-06-13', '2005-06-14', '2005-06-15',
'2005-06-16', '2005-06-17', '2005-06-18', '2005-06-19', '2005-06-20',
'2005-06-21', '2005-06-22', '2005-06-23',
'2005-06-24', '2005-06-25', '2005-06-26', '2005-06-27', '2005-06-28',
'2005-06-29', '2005-06-30', '2005-07-01',
'2005-07-02', '2005-07-03', '2005-07-04')
go

-- This Partition MUST be on Left side, so the data MUST be the
-- day before of the first day.

create partition function pfDaily_Aux (datetime)
as RANGE RIGHT for values(
'2005-05-07',
'2005-05-08')
go

--Both partitions will be placed at the same FileGroup since the
--system is planned to run on SAN disk.

CREATE PARTITION SCHEME pfDaily as partition pfDaily all
to ([primary])
go

CREATE PARTITION SCHEME pfDaily_Aux as partition pfDaily_Aux all
```

```
to ([primary])
go

CREATE TABLE [dbo].[CDR] (
   [ID_CDR] [int] NOT NULL ,
   [Route] [int] NULL ,
   [Direction] [tinyint] NULL ,
   [IAM_Date] [datetime] NOT NULL ,
   [ACM_Date] [datetime] NULL ,
   [ANM_Date] [datetime] NULL ,
   [REL_Date] [datetime] NULL ,
   [RLC_Date] [datetime] NULL ,
   [End_Date] [datetime] NOT NULL
) on pfDaily ([End_Date])
GO

CREATE  CLUSTERED INDEX [IX_End_Date]
ON [dbo].[CDR]([End_Date]) ON  pfDaily ([End_Date])
GO

ALTER TABLE [dbo].[CDR] WITH NOCHECK ADD
   CONSTRAINT [PK_CDR] PRIMARY KEY  NONCLUSTERED
   (
      [Id_CDR],
      [End_Date]
   )  on pfDaily ([End_Date])
GO

CREATE TABLE [dbo].[CDR_AUX] (
   [ID_CDR] [int] NOT NULL ,
   [Route] [int] NULL ,
   [Direction] [tinyint] NULL ,
   [IAM_Date] [datetime] NOT NULL ,
   [ACM_Date] [datetime] NULL ,
   [ANM_Date] [datetime] NULL ,
   [REL_Date] [datetime] NULL ,
   [RLC_Date] [datetime] NULL ,
   [End_Date] [datetime] NOT NULL
) on pfDaily_Aux ([End_Date])
GO

CREATE  CLUSTERED INDEX [IX_End_Date]
ON [dbo].[CDR_AUX]([End_Date]) ON  pfDaily_aux ([End_Date])
GO

ALTER TABLE [dbo].[CDR_AUX] WITH NOCHECK ADD
   CONSTRAINT [PK_CDR_AUX] PRIMARY KEY  NONCLUSTERED
   (
```

```
      [Id_CDR],
      [End_Date]
   )  on pfDaily_aux ([End_Date])
```

To check the structure created you can issue:

Copy

```
select * from sys.partition_range_values
 where function_id in (select function_id
      from sys.partition_functions
       where name in ('pfDaily', 'pfDaily_aux'))
```

The right part:

Copy

```
--
*************************************************************************
**************
--
-- Summary:    Managing a Range Partitioned Table
--      Creates a new partition at Right
--      Add a new partition on the end of table CDR for the next day
--      There is no need to pass any parameter. The routine reads CDR
--      metadata to discover right boundary.
--
--
*************************************************************************
**************

USE AdventureWorks;
GO

CREATE PROCEDURE PRC_ADD_PARTITION_RIGHT_ON_CDR
as

DECLARE @Day datetime

SET @Day = cast((select top 1 [value] from sys.partition_range_values
       where function_id = (select function_id
              from sys.partition_functions
              where name = 'pfDaily')
      order by boundary_id DESC) as datetime)

SET @Day = DATEADD(DAY, 1, @Day)
```

```
ALTER PARTITION SCHEME pfDaily
NEXT USED [PRIMARY];

ALTER PARTITION FUNCTION pfDaily()
SPLIT RANGE (@Day);

GO
```

The left part:

Copy

```
--
************************************************************************
**************
--
-- Summary:       Managing a Range Partitioned Table
--          Delete data on 2nd most left partition.
--          It means that the most left partition will always stay there
--          to guarantee the size of the second one. This one will be
--          moved. The most left partition will be empty.
--
--
************************************************************************
**************

USE AdventureWorks;
GO

CREATE PROCEDURE PRC_DEL_PARTITION_LEFT_ON_CDR
AS

ALTER PARTITION SCHEME pfDaily
NEXT USED [PRIMARY];

ALTER PARTITION SCHEME pfDaily_Aux
NEXT USED [PRIMARY];

DECLARE @Day      datetime
DECLARE @Day_Next2   datetime
DECLARE @Scratch   varchar(2000)

SET @Day = cast((select top 1 [value] from sys.partition_range_values
          where function_id = (select function_id
                  from sys.partition_functions
                  where name = 'pfDaily')
         order by boundary_id) as datetime)
```

```
SET @Day_Next2  = DATEADD(DAY,  2, @Day)

-- STEP 1
-- Add a new partition to table CDR_Aux to hold the
-- Data from 2nd Left Partition of CDR.

ALTER PARTITION FUNCTION pfDaily_Aux()
SPLIT RANGE (@Day_Next2);


-- STEP 2
-- Move the data for 2nd FAR LEFT Partition from table CDR to
-- table CDR_AUX.

ALTER TABLE CDR
SWITCH PARTITION 2
TO CDR_AUX PARTITION 2;

-- STEP 3
-- Merge the 1st and 2nd partitions of table CDR.

ALTER PARTITION FUNCTION pfDaily()
MERGE RANGE (@Day);

-- STEP 4
-- Merge the partition of table CDR_AUX
-- with the first partition.

ALTER PARTITION FUNCTION pfDaily_Aux()
MERGE RANGE (@Day);

-- delete the data on CDR_AUX
TRUNCATE TABLE CDR_AUX

GO
```

At this point, you are done!

To see how it works, just call the stored procedures and look at the schema.

https://msdn.microsoft.com/en-us/library/aa964122(SQL.90).aspx

Scenario 3 – not tested

Here is the code with which you can automate the processing of managing a sliding window scenario. The procedure has been created to manage a sliding window scenario for a partitioned table, http://www.databasejournal.com/features/mssql/partitioning-in-sql-server-managing-sliding-window-scenario.html

Has yearly partitions, this stored procedure can be scheduled to run on the first day of every year. The procedure checks if the partition for the current year is already created or not; if not then it does following:

- Take out the oldest partition from the partitioned table into the staging table.
- Modify the partition function to remove the boundary value of the oldest partition with the MERGE clause with ALTER PARTITION FUNCTION statement.
- Modify the partition scheme to designate the new file group to be used by the newest partition with the NEXT USED clause of the ALTER PARTITION SCHEME statement.
- Modify the partition function to add the boundary value of the newest partition with the SPLIT clause with the ALTER PARTITION FUNCTION statement.

```
CREATE PROCEDURE [dbo].[ManageFactSlidingWindow]
/*****************************************************************************
PROCEDURE NAME:     [ManageFactSlidingWindow]
AUTHOR:             Arshad Ali
CREATED:            02/24/2013
DESCRIPTION:        This stored procedure manages sliding   window for the partitioned table

VERSION HISTORY:
DATE   EMAIL  Company          DESCRIPTION

*****************************************************************************/
AS
BEGIN
    BEGIN TRY --Start the Try Block

    DECLARE @MinYear int
    DECLARE @MaxYear int
    DECLARE @CurrentYear   int = YEAR(GETDATE())

    SELECT @MinYear = MIN(CONVERT(int, Value)), @MaxYear = MAX(CONVERT(int, Value)) FROM
sys.partition_functions f
    INNER JOIN sys.partition_range_values r
    ON f.function_id = r.function_id
    WHERE f.name = 'FactPartitionFunction'

    IF @MaxYear < @CurrentYear AND NOT EXISTS (SELECT TOP 1 1 FROM
dbo.ArchiveFactResellerSalesWithPartition)
```

```
    BEGIN
        BEGIN TRANSACTION

            ALTER TABLE dbo.FactResellerSalesWithPartition SWITCH PARTITION 1 TO
dbo.ArchiveFactResellerSalesWithPartition

            ALTER PARTITION FUNCTION FactPartitionFunction()
            MERGE RANGE (@MinYear)

            ALTER PARTITION SCHEME FactPartitionScheme
            NEXT USED DM

            ALTER PARTITION FUNCTION FactPartitionFunction()
            SPLIT RANGE (@MaxYear+1)

        COMMIT TRANSACTION
    END
    END TRY
    BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRAN --RollBack in case of Error

    END CATCH
END
```

The staging table in which we moved data from the oldest partition can be either dropped or can be archived. Even to archive, we have two options, move data into a regular table or move data into a partitioned archive table. In the code below, I am moving data from the staging table to another table in another database for archival.

```
CREATE PROCEDURE [dbo].[ArchiveFact]
/*****************************************************************************
PROCEDURE NAME:     [ArchiveFact]
AUTHOR:          Arshad Ali
CREATED:         02/24/2013
DESCRIPTION:       This stored procedure is archive data from   stage table to archive tables in archive
database

VERSION HISTORY:
DATE   EMAIL  Company          DESCRIPTION

*****************************************************************************/
AS
BEGIN
    BEGIN TRY --Start the Try Block
```

```
        IF OBJECT_ID('ArchiveFactResellerSalesWithPartition') IS NOT NULL
        BEGIN
            BEGIN TRANSACTION
                INSERT INTO AdventureWorksDW2012Archive.dbo.FactResellerSalesWithPartition
                SELECT * FROM AdventureWorksDW2012.dbo.ArchiveFactResellerSalesWithPartition
                TRUNCATE TABLE AdventureWorksDW2012.dbo.ArchiveFactResellerSalesWithPartition
            COMMIT TRANSACTION
        END
    END TRY
    BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRAN --RollBack in case of Error

    END CATCH
END
```

SSIS Simple ETL Lab steps: Basic Steps - http://msdn.microsoft.com/en-us/library/ms169917.aspx

Windows Failover Clustering (WSFC) & SQL Failover Cluster Instances (FCI)

A failover cluster is a combination of one or more physical disks in a Microsoft Cluster Service (MSCS) cluster group, known as a resource group, that are participating nodes of the cluster. The resource group is configured as a failover clustered instance that hosts an instance of SQL Server.

A SQL Server failover clustered instance appears on the network as if it were a single computer, but has functionality that provides failover from one node to another if one node becomes unavailable. For more information, see AlwaysOn Failover Cluster Instances (SQL Server).

Failover clusters provide high-availability support for an entire Microsoft SQL Server instance, in contrast to database mirroring, which provides high-availability support for a single database. Database mirroring works between failover clusters and, also, between a failover cluster and a nonclustered host.

AlwaysOn Availability Groups, the high availability and disaster recovery solution introduced in SQL Server 2016, requires Windows Server Failover Clustering (WSFC). Also, though AlwaysOn Availability Groups is not dependent upon SQL Server Failover Clustering, you can use a failover clustering instance (FCI) to host an availability replica for an availability group. It is important to know the role of each clustering technology, and to know what considerations are necessary as you design your AlwaysOn Availability Groups environment.

References:

1. Partitioned Tables and Indexes: https://msdn.microsoft.com/en-us/library/ms190787.aspx
2. Create Partitioned Tables and Indexes: https://msdn.microsoft.com/en-us/library/ms188730.aspx
3. SQL Server Partition Management: (CMD line utility): http://sqlpartitionmgmt.codeplex.com/
4. How to Implement an Automatic Sliding Window in a Partitioned Table on SQL Server 2005: https://msdn.microsoft.com/library/aa964122(sql.90).aspx
5. Top 10 steps to optimize data access in SQL Server: Part V (Optimize database files and apply partitioning): http://www.codeproject.com/Articles/43629/Top-steps-to-optimize-data-access-in-SQL-Serv
6. Database Mirroring and SQL Server Failover Cluster Instances: https://technet.microsoft.com/en-us/library/ms191309(v=sql.110).aspx
7. AlwaysOn Failover Cluster Instances (SQL Server): https://msdn.microsoft.com/en-us/library/ms189134.aspx
8. Failover Clustering and AlwaysOn Availability Groups (SQL Server): https://msdn.microsoft.com/en-us/library/ff929171.aspx
9. SSIS Home page 2016: https://msdn.microsoft.com/en-us/library/ms141026
10. AlwaysOn for SSISDB: https://msdn.microsoft.com/en-us/library/bb522534#AlwaysOn
11. The Excel Connection Manager, the Excel Source and the Excel Destination, and the SQL Server Import and Export Wizard now support Excel 2013 data sources. 01/07/2016
12. Microsoft SQL Server Data Tools ... http://msdn.microsoft.com/en-us/data/tools.aspx
13. SQL Server Integration Services: http://technet.microsoft.com/en-us/library/ms141026.aspx
14. SSIS Tutorial: Creating a Simple ETL Package - http://msdn.microsoft.com/en-us/library/ms169917.aspx
15. Create the SSIS Catalog: http://msdn.microsoft.com/en-us/library/gg471509.aspx

Other References:

1. BI for Dummies: http://it-ebooks.info/book/1388/