Microsoft SQL Server 9.0 Technical Articles

How to Implement an Automatic Sliding Window in a Partitioned Table on SQL Server 2005

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.

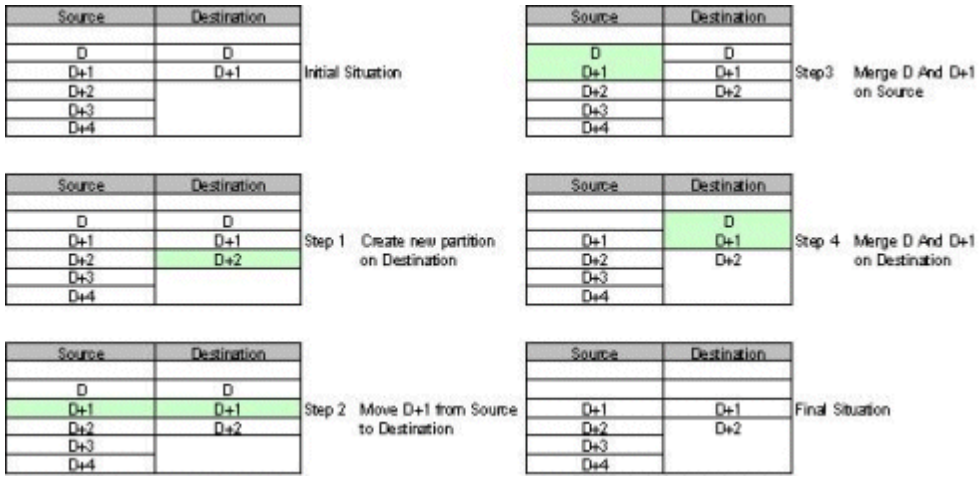
The first problem encountered in transforming the script is the ADD CONSTRAINT statement used to establish a left boundary on the source table. In order to automate the script, it is necessary to have all dates based on the partition 1 date. In earlier versions of SQL Server, this was accomplished by 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 Code

```
--*************************************************************************************************
```

```
--
-- Summary: Range partition tables CDR and CDR_AUX
--      - Creates partition function pfDaily on the End_Date column of the CDR table, so tl
--      - Creates partition scheme pfDaily to map the partitions to filegroups. All partit
--      - Drops and re-creates the CDR table specifying the partition scheme pfDaily as the
--      - Creates partition function pfDaily_Aux on the End_Date column of the CDR_AUX tab
--      - Creates partition scheme pfDaily_Aux to map the partitions to filegroups. All pa
--      - Drops and re-creates the CDR_AUX table specifying the partition scheme pfDaily_Au
--
--
-- 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', '2005-0
'2005-05-15', '2005-05-16', '2005-05-17', '2005-05-18', '2005-05-19', '2005-05-20', '2005-0
'2005-05-23', '2005-05-24', '2005-05-25', '2005-05-26', '2005-05-27', '2005-05-28', '2005-0
'2005-05-31', '2005-06-01', '2005-06-02', '2005-06-03', '2005-06-04', '2005-06-05', '2005-0
'2005-06-08', '2005-06-09', '2005-06-10', '2005-06-11', '2005-06-12', '2005-06-13', '2005-0
'2005-06-16', '2005-06-17', '2005-06-18', '2005-06-19', '2005-06-20', '2005-06-21', '2005-0
'2005-06-24', '2005-06-25', '2005-06-26', '2005-06-27', '2005-06-28', '2005-06-29', '2005-0
'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 Code

```
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 Code

```
--*****************************************************************************************
--
-- 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 Code

```
--*****************************************************************************************
--
-- 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.