[<< Are you a DBA Monkey?](http://weblogs.sqlteam.com/dang/archive/2008/08/01/Are-you-a-DBA-Monkey.aspx" \o "previous post) | [Home](http://weblogs.sqlteam.com/dang/default.aspx) | [Automating Sliding Window Maintenance >>](http://weblogs.sqlteam.com/dang/archive/2008/09/14/Automating-Sliding-Window-Maintenance.aspx)

[Sliding Window Table Partitioning](http://weblogs.sqlteam.com/dang/archive/2008/08/30/Sliding-Window-Table-Partitioning.aspx)

SQL Server table partitioning provides a great way to manage a time-based sliding window. By mapping each time period to an individual partition, old data can be efficiently purged or archived using a nearly instantaneous switch out of an entire partition. However, there are a couple of aspects of a time-base sliding window strategy that require some thought and planning.

**RANGE LEFT or RIGHT**

The RANGE LEFT or RIGHT partition function specification indicates which partition includes the exact match on the partition boundary. It is especially important to consider both the RANGE specification and boundary value when the partition function data type includes a time component (smalldatetime, datetime, datatime2 and datetimeoffset) because one usually wants all data for a given date in the same partition.

In the case of a RANGE LEFT, specify the latest time allowable time for the data type and beware of implicit rounding. For example, a boundary value of ‘2008-08-30T23:59:59.999’ for a datetime partition function will get rounded up to ‘2008-08-31T00:00:00.000’ and result in data for midnight (‘2008-08-31T00:00:00’) getting inserted into the left partition instead of the right. To avoid this rounding issue, specify ‘2008-08-30T23:59:59.997’ instead.

The following table shows examples that ensure ‘2008-08-31T00:00:00’ always ends up in the RIGHT partition. The datetimeoffset examples assume UTC time zone (+00:00) is the desired date boundary but this can be adjusted as desired.

|  |  |
| --- | --- |
| **Partition Function Datatype** | **Range LEFT Boundary Specification** |
| smalldatetime | ‘2008-08-30T23:59:00’ |
| datetime | ‘2008-08-30T23:59:59.997’ |
| datetime2(0) | ‘2008-08-30T23:59:59’ |
| datetime2(1) | ‘2008-08-30T23:59:59.9 |
| datetime2(2) | ‘2008-08-30T23:59:59.99’ |
| datetime2(3) | ‘2008-08-30T23:59:59.999’ |
| datetime2(4) | ‘2008-08-30T23:59:59.9999’ |
| datetime2(5) | ‘2008-08-30T23:59:59.99999’ |
| datetime2(6) | ‘2008-08-30T23:59:59.999999’ |
| datetime2(7) | ‘2008-08-30T23:59:59.9999999’ |
| datetime2 | ‘2008-08-30T23:59:59.9999999’ |
| datetimeoffset(0) | ‘2008-08-30T23:59:59+00:00’ |
| datetimeoffset(1) | ‘2008-08-30T23:59:59.9+00:00’ |
| datetimeoffset(2) | ‘2008-08-30T23:59:59.99+00:00’ |
| datetimeoffset(3) | ‘2008-08-30T23:59:59.999+00:00’ |
| datetimeoffset(4) | ‘2008-08-30T23:59:59.9999+00:00’ |
| datetimeoffset(5) | ‘2008-08-30T23:59:59.99999+00:00’ |
| datetimeoffset(6) | ‘2008-08-30T23:59:59.999999+00:00’ |
| datetimeoffset(7) | ‘2008-08-30T23:59:59.9999999+00:00’ |
| datetimeoffset | ‘2008-08-30T23:59:59.9999999+00:00’ |

With a RANGE RIGHT partition function, specify only the desired date (or specify date with time of midnight). This will ensure that all data for the specified date is in the same partition.

**Avoid Data movement with SPLIT and MERGE**

SPLIT and MERGE is very fast when the partitions are empty because no data movement is necessary. The goal is to plan sliding window partition maintenance accordingly. After the start of a new period, SWITCH out old data and remove the vacated empty partition with MERGE. SPLIT the last partition (empty) in anticipation of future data.

The choice of a RANGE LEFT or RIGHT partition also affects how one uses SWITCH, SPLIT and MERGE to maintain the sliding window.

**Sliding a LEFT Partition Function Window**

At the start of a new period with a LEFT partition function, partition 1 contains the old data, the second from last partition contains the current data and then last partition (empty) is for future data. The partition can be maintained as follows:

1) SWITCH out partition 1 to a staging table for archive/purge

2) MERGE empty partition 1 with non-empty partition 2 to form a new partition 1 (with data)

3) SPLIT last partition so that the last 2 partitions are now empty

4) Repeat steps 1 through 3 after the start of the next period

**Sliding a RIGHT Partition Function Window**

The RIGHT partition maintenance strategy is slightly different that the LEFT strategy. The difference is that partition 2 rather than partition 1 contains the old data at the start of a new period. However, just like the LEFT strategy, the second from last partition contains the current data and then last partition (empty) is for future data. The partition can be maintained as follows:

1) SWITCH out partition 2 to a staging table for archive/purge

2) MERGE empty partition 1 with empty partition 2 to form a new partition 1 (empty)

3) SPLIT last partition so the last 2 partitions are now empty

4) Repeat steps 1 through 3 after the start of the next period

**Automating the Sliding Window Maintenance**

In my next post, I’ll share a script that will help automate sliding window maintenance.

posted @ Saturday, August 30, 2008 6:58 PM

[Print](javascript:window.print();)

**Comments on this entry:**

[**#**](http://weblogs.sqlteam.com/dang/archive/2008/08/30/Sliding-Window-Table-Partitioning.aspx#42369)**re: Sliding Window Table Partitioning**

Left by Sunil at 1/20/2010 7:52 AM

Gravatar

Thanks for great post.

[**#**](http://weblogs.sqlteam.com/dang/archive/2008/08/30/Sliding-Window-Table-Partitioning.aspx#45237)**re: Sliding Window Table Partitioning**

Left by Chirag Shah at 12/9/2010 10:59 AM

Gravatar

Dan Guzman,  
  
An excellent post, I must admire you to explain things that sometime rather complicated in much simple way.  
  
Hopefully, you will write something new on the SQL server sometime soon in your blog. I'll keep checking...