SQL Server Can't Handle Milliseconds

Posted At : October 19, 2010 10:36 AM | Posted By : Jon Hartmann
Related Categories: Microsoft Tools

I've started up a new job, and one of the tasks I'm going to have to tackle is creating a system where nearly every record has an effective date. Only the information with the most current date that has happened so far is considered in use. This means that I've got to do some crazy date manipulation to keep things running smoothly. While working on some stored procedures, I found an issue with SQL Server and its handling of `datetime` values when incrementing in milliseconds.

The Task

I need to setup a stored procedure that takes in information for a given record that is set to be effective on a certain day. The system should automatically inspect the database for existing entries for this record on that day and pick the next closest possible `datetime` value for use when inserting to the database. To accomplish this task I decided to use SQL Server's `DATEADD()` function, and the `ms`, or millisecond, date part since its the smallest unit handled by a standard `datetime` column.

The Problem

When incrementing a given `datetime` by 1 millisecond, SQL Server doesn't understand that there was a change in value. For example, try running the following code:

```
DECLARE @TestDatetime datetime;
DECLARE @Increment int;
DECLARE @TestDatetimeWithIncrement datetime;SET @TestDatetime = '2010-10-19 12:00:00.000';SET @Increment = 1;SET @TestDatetimeWithIncrement = DATEADD(ms, @Increment, @TestDatetime);SELECT
 @TestDatetime AS TestDateTime,
 @Increment AS MillisecondIncrement,
 @TestDatetimeWithIncrement AS TestPlusIncrement,
 CASE @TestDatetime
  WHEN @TestDatetimeWithIncrement THEN 'Yes'
  ELSE 'No'
 END AS Match,
 DATEDIFF(ms, @TestDatetime, @TestDatetimeWithIncrement) AS MillisecondDifference;
```

The result looks something like this:

TestDateTime	2010-10-19 12:00:00.000
MillisecondIncrement	1
TestPlusIncrement	2010-10-19 12:00:00.000
Match	YES
MillisecondDifference	0

Thats no good... SQL Server can't figure out that I incremented the value at all. Whats even weird is if I try increasing the increment to `2`:

TestDateTime	2010-10-19 12:00:00.000
MillisecondIncrement	2
TestPlusIncrement	2010-10-19 12:00:00.003
Match	No
MillisecondDifference	3

Thats even worse... SQL Server knows it changed the value, but it gets its calculation wrong and thinks it increased by 3 milliseconds. At 3 milliseconds things seem ok, but increase to 4 milliseconds and you get this:

TestDateTime	2010-10-19 12:00:00.000
MillisecondIncrement	4
TestPlusIncrement	2010-10-19 12:00:00.003
Match	No
MillisecondDifference	3

So SQL Server can't seem to get anything right on that. This lead me to believe that SQL Server is actually incrementing by some fraction of a millisecond and then rounding the value. This lead me to test sending the incremented value through `DATEADD()` a second time, and got pretty much the results I expected... incrementing by 8 was the worst:

```
DECLARE @TestDatetime datetime;
DECLARE @Increment int;
DECLARE @TestDatetimeWithIncrement datetime;SET @TestDatetime = '2010-10-19 12:00:00.000';SET @Increment = 8;SET @TestDatetimeWithIncrement = DATEADD(ms, @Increment, @TestDatetime);SELECT
 @TestDatetime AS TestDateTime,
 @Increment AS MillisecondIncrement,
 @TestDatetimeWithIncrement AS TestPlusIncrement,
 DATEDIFF(ms, @TestDatetime, @TestDatetimeWithIncrement) AS MillisecondDifference,
 DATEADD(ms, @Increment, @TestDatetimeWithIncrement) AS TestPlusIncrementX2,
 DATEDIFF(ms, @TestDatetime, DATEADD(ms, @Increment, @TestDatetimeWithIncrement)) AS X2MillisecondDifference;
```

Brought back:

TestDateTime	2010-10-19 12:00:00.000
MillisecondIncrement	8
TestPlusIncrement	2010-10-19 12:00:00.007
MillisecondDifference	6
TestPlusIncrementX2	2010-10-19 12:00:00.013
X2MillisecondDifference	13

Notice that it increments by 8, shows a 7 second difference, detects a 6 millisecond difference, and than when incremented again shows a 13 millisecond gap (6.5 ms per increment).

The Solution

I have no idea how to get around this error, so I cheated... incrementing by 10 seems to be reliable, so I went with it. Sure, its a cop, but it works, and really, 8,640,000 possible effective dates per day for any given record should be more than enough for anyone.