**A SQL Server Date Handling Issue**

**By** [**David Ozono**](http://www.sqlservercentral.com/Authors/Articles/David_Ozono/1962000/)**, 2016/06/20**

**FROM:** [**http://www.sqlservercentral.com/articles/SQL+Server+Date+handling/141168/**](http://www.sqlservercentral.com/articles/SQL+Server+Date+handling/141168/)

I was debugging a stored procedure to find out why it failed and came across a scenario which was entirely new to me. The stored procedure wanted to assign an ID column based on the following condition:

WHERE CONVERT(CHAR(8), StartDate,108) BETWEEN FROM\_TMS AND TO\_TMS

The FROM\_TMS and TO\_TMS fields are of TIME datatype, as shown in the following data sample:

|  |  |  |
| --- | --- | --- |
| **ID** | **FROM\_TMS** | **TO\_TMS** |
| 1 | 23:59:00 | 23:59:59 |
| 2 | 00:00:00 | 00:00:59 |
| 3 | 00:01:00 | 00:01:59 |
| 4 | 00:02:00 | 00:02:59 |
| 5 | 00:03:00 | 00:03:59 |
| 6 | 00:04:00 | 00:04:59 |
| 7 | 00:05:00 | 00:05:59 |

I believe that when this condition was added to the JOIN clause of the stored procedure, an assumption was made that the StartDate column is of the DateTime datatype. Hence the developer thought they were extracting the TIME portion of the date from a datetime field.

I say it was an assumption because in reality, the StartDate column was a VARCHAR(20). The data in this column had the following sample data:

|  |  |
| --- | --- |
| **DateID** | **StartDate** |
| A | 18-Aug-2014 |
| D | 19-Aug-2014 |
| E | 20-Aug-2014 |
| F | 21-Aug-2014 |
| K | 22-Aug-2014 |
| S | 25-Aug-2014 |
| B | 26-Aug-2014 |

When SQL Server came to resolve the WHERE clause, the data presented was

'18-AUG-2'  BETWEEN FROM\_TMS AND TO\_TMS

and it assigned an ID of 2 based on the TIME table shown above. I did not understand how SQL Server was able to assign an ID of 2, which indicates that the TIME element of the date was between 00:00:00 and 00:00:59.It did this all the time and it looked like the code was working until the failure which led to this article.

**What happened?**

I thought straight away that the fact that SQL Server was presented with the string '18-AUG-2' was causing the failure of the stored procedure because there was no way SQL Server was going to be able to return a valid ID with a string value of 18-AUG-2 being compared against TIME columns.

**What did SQL Server do?**

1. In order to resolve the WHERE clause, SQL Server will convert the text returned from the CONVERT(CHAR(8), StartDate,108) first to a datetime. So using the sample above, It will convert '18-AUG-2' to a datetime before using the TIME portion. In effect doing the equivalent of the following;

SELECT CAST('18-AUG-2'as DATE)

This returns the value: 2002-08-18

2. SQL Server will then use the midnight of that day 00:00:00 as the TIME portion of the date since none was provided.

3. This  will therefore result in the assignment of an ID of 2 to that day (as per above sample data)

So the stored procedure did not encounter any issues until the day we received a StartDate value of 29-Feb-2016 which when applied to the WHERE clause resulted in a string of 29-FEB-2. This, as you would notice is, the leap day in a leap year 2016. Yes, it failed, but I could not understand why it failed because 29-Feb-2016 is a valid leap date in a valid leap year.

**This Is Why It Failed**

The reason for failure confirms my earlier observation about what SQL Server was doing with the strings passed into the WHERE clause. As I said earlier, SQL Server did the following;

SELECT CAST('29-FEB-2'as DATE)

This returns the value: 2002-02-29. This is not a valid day in the calendar because year 2002 is not a leap year. Hence the conversion to a datetime failed. Whereas 29 FEB 2016 is a valid day, SQL Server's attempt to use that date based on the condition was what actually caused it to fail.

Can you work out the next leap year this code will succeed on a leap date?

I worked out that the next leap year on which it will succeed is the years starting with 4000. I don't know about you but I don't plan to hang around for that long!