What is the significance of 1/1/1753 in SQL Server?

Asked 14 years, 5 months ago Modified 2 months ago

Viewed 177k times

557

Why 1753? What do they have against 1752? My great great great great great great grandfather would be very offended.

sql-server t-sql datetime localization internationalization

Share

Improve this question

Follow

edited Nov 21, 2016 at 14:06

gotqn

43.5k ● 46 ● 165 ● 254

asked Jul 22, 2010 at 15:28

Daniel

7,008 ● 6 ● 32 ● 30

7 Answers

Sorted by: Highest score (default)

951

The decision to use 1st January 1753 (`1753-01-01`) as the minimum date value for a datetime in SQL Server goes back to its Sybase origins.

The significance of the date itself though can be attributed to this man.

Philip Stanhope, 4th Earl of Chesterfield. Who steered the Calendar (New Style) Act 1750 through the British Parliament. This legislated for the adoption of the Gregorian calendar for Britain and its then colonies.

There were some missing days (internet-archive-link) in the British calendar in 1752 when the adjustment was finally made from the Julian calendar. September 3, 1752 to September 13, 1752 were lost.

Kalen Delaney explained the choice this way

> So, with 12 days lost, how can you compute dates? For example, how can you compute the number of days between October 12, 1492, and July 4, 1776? Do you include those missing 12 days? To avoid having to solve this problem, the

> original Sybase SQL Server developers decided not to allow dates before 1753. You can store earlier dates by using character fields, but you can't use any datetime functions with the earlier dates that you store in character fields.

The choice of 1753 does seem somewhat anglocentric however as many catholic countries in Europe had been using the calendar for 170 years before the British implementation (originally delayed due to opposition by the church). Conversely many countries did not reform their calendars until much later, 1918 in Russia. Indeed the October Revolution of 1917 started on 7 November under the Gregorian calendar.

Both `datetime` and the new `datetime2` datatype mentioned in Joe's answer do not attempt to account for these local differences and simply use the Gregorian Calendar.

So with the greater range of `datetime2`

```
SELECT CONVERT(VARCHAR, DATEADD(DAY,-5,CAST('1752-09-1
```

Returns

```
Sep  8 1752 12:00AM
```

One final point with the `datetime2` data type is that it uses the proleptic Gregorian calendar projected

backwards to well before it was actually invented so is of limited use in dealing with historic dates.

This contrasts with other Software implementations such as the Java Gregorian Calendar class which defaults to following the Julian Calendar for dates until October 4, 1582 then jumping to October 15, 1582 in the new Gregorian calendar. It correctly handles the Julian model of leap year before that date and the Gregorian model after that date. The cutover date may be changed by the caller by calling `setGregorianChange()`.

A fairly entertaining article discussing some more peculiarities with the adoption of the calendar can be found here.

Share Improve this answer Follow

edited Apr 26, 2020 at 10:54

answered Jul 22, 2010 at 15:31

Martin Smith

452k ● 94 ● 767 ● 870

89 +1 for great portrait of non-offended great great great great great great great grandfather. Also other shining attributes of this answer. – Smandoli Sep 22, 2010 at 21:52

106 +1 for an answer that is both technical and historical. On a board frequented by heavy left-brainers, this is an enjoyable read. And yes, I did go to a liberal arts college. – Matt Hamsmith Jan 25, 2011 at 20:57

1 Regarding this link: imagine someone born in Sweden on the 30th of February 1712 - they'd have never aged! :P

Also, what on earth did Lithuania and Latvia do for all that time!? – Isaac Reefman Feb 5, 2019 at 1:33

1 Speaking of this man, the Chesterfield sofa is named after him as well. – Alex Nov 13, 2023 at 17:25

112

Your great great great great great great great grandfather should upgrade to SQL Server 2008 and use the DateTime2 data type, which supports dates in the range: 0001-01-01 through 9999-12-31.

Share Improve this answer

Follow

answered Jul 22, 2010 at 15:36

Joe Stefanelli

136k ● 21 ● 240 ● 238

41 @CRMay: Tell him you plan to start work on Y10K compliance on Thursday, 5000-01-02; that leaves you just under 5 millennia to get it fixed. – Jonathan Leffler Jul 24, 2010 at 1:44

10 mm I'm guessing someone missed the answer to the actual question: why *1753*, of all years? – sehe Sep 2, 2011 at 21:06

1 Yess.... but try getting that change of data type through in a company that has a massive installed base of SQL Server databases, and more lines of defence against the dreaded enemy CHANGE than the US had against Russian ICBMs at the height of the Cold War... – SebTHU Mar 7, 2016 at 11:00

1 I think it is a better answer, being concise and telling the solution instead of history, would you query using history or data type – abdul qayyum May 23, 2018 at 7:06

3 I enjoy both learning and having a good laugh when reading StackOverflow! – Tore Aurstad May 28, 2020 at 14:28

30

1752 was the year of Britain switching from the Julian to the Gregorian calendar. I believe two weeks in September 1752 never happened as a result, which has implications for dates in that general area.

An explanation: (Internet Archive version)

Share Improve this answer

Follow

edited Sep 12, 2023 at 17:28

Cesar M ♦

101 ● 3 ● 12

answered Jul 22, 2010 at 15:34

Gian

13.9k ● 49 ● 54

23

This is whole story how date problem was and how Big DBMSs handled these problems.

> During the period between 1 A.D. and today, the Western world has actually used two main calendars: the Julian calendar of Julius Caesar and the Gregorian calendar of Pope Gregory XIII. The two calendars differ with respect to only one rule: the rule for deciding what a leap year is. In the Julian calendar, all years divisible by four are leap years. In the Gregorian calendar, all years divisible by four are leap years, except that

> years divisible by 100 (but not divisible by 400) are not leap years. Thus, the years 1700, 1800, and 1900 are leap years in the Julian calendar but not in the Gregorian calendar, while the years 1600 and 2000 are leap years in both calendars.
>
> When Pope Gregory XIII introduced his calendar in 1582, he also directed that the days between October 4, 1582, and October 15, 1582, should be skipped—that is, he said that the day after October 4 should be October 15. Many countries delayed changing over, though. England and her colonies didn't switch from Julian to Gregorian reckoning until 1752, so for them, the skipped dates were between September 4 and September 14, 1752. Other countries switched at other times, but 1582 and 1752 are the relevant dates for the DBMSs that we're discussing.
>
> Thus, two problems arise with date arithmetic when one goes back many years. The first is, should leap years before the switch be calculated according to the Julian or the Gregorian rules? The second problem is, when and how should the skipped days be handled?
>
> This is how the Big DBMSs handle these questions:
>
> - Pretend there was no switch. This is what the SQL Standard seems to require, although the standard document is unclear:

It just says that dates are "constrained by the natural rules for dates using the Gregorian calendar"—whatever "natural rules" are. This is the option that DB2 chose. When there is a pretence that a single calendar's rules have always applied even to times when nobody heard of the calendar, the technical term is that a "proleptic" calendar is in force. So, for example, we could say that DB2 follows a proleptic Gregorian calendar.

- Avoid the problem entirely. Microsoft and Sybase set their minimum date values at January 1, 1753, safely past the time that America switched calendars. This is defendable, but from time to time complaints surface that these two DBMSs lack a useful functionality that the other DBMSs have and that the SQL Standard requires.
- Pick 1582. This is what Oracle did. An Oracle user would find that the date-arithmetic expression October 15 1582 minus October 4 1582 yields a value of 1 day (because October 5–14 don't exist) and that the date February 29 1300 is valid (because the Julian leap-year rule applies). Why did Oracle go to extra trouble when the SQL Standard doesn't seem to require it? The answer is that users might require it. Historians and astronomers use this hybrid

> system instead of a proleptic Gregorian calendar. (This is also the default option that Sun picked when implementing the GregorianCalendar class for Java—despite the name, GregorianCalendar is a hybrid calendar.)

Source 1 and 2

Share Improve this answer

Follow

edited May 23, 2017 at 12:02

Community Bot

1 • 1

answered Aug 3, 2016 at 9:30

Somnath Muluk

57.5k • 38 • 224 • 228

9

Incidentally, Windows no longer knows how to correctly convert UTC to U.S. local time for certain dates in March/April or October/November of past years. UTC-based timestamps from those dates are now somewhat nonsensical. It would be very icky for the OS to simply refuse to handle any timestamps prior to the U.S. government's latest set of DST rules, so it simply handles some of them wrong. SQL Server refuses to process dates before 1753 because lots of extra special logic would be required to handle them correctly and it doesn't want to handle them wrong.

Share Improve this answer

answered Jul 22, 2010 at 15:42

Follow

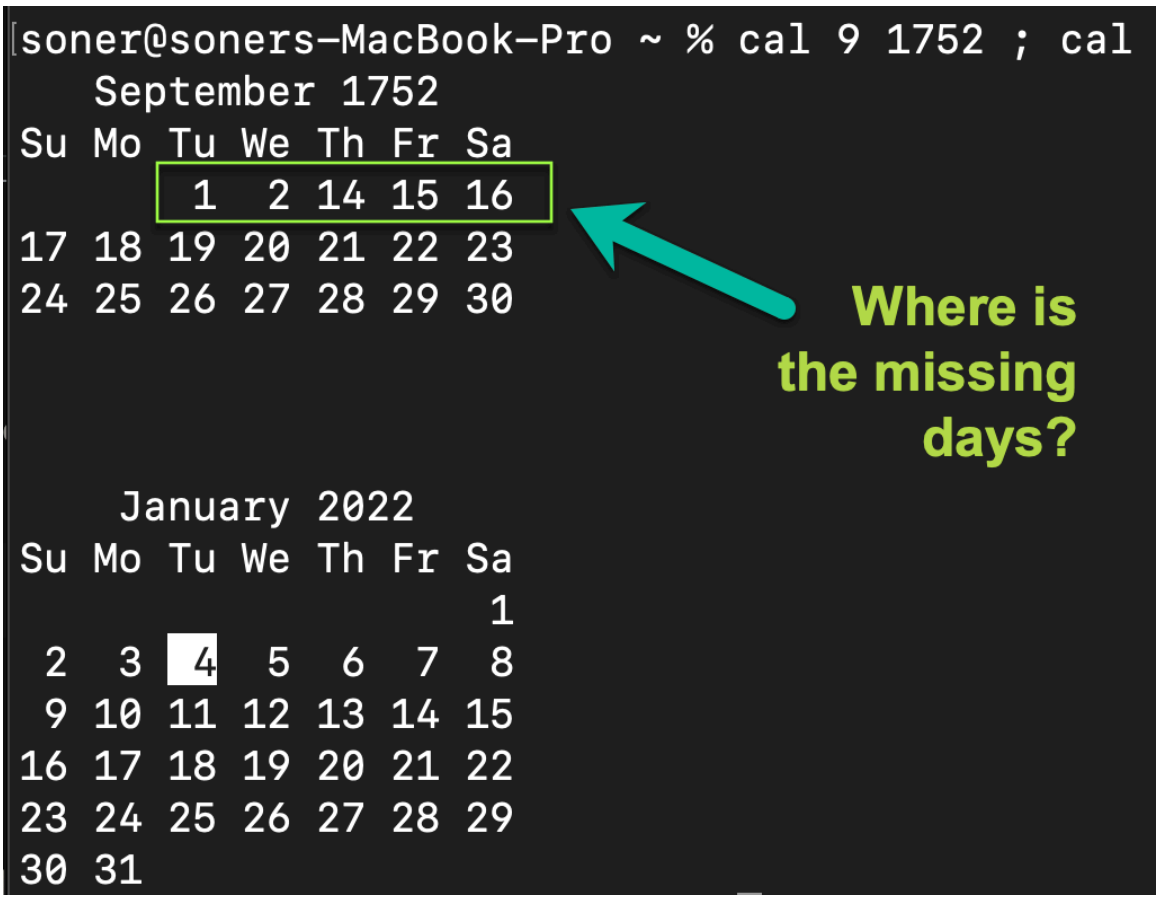
supercat
80.8k ● 9 ● 174 ● 220

5

For those wanting genuinely to be surprised in a gargantuan manner.

If you are on a Linux/Unix based system, you can try the following command involving cal command, (if it is used per se, it displays the date of today)

```
cal 9 1752 ; cal
```



For Online Linux Terminal, click here.

Share Improve this answer

Follow

answered Jan 4, 2022 at 11:07

Soner from The Ottoman Empire
20.8k ● 8 ● 92 ● 118

Can you please explain why this is? – Drazisil May 9 at 9:20

1 @Drazisil yes, historic-uk.com/HistoryUK/HistoryofBritain/… – Soner from The Ottoman Empire May 9 at 21:39

0

```
SELECT DATEDIFF(dd,'1753-01-01', '0001-01-01')
```

Out:

```
-639905
```

And anything below year `1` like `SELECT DATEDIFF(dd,'1753-01-01', '-1000-01-01')` crashes the `DATEDIFF()` function with:

> Conversion failed when converting date and/or time from character string.

You can cast it to integer as days from date "x" on, for example to get rid of the date data type in your database and reach slightly faster joins in large tables or if you really need to check dates earlier than `1753-01-01`. Then you need to add to year `1`:

- `639905` for a begin at `1753-01-01` and
- `693595` for a begin at `1900-01-01`.

The other way round, if you have an integer in the `700000`s as the days from year `1` and want to get back to a date, you need to subtract:

- `639905` for a begin at `1753-01-01` and
- `693595` for a beginn at `1900-01-01`.

Example:

`DATEADD(dd, 731573 - 639906, '17530101')` makes `2003-12-24 00:00:00.000`.
`DATEADD(dd, 731573 - 693596, '19000101')` makes `2003-12-24 00:00:00.000`.

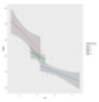
For more, see [Integer number in the 700000s as the days from year 1: how can this be cast in tsql to a date and back if the oldest datetime date is 1753-01-01? - Database Administrators].

PS: I fell upon `int_days - 693594` in some scripts as well. You need to check by examples which is true. Even though I have checked my examples, I cannot guarantee that I understood it right. I hope that `693594` is wrong, only one of the two, to subtract `693594` or `693596`, can be right, and I hope that `693596` is right as I wrote it. Yet, the `693594` comes from a professional setting.

Share Improve this answer edited Oct 9 at 11:32

Follow

answered Oct 9 at 10:51



questionto42

9,382 6 72 108