PROBLEMS OF DATE AND TIME DATA TYPES IN RELATIONAL MODEL OF DATA

Dimitrov V.

University of Sofia, Sofia, Bulgaria

E-mail: cht@fmi.uni-sofia.bg

Several years after the initial announcement of the relational model of data, Codd published a review on the model, so called Version 2. This review is based on the experience of relational database systems implementation in the intermediate period. One of the main corrections are recommendation on date and time data types. This paper reinvestigate the topic from the nowadays point of view.

Keywords: date, time, relational model of data.

1. Introduction

Every computer system is a hierarchy of virtual machines: hardware, BIOS, operating system, system software (programming languages, database systems, run-times), middleware, application. Every such a "virtual machine" uses its own system clock and date-time representation. Usually, system clocks and date-time representations are incompatible with each other that is a stable source for vulnerabilities. The most popular among these vulnerabilities is Year 2000 problem.

The system clock's time scale begins from some fixed time point called "epoch start" and ends in another time point – "epoch end". The time scale is discrete based on some fixed tick, like seconds, milliseconds etc. The time points are measured in the number of ticks since the epoch start.

For example, UNIX and Posix system's epoch begins at 1970-01-01T00:00:0Z, the tick is one second. Date-time is stored in 32-bits or 64-bits signed integer. This means that for 32-bits systems, epoch ends at 2038-01-19T03:14:07Z, or in other words it is Year 2038 problem.

Microsoft supports several epochs for their offerings, but the tick is 100 nanoseconds.

Datetime functions convert time points into calendar dates and times using information about leap seconds, time zones, daylight saving times etc.

In the initial representation of the relational model of data [1], Codd introduced the domains as sets of atomic data. Date and time data types are very important business data, and in the next version of the model, so called RMD-2 (Relational Model of Data – Version 2) [2], he defined four classes date and time data types (RT-4 – RT-7). There are 14 specific recommendation for dates listed in RT-4, and 12 for times in RT-7.

Initial implementations did not pay much attention on dates and times. Usually, they were local dates and times, based on the operating system clock or the programming language's system clock used for DBMS implementation. The next generations of DBMS's have more sophisticated dates and times with time zones etc., but continue to support compatibility with these early date and time data types.

ISO tries to put some order introducing standards for date and time formats for data interchange [3] and for SQL [4]. These standards fix the current state of the art introducing more features as "interchanging parts agreements" or "implementer's option".

The aim of this paper is to give a clear vision on date and time data types in the relational model of data without any compatibility considerations. For that purpose, Codd's recommendations are the base.

2. RT-4 Calendar Dates & RT-5 Clock Times & RT-6 Coupling of Dates with Times & RT-7 Time-zone Conversion

RT-4: From the user's standpoint, dates appear to be treated by the DBMS as if they were atomic values. However, the DBMS supports functions that are capable of treating as separate components the year, month, and day of the month.

RT-5: From the user's standpoint, clock times appear to be treated by the DBMS as if they were atomic values. The DBMS however, supports functions that are capable of treating as separate components the hours, minutes of the hour, and seconds of the minute. The services provided include counterparts to the first 12 of the 14 services listed in the discussion of RT-4.

RT-6: The DBMS supports a composite data type consisting of the data type DATE coupled with the data type TIME, allowing the functions applicable to dates alone or times alone to be applied to combinations in which DATE plays the role of the high-order part and TIME the low-order part.

RT-7: The DBMS supports (1) the conversion of every date-time pair from any specified time zone to Greenwich date and Greenwich mean time, and (2) the inverse conversion of Greenwich date-time pairs back into a specified time zone.

The services provided by Feature RT-4 include the 14 that follow:

1. Independence of date and time from particular time zones in which users are located, by use of Greenwich dates and Greenwich mean time

Codd recommends database system to store dates and times in the database only as UTC. This means that there is no need to store time zones for every date or time entry.

This recommendation, especially in the current situation of business globalization, is very reasonable – space for and processing of time zones would be saved. In the worst case, processing of time zones is nearly the same as conversion to the local time zone date and time.

In the context of this recommendation, the database system clock's epoch can be that of ISO/DIS 8601:2018 – proleptic Gregorian calendar with year zero (leap year), with negative dates before the epoch beginning.

For all other calendars, like Julian, Indian, Islamic, Jew's etc., suitable conversations can be implemented.

2. The function called NOW yields for any site the current date and time that are in effect in the time zone of the site.

The function NOW is very simple at first glance, but its implementation has many variations in the currently available database systems. The term "local time" may be the local time of the site from which the session has been initiated, but what if the user cross time zone during the session, for example if he/she is traveling in a plane, train or car?

Another interpretation of the term local time is the database site local time, but if the database is distributed on several time zones, or the DBMS run on a cloud spread on several time zones?

The general solution is simply to fix local time zone in some global parameters of the DBMS.

3. Extraction of any one or any pair of the three components, a form of truncation.

Component extraction from dates and times is reasonable to be done only for single component, but not for pairs, triples etc., because some kind of pairs have no reason, like the pair year and day of the month.

There are very many combinations for datetime, but only few of them are usable.

If the user can extract single component, he/she can combine single components in a way that he/she needs.

Implementation of component extraction must be at data manipulation language's level. At implementation level, this is usually an embedded feature in the programing language if the dates and times are stored as structures for year, month, day, hour, minute and second.

Another possibility is dates and times are stored as ticks passed from the beginning of the epoch. In that case sophisticated conversation from ticks to calendar dates and vice versa must be implemented.

Usually, database systems store dates and times as string of components.

4. Extraction with rounding of either year alone or year followed by month.

Extraction with rounding to some finer component is not very useful functionality.

For ISO/DIS 8601:2018 this recommendation can make sense especially for the date and time formats where the finer component is represented with decimal fraction. For example, the component day, if it is the finest component, may contain decimal point and fraction for hour, minute, and second.

5. Conversion of the combination year, month, day of the month to the year followed by day of the year, as well as conversion in the opposite direction.

There are many more formats of the kind year – day of the year, such like:

- year – hour of the year;
- year – minute of the year;
- year – second of the year,

but also:

- year, month – hour of the month;
- year, month – second of the month;
- year – day of the year – hour of the day;
- year – day of the year – second of the day;
- year – hour of the year, second.

In addition, ISO/DIS 8601:2018 has a format for dates: year – week of the year – day of the week. If this format of the date in combine with the time, many more formats would be generated.

It is clear that a few of these formats are useful, but many more are not. So, must be decided which formats to implement and which ones – not.

The best solution is if the data manipulation language is open to new data types. In that case, the user can create the date and time type he/she needs. Some basic functionality must be implemented like conversion of datetime to year – day of the year, hour, minute, second; to year – minute of the year, second; to year – second of the year etc.

If the date format year – week of the year – day of the week is supported, at least conversion to year, month, and day format must exists.

For conversions with the time, date component must be available, because there are no other way to have information about the leap seconds.

6. *Computation of the difference between two dates of similar or distinct external types, where each argument is expressed as*

   a. *years only, or*

   b. *years and months, or*

   c. *years, months, and days of the month, or*

   d. *years, and days of the year*.

These four options must be available to users, and the result must be of the same external type as the argument that is coarser.

The interval between two dates must be in nominal units, because the nominal units are practically used.

What exactly are nominal units? For example, nominal month lasts from a day in the current month until the beginning of the same day in the next month. If such a day in the next month does not exist then the end of the next month is used for such purpose. This rule applies to all other nominal units like year, day, hour and minute.

ISO/DIS 8601:2018 suggests this rule for nominal units, but it is not obligatory rule for the interchanging parties. Even more, the standard suppose that the difference between two date (datetimes or times) can be negative.

The idea behind the interval is that when to the earlier date the interval is added the result would be the second date. In that case, rounding and truncates to the result are not acceptable. The user can extract from the interval its nominal components.

7. *Conversion of date intervals into years only or months only or days only, using truncation or rounding as specified, if the conversion is from fine units to coarser units.*

The user can extract interval components and if he/she want to round them. This must not be a part of the DMBS implementation.

8. *Arithmetic on dates, including computation of a date from a given date plus or minus a date interval, without the adoption of dates and date intervals as distinct data types.*

Intervals are composed of nominal components. This means that it is possible day and month components in the interval to have zero value, but this is impossible for them in a date. Therefore, the dates cannot be intervals. That is why the database systems implement intervals as different data type from the dates.

Allowed arithmetic with dates (datetimes) and intervals is:

- difference of two dates (datetimes) gives an interval;
- to a date (datetime) an interval can be added and the result is another valid date;
- from a date (datetime) an interval can be subtracted and the result is another valid date.

Arithmetic with only times is not well defined.

9. *Pairwise comparison of dates, including testing of pairs of dates to see which is the more recent and which is the less recent.*

Comparison supposes that some ordering exists in the domain.

10. *Finding the most recent date of a collection.*

The key term, here, is the “collection”. In the context of relational model of data, collection of data must be a set of values from the same domain or base type. Here, base types are date, time and datetime. All other external dates and times are domains (in SQL distinct types) based on them.

11. *Finding the least recent date of a collection.*

Here, the above considerations are applied.

12. *All varieties of joins based on comparing dates.*

In RMD-2, joins require both columns to be from the same domain or base type. In this case, domains are based on the base types date, time and datetime. Therefore, both column must have as base type date, time or datetime.

13. *The ability to report dates in at least one of the following formats:*
    a. *European format: D,M,Y;*
    b. *North American format: M,D,Y;*
    c. *computer format: Y,M,D;*
    d. *in the Indian calendar with lunar months.*

Most of currently available database systems support this recommendation. The problem is with some exotic and rare calendars.

14. *Two types of date-conversion functions:*
    a. *DATE_IN for transforming dates from external representation of dates to the internal representation;*
    b. *DATE_OUT for transforming dates in the reverse direction, with the DBA having the option of putting into effect functions defined and specified by the DBA either for all*

users of a given DBMS or for specified classes of users (instead of or in addition to those supplied by the DBMS vendor).

This option is needed because users with different responsibilities and those located in different countries (even within a single country) may employ different kinds of dates externally with respect to the DBMS.

These two date conversion functions stay aside from all other considerations. The data manipulation language has reach set of data types for dates and times. The DBMS implements date, time and datetime. Conversion of external data types to internal ones is a problem of the language translator (interpreter) but not a user or implementation problem.

4. Conclusion

Recommendations listed here with the comments on them can be used as methodology for dates and times. Some of above-mentioned functionalities are implemented in the currently available database systems. Other functionalities can be implemented in SQL. There are no need to wait for a DBMS that totally supports date and time recommendations.

Even, it is possible to implement all above recommendations with different date and time representations from the DBMS native one.

Acknowledgements

This work is supported by the project ДН 02/9/17.12.2016 of the Bulgarian Science fund.

References

[1] Codd, E. F. (1970) A relational model of data for large shared data banks. Commun. ACM 13, 6 (June 1970), 377-387. DOI: https://doi.org/10.1145/362384.362685

[2] Codd, E. F. (1990) The Relational Model for Database Management: Version 2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] ISO/DIS 8601:2018 Date and time – Representations for information interchange.

[4] ISO/IEC 9075:2016 Information technology – Database languages – SQL.