Serdica J. Computing **12** (2018), No 4, 247–264

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

CODD'S RECOMMENDATIONS FOR DATE & TIME DATA TYPES AND THEIR IMPLEMENTATIONS IN ISO SQL, DB2, ORACLE, AND TRANSACT-SQL

Vladimir Dimitrov

Abstract. Date and time data types are very important for business. From the point of view of the relational model of data, these data types are simply atomic domains—their structure and operations are unimportant. After the initial introduction of the relational model of data and the next following implementations based on that model, Codd, in Version 2 of the relational model of data, corrected his point of view introducing 14 recommendations about date and time data types. This paper investigates implementations of date and time data types in ISO SQL, DB2, Oracle and Transact-SQL.

1. Introduction. Codd introduced the relational model of data in [1]. Twenty years after that, he published in [2] Version 2 of the relational model of data. This was an attempt to standardize the relational model of data taking in account the initial experience of its implementations. The commercially available at that time implementations very freely interpreted the relational model of data. Implementation of Version 2 was very difficult for

· *ACM Computing Classification System* (1998): H.2.1, H.2.2, H.2.3.
Key words: relational model of data, data types, date, time.

the database system vendors due to the backward compatibility. Many years after that Version 2 is still only partly adapted.

One of the corner stones of Version 2 are so called "extended data types". Codd does not clarify the type system for theses extended data types. The object-oriented approach influenced the concept of extended data types.

Codd talks about external (query language) and internal (implementation) data types. This means that Codd does not accept SQL as the only query language standard and opens a door for alternatives.

Codd recommended in [2] RT-4 – RT-7 (RT-4 Calendar Dates, RT-5 Clock Times, RT-6 Coupling of Dates with Times, and RT-7 Time-zone Conversion), date and time to be represented as extended data types in the relational model.

Following the date and time recommendations from [2], this paper analyzes implementations of ISO SQL [3], DB2 [4], Oracle [5] and Transact-SQL [6].

2. Codd's Recommendations.

"***RT-4 Calendar Dates****: 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.*"

The relational model of data uses the term "domain" that is a set of values. The type is a set of values plus operations on these values. Actually, a type without its operations is a domain and the relational model of data implementations use the types in that sense – as sets of values.

The date (time) data type is atomic. However, from practical point of view, this type plays special role and Codd describes it as an extended data type with its structure and operations.

The date type structure consists of year, month and day.

The operations on dates have 14 features:

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

This feature about internal dates and times means that dates and times must be stored in the database as GMT (Greenwich Mean Time) or precisely as UTC (Coordinated Universal Time) – no local dates and times. If someone needs of local dates and times, the DBMS should offer such a conversion.

At the time when Codd published these recommendations, almost all relational databases had stored only local dates and times. The problem with this recommendation is the backward compatibility with the older databases. Many years after that the situation is still the same. Some vendors support a particular time zone by default for the database. For the dates and times that are not from the default time zone, they offer date and time types with time zones. It is possible all databases by default to have UTC time zone and to satisfy this feature but the backward compatibility problem remains.

The situation is more complex when the DBMS supports dates and times for the system on which the system is running, for the user's session location, for the hardware system etc.

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

This function is external one – in the query language.

The users at their session site can use the function NOW. It returns the local date and time, but the DBMS must store returned value in UTC.

Other possible interpretations of the term "site" are the database system site, the database site, the data source site, the hardware system site etc. These interpretations are available in the current database system implementations expanding system complexity.

- "*4.3. Extraction of any one or any pair of the three components, a form of truncation.*"

However, the idea to extract any pair of the three components is not brilliant one. It is enough if the database system supports extraction of any single component.

The DBMS must support this feature at the external level. At least four functions for date have to be implemented and many more combinations for time and date-time.

For this feature, some vendors have implemented functions with formatting patterns. Using the pattern, the user can specify the needed combination.

- "*4.4. Extraction with rounding of either year alone or year followed by month.*"

This feature is on the external level. The DBMS must implement two functions. The first function should return a rounded year, i. e., if the date is before the middle of the year, the current year would be returned, else the next year.

The second function must return a rounded month and the year, i. e., if the day is before the middle of the month, the current month would be returned, else the next month.

The DBMS must implement just a same functionality for times and the other date-time combinations.

The value of this functionality is not obvious.

It is better if the query language offers to the users to define its own functions and if he/she needs of such a functionality like discussed one to define the corresponding functions. SQL has such a construction.

- "*4.5. 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.*"

The DBMS must implement two functions at the external level for this functionality: one to convert date to year and day of the year and the opposite one. This is not trivial functionality because of leap years. These functions have some important business value.

This feature in the case of time and date-time types is more complex because of leap seconds and the field combinations.

This functionality is not trivial one and must be faced by the vendors in way not like that caused "Year 2000 problem", but still there no indications for such complex solutions: vendors support only some date intervals, usually from the beginning of Gregorian calendar; no support of dates and times BC; no support of Julian calendar etc.

- "*4.6. 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."

Here, Codd clearly introduces two kinds of dates: external and internal ones. There are four external date types and only one internal date type.

One approach is to implement the external date types as internal ones and then to implement operations between these types, but such an implementation would be very huge.

Better approach is to have only one internal date type and to convert external dates to internal ones before the operations. The DBMS shall convert operation result to external dates.

There are 10 combinations between external date types for this feature; time and date-times add many more combinations. Some vendors solve the problem by offering cast operations at query language level, but usually they do not offer all possible combinations.

This feature is valuable in some combinations, the query language, like SQL, and the DBMS must implement and support it. User-defined functions can implement a part of this functionality.

- "*4.7. 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.*"

This feature uses the external date types – 10 possible combinations; there are two possible results: in years, in months and in dates, and there are two kind of operations – truncation and rounding, therefore 60 functions must be implemented.

The next problem is how to represent internally date intervals?

With this feature, the date type implementation is very huge, but if time and date-time types have to support this feature – the implementation is enormous.

For this functionality, vendors usually introduce special types like INTERVAL, DURATION etc.

- "*4.8. 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.*"

Sixty functions at external level must implement this feature and many more functions for time and date-time combinations.

Very important is the interpretation of the date intervals. At internal level, the best way is to represent date intervals as the number of the days between the two dates.

Vendors support arithmetic on dates and times using specialized interval types. Here, Codd is very wrong with the presumption "without the adoption of dates and date intervals as distinct data types" because dates and date intervals are distinct types: an interval has a fixed value of years, months, or days, but the date is a fixed point in the time scale.

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

This is an external level feature. The result of comparison is true or false (Boolean) or more recent / less recent value (MIN/MAX functions).

Twenty combinations are available at the external level (with first and second argument) returning 15 results, therefore 35 functions; and 70 functions for MIN and MAX functions. The combinations with time and date-time make this feature very huge.

May be the simplest approach would be only one external data-time type to be supported – at internal level this is the solution.

- "*4.10. Finding the most recent date of a collection.*"

Here, the problem is with the term "collection". Is it a collection of dates from the same external date type or a collection of dates represented in different external date types? One internal implementation or four different implementations by four kinds of results, therefore four implementations or 16 implementations. The time and date-time types introduce many more implementations.

The term "collection" must be interpreted as "set" not only because it simplifies the implementation but because it is the intuitive way of comparing.

At internal level, only a set of dates (date-times) is acceptable.

- "*4.11. Finding the least recent date of a collection.*"

The notes from 4.10 apply here.

- "*4.12. All varieties of joins based on comparing dates.*"

This feature must be available in the query language (SQL) and implemented in the DBMS.

- "*4.13. 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."

This feature is usually available in any DBMS implementation, but may be in the globalization context must be available for all these formats and some more? What about the daytime like Byzantine one etc.?

- "*4.14. Two types of date-conversion functions:*
  - *DATE_IN for transforming dates from external representation of dates to the internal representation;*
  - *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."

DATE_IN and DATE_OUT are functions that convert from external to internal formats and in opposite direction. Something here is not very clear: the functions convert between the external and internal representations; therefore, the internal representation of the date is available at the external level as an external date type.

These functions are available on the external level. The users use them by the query language translator. The users can define additional functions of that type.

An external date type must represent the internal date type because the users use only external types.

"***RT-5 Clock Times****: 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.*"

The notes about clock times are available in RT-4.

"***RT-6 Coupling of Dates with Times****: 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.*"

This composite date-time type can be the base for the internal representation of date and time, but not necessary.

The alternative way is to represent data and time separately and date-time type to be their composition.

"***RT-7 Time-zone Conversion****: 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.*"

This functionality is necessary for UTC representation at external and internal level.

Many vendors support this feature.

3. ISO Data Types. SQL is a standard query language. It is at the relational DBMS external level. There are no commercially available database system strictly following this standard.

Section "4.7 Datetimes and intervals" of ANSI/ISO/IEC International Standard, Database Language SQL [3] specifies date-time types. They are DATE, TIME WITHOUT TIME ZONE, TIMESTAMP WITHOUT TIME ZONE, TIME WITH TIME ZONE, and TIMESTAMP WITH TIME ZONE. DATE type is without time zone; TIME and TIMESTAMP types are with and without time zones.

There are three classes of datetime data types with the primary fields (structure): DATE (YEAR, MONTH, and DAY), TIME (HOUR, MINUTE, and SECOND), and TIMESTAMP (YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND). The standard ISO/IEC, ISO/DIS 8601-1 [7–8] contains more details about the fields.

The field YEAR ranges in 0–9999 (no dates BC), the field MONTH in 1–12, and the field DAY in 1–31.

The standard ISO/IEC, ISO/DIS 8601-1 [7] uses the proleptic Gregorian calendar with astronomical year numbering. This means year 0 exists and it is a leap year. The years before year 0 are negative ones (astronomically), but this is not part of the standard. Years in the range 0–1582 are under agreement of communicating parties (in the year 1582, the Gregorian calendar replaces the Julian calendar).

The most consistent is the astronomical year numbering with year 0 and negative years BC.

The field HOUR ranges in 0–23, the field MINUTE in 0–59, and the field SECOND in 0–60. There are "leap seconds" – positive or negative ones. In [6], the 60th second of the minute is clarified as "A positive leap second is inserted after [23:59:59Z] and is represented as [23:59:60Z]. The negative leap second is simply omission of [23:59:59Z]. Insertions or omissions take place as determined by IERS normally on 30 June or 31 December, but if necessary on 31 March or 30 September." The SQL standard [3] accepts that it is possible a minute to contain exactly 59, 61, or 62 seconds, but this is in contradiction with ISO standard [7], which is the leading standard in that case. The support of leap seconds is not obligatory by the standard.

It is possible a SECOND to contain fraction and in that case it ranges in 0–60.999... including leap seconds.

The datetimes types are with or without time zone. The time zone displacement ranges in the INTERVAL –'12:59' HOUR TO MINUTE and INTERVAL +'13:00' HOUR TO MINUTE.

The interval data types are INTERVAL (year-month interval or a day-time interval). There are no other kinds of intervals.

The year-month intervals contains YEAR and MONTH fields, but both are not obligatory. The meaning of these fields is the number of years and months. The YEAR is an integer – unconstrained. The MONTH ranges in 0–11 – months in the year, i. e., the number of full months from the beginning of the year.

The day-time intervals contains fields DAY, HOUR, MINUTE, and SECOND. The meaning of these fields is the number of days, hours, minutes, and seconds. It is possible for seconds to contain fractions.

Any contiguous subset of these fields can comprise a value. For example, DAY-HOUR, HOUR-MINUTE, MINUTE-SECOND, but not DAY-MINUTE, DAY-SECOND, or HOUR-SECOND.

The field DAY is unconstrained integer, the field HOUR ranges in 0–23, the filed MINUTE – in 0–59, and the field SECOND – in 0–59.999... Leap seconds are not included here – more appropriate is the field SECOND to range in 0–60.999...

Comments on dates and times features by their numbers:

1. The introduction in SQL of datetime types with time zones is wrong from Codd' point of view. The idea of RT-7 is that the database system should store dates and times in UTC and should support datetime conversion between time zones.

   The commercially available database systems have adapted datetime types with and without time zones.

2. The functions "CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP respectively return the current date, current time, and current timestamp; the time and timestamp values are returned with time zone displacement equal to the current time zone displacement of the SQL-session." [3]. In other words, the SQL query language supports the function NOW.

3. First, the DATE type. The extraction of YEAR-DAY combination is non-sense. The other possible extractions are YEAR-MONTH, MONTH-DAY, YEAR, MONTH, and DAY.

   Second, the TIME type. The reasonable extraction combinations are HOUR-MINUTE, MINUTE-SECOND, HOUR, MINUTE, and SECOND. The combination HOUR-SECOND is non-sense.

   Third, the TIMESTAMP type. It is a combination of DATE and TIME types.

   These operations are supported by ‹extract expression› that can extract any combination of fields from the datetimes.

4. For the TIME type, that service sounds like "extraction with rounding either to hour alone or hour followed by minute".

   Codd has not defined rounding for TIMESTAMP but rounding can start from year, month, day, hour, and minute, followed by the next successive fields.

   There are no standard functions of that type but the user can define them. SQL does not support this feature.

5. For the TIME type: "Conversion of the combination hour, minute, second of the minute to the hour followed by second of the hour, as well as conversion in the opposite direction".

For the TIMESTAMP type more combinations are possible but with rare usage.

There are no such functions in SQL.

6. Actually, Codd defines four new "external" types: YEAR, YEAR-MONTH, and YEAR-DAY_OF_THE_YEAR.

   The TIME counterparts are for these new "external" types are HOUR, HOUR-MINUTES, HOUR-SECOND_OF_THE_HOUR.

   The TIMESTAMP counterparts "external" types are YEAR-HOUR, YEAR-MINUTE, YEAR-SECOND, MONTH-HOUR, MONTH-MINUTE, MONTH-SECOND, DAY-HOUR, DAY-MINUTE, and DAY-SECOND.

   There are no such types in the SQL standard. Users can define them using the type system and this approach is better.

7. The SQL standard defines rounding as follows "An approximation obtained by rounding of a datetime or interval value D for a ‹datetime type› or ‹interval type› T is a value V in T such that the absolute value of the difference between D and the numeric value of V is not greater than half the absolute value of the difference between two successive datetime or interval values in T. If there is more than one such value V, then it is implementation-defined which one is taken."

   This functionality is not included in the SQL standard.

8. It is clear that dates and date intervals are distinct data types.

   Arithmetic on dates are operations +, −, *, and /. Codd is not precise on the topic.

Table 1. Valid operations on datetimes and intervals in the SQL standard

Operand 1	**Operator**	**Operand 2**	**Result Type**
Datetime	+	Datetime	Interval
Datetime	+ or −	Interval	Datetime
Interval	+	Datetime	Datetime
Interval	+ or −	Interval	Interval
Interval	* or /	Numeric	Interval
Numeric	*	Interval	Interval

The intervals are well defined and they are really an implementation of feature 8.

9. The SQL standard defines data and datetime conversions between the various datetime data types. First, the user have to convert datetimes to the most precise type and then to do comparison.

   In SQL, the year-month (day-time) intervals are mutually comparable only with year-month (day-time) intervals.

10. The term "collection" is not clear in that context. If the collection is a set of datetimes (selected for example from a table column) then SQL supports that service, otherwise – not.

11. The same reasoning as above in feature 11.

12. The SQL standard supports this feature for all standard joins.

    The idea to store datetimes in different internal representations is catastrophic. The joins on columns with datetimes of different formats because their values must be converted before the join comparisons. If the database stores only UTC datetimes this conversion problem would not exist.

13. SQL supports several calendar representations at different levels and with many options.

14. SQL does not clarify what the internal representation of datetimes is.

4. DB2 Data Types. DB2 for LUW (Linux, UNIX, and Windows) is in focus here.

DB2 datetime data types are DATE, TIME, and TIMESTAMP. They follow ISO SQL standard and there are additionally types WITHOUT TIMEZONE.

The type TIMESTAMP WITH TIMEZONE is available for Z/OS versions of DB2. There are no TIME WITH TIMEZONE in DB2 for LUW.

There are no the INTERVAL type as in the ISO standard but types for time durations of different kinds are available.

Comments on date and time features:

1. The datetime types either are local ones or as in DB2 Z/OS versions with/without time zone (for TIMESTAMP type).

2. There are several special registers, which support that service: CURRENT TIME, CURRENT TIMESTAMP and CURRENT TIMEZONE. There is only one local time zone – C runtime functions determines its value!

3. The EXTRACT, TRUNCATE (TRUNC), YEAR, MONTH, DAY, DATE_TRUNC, DATE_PART, TIME, HOUR, MINUTE, SECOND,

MICROSECOND, and TRUNC_TIMESTAMP functions with a format-string support truncation.

4. The functions ROUND, and ROUND_TIMESTAMP using format-string support rounding.
5. There are no functions converting date to year and day of the year, but the user can implement this functionality, with the DAYS function and some other build-in functions.
6. There are no external types like YEAR, YEAR-MONTH etc. Therefore, the difference between dates of these types is not available, but the function TIMESTAMPDIFF is the base for implementation of this feature with the above-mentioned extended data types.
7. Conversion of date intervals into years, years-months etc. is not available but can be implemented as user data types and functions.

   The statement CAST may play the leading role for that feature.
8. Arithmetic on dates, times and timestamps arithmetic with durations is available. It is built-in but partly implemented.
9. Datetimes comparison is built-in and available for all supported types.
10. Functions MAX and GREATEST are applicable only to the supported types.
11. Functions MIN and LEAST implement this service.
12. All supported by DB2 kind of joins can use datetimes columns, but with data casting.
13. The functions TO_CHAR and TO_NCHAR support several report formats for dates and times.
14. The functions TO_DATE and TO_TIMESTAMP convert to and from date and date-time, but there are no internal format for date and time.

5. Oracle Data Types. Oracle has full support of ISO datetime and interval types: DATE, TIME WITHOUT TIME ZONE, TIMESTAMP WITHOUT TIME ZONE, TIME WITH TIME ZONE, TIMESTAMP WITH TIME ZONE, INTERVAL YEAR TO MONTH, and INTERVAL DAY TO SECOND. Therefore, all comments for ISO SQL are applicable here.

Comments on dates and times features:

1. Independence of date and time from particular time zones, in the sense of UTC only zone for the database is not supported [5]:
   - “Data is normalized to the database time zone when it is stored in the database.

- o When the data is retrieved, users see the data in the session time zone."

This solution is something like Code's recommendation, but not exactly.

2. The functions CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP supports the concept of function NOW.
3. The functions EXTRACT and TRUNC support truncation on the ISO datetime types.
4. The function ROUND supports rounding of dates. There are no special rounding function for times, but the user can implement such functions using the Oracle type system.
5. There are no functions to convert dates to year and day of the year etc., but the user write them in PL/SQL.
6. Arithmetic with date-times (DATE, TIMESTAMP, INTERVAL and NUMBER) is very simplified.
7. Conversion of date intervals into years only, months only, days only, using truncation, rounding, etc., is not directly supported, but there are enough functions that can be used to achieve this functionality.
8. Oracle does not support arithmetic on dates without the adoption of dates and date intervals as distinct data types: date types are distinct and the system regulates the arithmetic with them in very simplified manner.
9. The comparison uses many implicit and explicit conversions.
10. In Oracle, the term collection is associated with set of same type elements and in that sense, the DBMS supports finding the most recent element is but it is not true in the broader sense.
11. Just a same as above for the least recent date.
12. Oracle implements all varieties of joins with date-time types.
13. Oracle supports many calendar formats including the recommended ones.
14. Oracle does not define conversion to internal and external datetimes.

6. MS SQL Server Data Types. Transact-SQL supports date-time types: time, date (0001-01-01 through 9999-12-31), smalldatetime (1900-01-01 through 2079-06-06), datetime (1753-01-01 through 9999-12-31), datetime2 (0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999), and datetimeoffset (0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 in UTC).

The operating system, on which the instance of SQL Server is running, defines the dates and the times.

Comments on dates and times features:

1. Only the type datetimeoffset supports time zones.
2. The functions CURRENT_TIMESTAMP, GETDATE, GETUTCDATE, SWITCHOFFSET, SYSDATETIME, SYSDATETIMEOFFSET, and SYSUTCDATETIME support the function NOW.
3. Only for the dates, the functions DATEPART, DAY, MONTH, and YEAR support the truncation.
4. MS SQL Server does not support extraction with rounding of datetimes.
5. MS SQL Server does not support conversion of the combination year, month, and day of the month to the year followed by day of the year.
6. The functions DATEDIFF and DATEDIFF_BIG partly support computation of the difference between two dates.
7. MS SQL Server does not support conversion of date intervals into years only or months only or days only, using truncation or rounding.
8. The functions DATEADD, DATEDIFF, and DATEDIFF_BIG partly support arithmetic on dates.
9. MS SQL Server does not support directly pairwise comparison of any dates and times.
10. MS SQL Server partly supports finding the most recent date of a collection with the function MAX.
11. MS SQL Server partly supports finding the least recent date of a collection with the function MIN.
12. All varieties of joins based on comparing dates the DBMS does not directly support.
13. The DBMS supports several calendars including some of the recommended ones.
14. There are no definition of internal and external datetimes.

7. Conclusion.

Table 2 summarizes all findings.

Table 2. Summary of findings

Recommendations / Features	**ISO SQL**	**DB2**	**Oracle**	**Transact-SQL**
RT-4 Calendar Dates: 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 Clock Times**: 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.				
4.1. Independence of date and time from particular time zones in which users are located, by use of Greenwich dates and Greenwich mean time.	no	no	no	no
4.2. The function called NOW yields for any site the current date and time that are in effect in the time zone of the site.	yes	no	yes	yes
4.3. Extraction of any one or any pair of the three components, a form of *truncation.*	yes	yes	yes	no
4.4. Extraction with rounding of either year alone or year followed by month.	no	yes	yes	no
4.5. 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.	no	no	no	no
4.6. 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	no	no	no	no

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 <u>the result must be of the same external type as the argument that is coarser</u>.				
4.7. 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.	no	no	no	no
4.8. 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.	no	no	no	no
4.9. Pairwise comparison of dates, including testing of pairs of dates to see which is the more recent and which is the less recent.	no	no	no	no
4.10. Finding the most recent date of a collection.	no	no	no	no
4.11. Finding the least recent date of a collection.	no	no	no	no
4.12. All varieties of joins based on comparing dates.	no	no	no	no

4.13. 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.
	yes	yes	yes	yes

4.14. 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.	no	no	no	no

RT-6 Coupling of Dates with Times: 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.	yes	yes	yes	yes
RT-7 Time-zone Conversion: 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.	yes	no	yes	no

Oracle and DB2 are comparable in many ways, but Oracle supports datetimes better.

Nevertheless, ISO SQL and all its implementations do not support Codd's recommendations.

The available implementations of the SQL dictated some of the ISO SQL decisions in the name of compatibility.

Some of Codd's recommendations are very old fashioned and generate very many conversion rules and functions that are practically not usable for implementation.

How to change that situation with datetimes? First, Codd's recommendations have to be updated in more implementation-oriented way. Second, for currently available DBMSs, a portable library with these new recommendations can be defined.

ISO SQL future versions will support compatibility with the older versions and the vendors will implement ISO SQL in the same way – without full ISO SQL support. This means that only portable libraries on portable data types can be implemented. The extended data types must be clarified in the terms of object-oriented approach to go further on the relational model of data after Version 2.

REFERENCES

[1] Codd E. F. A relational model of data for large shared data banks. *Communications of the ACM,* **13** (1970), No 6, 377.

[2] Codd E. F. The Relational Model for Database Management: Version 2. Addison-Wesley Publ. Co, 1990.

[3] ANSI/ISO/IEC International Standard (IS), Database Language SQL—Part 2: Foundation (SQL/Foundation), September 1999.

[4] IBM DB2 11.1 for Linux, UNIX and Windows Knowledge Center. `https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.welcome.doc/doc/welcome.html`, 21 November 2017.

[5] Oracle Database Online Documentation 12c Release 1 (12.1). `https://docs.oracle.com/database/121/index.htm`, 21 November 2017.

[6] Microsoft, Data types (Transact-SQL). `https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql`, 21 November 2017.

[7] ISO/IEC, ISO/DIS 8601-1, Data elements and interchange formats – Information interchange – Representation of dates and times – Part 1: Basic rules, 2016.

[8] ISO/IEC, ISO/DIS 8601-1, Data elements and interchange formats – Information interchange – Representation of dates and times – Part 2: Extensions, 2016.

Vladimir Dimitrov
Faculty of Mathematics and Informatics
St Kliment Ohridski Sofia University
5, James Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: `cht@fmi.uni-sofia.bg`

Received February 5, 2018
Final Accepted May 15, 2018