International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 09 | Sep -2017 www.irjet.net p-ISSN: 2395-0072

SQL and Temporal Database Research: Unified Review and Future Directions

Rose-Mary Owusuaa Mensah[1], Vincent Amankona[2]

[1]*Postgraduate Student, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana*
[2] *Postgraduate Student, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana*

Abstract - *Several attempts to incorporate temporal extensions into the Structured Query Language, SQL, one of the most popular query languages for databases date back to the nineteenth and twentieth century. Although a lot of work and research has been done on temporal databases and SQL, there exist very limited literature clearly outlining the various events which have taken place with regards to temporal extensions of SQL over the years till the present state in a concise document. Consequently, researchers need to gather several pieces of literature before they can obtain a vivid pictorial timeline of the history and the current state of these temporal extensions for research and software development purposes. In order to align current research with that from the past, there is the need for a consolidation of information. In this paper, we seek to present the past and current state of temporal support in relational databases and also describe the various attempts and proposals to introduce temporal extensions into SQL over the years, discuss their level of success and impact on the most recent SQL standard. We also identify desirable temporal features which have not yet been introduced into any of the SQL standards and also determine the level of acceptance of the available SQL temporal extensions by some commercial database management system vendors.*

Key Words: SQL, temporal, databases, SQL:2011, DBMS, TSQL2,SQL-92

1. INTRODUCTION

Different users and applications have diverse requirements and as such wish to access data at different points in time; this requires a form of database management system which is able to keep track of past, current and future states of data. Many computer applications, such as banking, inventory control, law, medical records and airline reservations deal with time varying information: hence it is important that database management systems, DBMSs, are designed to be able to handle the temporal information. A database which provides in-built support for time is referred to as a temporal database. Researchers have engaged in the study of temporal concepts in databases over the years and have written various scientific papers, articles and books in this area.

In the early 1990s, a renowned researcher in the database community, Richard Snodgrass, proposed that temporal extensions to SQL be developed because few of them existed. In response to his proposal [1], a committee was formed to consolidate past research and suggestions from the research community in order to design temporal extensions to the 1992 edition of the SQL standard. Those extensions, known as TSQL2, were then developed by this committee and in 1993, they presented proposals to the ANSI SQL Technical Committee. Based on responses to the proposals, changes were made to the Language, and the definitive version of the TSQL2 Language Specification was published as a technical report in September, 1994 [2].

Almost five years later, a new standard SQL:1999 [3] was to be released and attempts were made to include a substandard, SQL/Temporal [4], which comprised parts of the TSQL2 specification. SQL/Temporal failed to make it to the SQL:1999 standard because the TSQL2 approach was heavily criticized. After the unsuccessful attempt, researchers continued to propose features to extend the SQL syntax for temporal support. A decade later, a new standard SQL:2011[5] has been developed, which provides a substantive support for temporal data management by introducing system-versioned tables, application-time period tables and bitemporal tables.

Although a lot of work and research has been done on temporal databases, there exists very limited literature which clearly outlines the various events which have taken place with regards to temporal extensions of SQL over the years till the present state in a single document. Researchers need to gather several pieces of literature before they can obtain a vivid pictorial timeline of the history and the current state of these temporal extensions, for research and software development purposes. It is for this reason that this paper seeks to provide a state-of-the-art survey on the temporal extensions of SQL, identify any drawbacks of the current extensions and present further research being conducted in this area. The survey conducted addresses the questions below:

- Which temporal features are available in the various SQL standards?
- What are the key activities and research which have been undertaken in the field of temporal

databases and their consequent impact on the most recent temporal extensions of SQL?

- How are the various database vendors adopting temporal extensions of SQL to their database management system products?

- Are there any shortcomings of the most recent SQL standard, SQL: 2011 and which further research is being conducted to solve these problems?

By gathering and analyzing various literature, other sources of information and extensive research, we attempt to understand the rationale behind the temporal concepts in SQL over the years.

2. TEMPORAL DATABASES

Conventional databases represent the state of information at a single point in time; new data can be added and modifications to existing data can be easily performed. However, a drawback of these conventional databases was the fact that old values were usually lost whenever changes occurred and as such it was impossible to track the change of information over time in the database. Also, in the early years, DBMSs provided very limited support for handling attributes involving time; Application programs had to implement functionalities for this purpose. Keeping track of data over time was therefore a major problem which needed to be addressed in databases.

Although the relational data model had gained popularity over the years, it provided very little support for addressing the temporal dimension of data. The need to handle time more comprehensively arose in the early 1970s especially in the area of medical information systems, where a patient's medical history is particularly important [6]. Other application areas which handle time related information in databases include banking, inventory management and airline reservation systems. Today, handling data that evolves over time can be effectively done using temporal databases. A temporal DBMS therefore has built-in support for time-varying data.

In temporal databases, timestamps can be attached to data or information. To establish the lifetime of a piece of information, we can represent its start time and end time in a database. The three notions of time which are relevant to temporal databases are user-defined time, valid time and transaction time. Valid and transaction times were coined by a very active researcher in the database community called Richard Snodgrass, together with his doctoral student, Ahn [7]. Valid time refers to when a fact is true in the real world. Transaction time is the time a fact is stored or modified in a database. User-defined time is used to include temporal information which is not handled by transaction time or valid time; and values of user-defined time are not interpreted by a DBMS.

As an example, consider Mr. X who registers for his health insurance on 4th June, 1995. However, the health insurance company could not record this information in their computer system because of technical challenges. On 8th July, 1995, the company is able to record Mr. X's information in their database. In this case, 4th June, 1995 is the valid time start event of his insurance and 8th July, 1995 becomes the transaction time. The need for temporal support in databases keeps increasing over the years: From the years 1982 to 1986, about 25 groups were also studying time in databases and 60 articles on temporal databases had been released [8]. It therefore became necessary to develop a common consensus glossary on temporal database concepts: this goal was successfully achieved [9].

Many years now, numerous temporal extensions have been proposed and therefore there is the need to further consolidate information. Unlike temporal databases, conventional databases model the dynamic real world as a snapshot at a particular point in time, and it may not reflect the status of the information in the real world. Therefore, as soon as an update is performed on such a database, the old information is lost and answering queries about past states became a major problem. These databases were referred to as snapshot databases [8]. To make such databases up-to-date, DBMSs provide various mechanisms. One of those was to store all the past states of the snapshot database and index them by the times they change (their transaction times).

Although this method was a step forward, we could only keep track of the transaction time history of the database without any support for history of the real world. Another approach to store past information is by using historical databases. These databases record a single historical state per relation, thereby taking into account the valid time [8]. In 1987, when Snodgrass introduced a new temporal relational query language, TQuel [8], he also proposed that the approaches to historical and rollback databases can be adopted in temporal databases which require support for both valid time and transaction time.

The effort of the database community to provide support for temporal data has really paid off. Today most DBMS vendors provide temporal support in their database products. Temporal databases can now be easily created to provide support for valid time, transaction time or both. The widely used query language for relational databases, SQL, has also undergone major changes to provide improved support for temporal data.

3. STRUCTURED QUERY LANGUAGE (SQL)

Structured Query Language (SQL) is the standard language used to retrieve information in relational databases. It was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s [6]. This initial version was referred to as Structured English Query Language, SEQUEL, and was designed purposely to manipulate data in IBM's first relational DBMS prototypes referred to as System R [6]. Subsequently, SQL was later introduced as a commercial database system in 1979 by Oracle Corporation.

In 1986, "Database Language SQL'' was formally adopted by the ANSI and ISO standards groups [10]. The official name of SQL as specified by the ISO is ISO/IET 9075 Standard:"Information Technology- Database Languages-SQL". According to Donald Chamberlin [6], making SQL a standard provided a mechanism for controlled evolution of the language and also created a forum for users and implementers to share ideas on the Language. From that time till today, the SQL standard has undergone major revisions and new features have been introduced (including e.g., outer joins, table expressions, recursion, triggered actions, user-defined types and functions, and online analytic processing (OLAP) functions). One of the most important features of SQL is its improved support for temporal data over the years. New versions of the SQL standard were released in 1989, 1992, 1999, 2003, 2006, 2008 and 2011.

4. TEMPORAL FEATURES OF SQL STANDARDS OVER THE YEARS

The SQL standard has undergone eight main revisions and the latest version was released in 2011. Each standard came with its own set of features by correcting problems in a previous standard and introducing entirely new features. However, not all the eight revised versions provided temporal support for SQL. The SQL standard has been divided into multiple parts to enable the relevant pieces to progress at different rates. Parts have been rejoined and split over the years based on extensive research and recommendations from the database community. Figure 1 depicts the timeline of events with respect to incorporating temporal extensions into SQL.

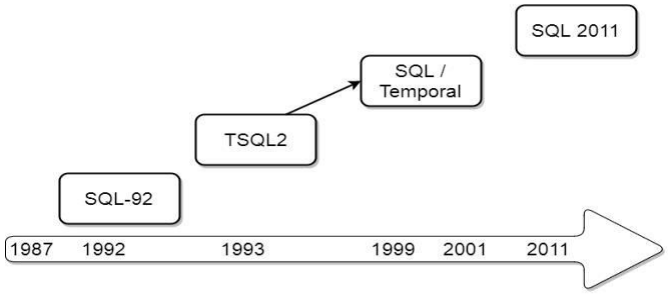

Figure 1: Timeline of history of temporal extensions of SQL

4.1SQL-92

The first major revision on the standard since its introduction occurred in 1992 and it was called SQL2 It is also known by names such as SQL/92, SQL-92, ISO/IEC 9075:1992, ANSI X3.135-1992, or FIPS 127-2 [11]. SQL-92 introduced significant features including some initial support for temporal data. Four data types were introduced, three of which were used in order to represent a point in time on the time axis referred to as an instant, and the other used to represent duration of time. These three data types were: DATE (a particular day, with a year in the range ad 1–9999), TIME (a particular second within a range of 24 hours), and TIMESTAMP (a particular fraction of a second, defaulting to microsecond, of a particular day) [12]. A DATE field consists of 10 characters which are used to denote day, month and year. DATE literals are specified in SQL by the keyword DATE, followed by a string of the form YYYY: MM: DD. In SQL, TIME is represented using the Universal Coordinated Time. TIME values are represented by the keyword TIME, followed by specifying the hour, minutes and seconds in the form, hh:mm:ss. TIME fields are usually 8 characters long but it is possible to specify precisions higher than a second: In this case, a dot is introduced after the default 8 character string and the number of digits after the dot must be placed in parenthesis after the keyword TIME.

TIMESTAMP data type contains year, month, day, hour, minute, and second values represented in the form YYYY-MM-DD hh:mm:ss. Just as in TIME data type, it is possible to specify higher precisions by introducing fractions of a second. The fourth data type introduced was the INTERVAL. A time interval is a directed duration of time with a known length, but not specific starting or ending instants. If we wish to specify the duration between two instant time points, the term PERIOD is used. By the definitions of both INTERVAL and PERIOD, it is clear that they represent entirely two different things. SQL-92 supports two kinds of intervals, month-year and second-day intervals [12]. There are three variants of month-year intervals namely INTERVAL <year literal> YEAR, INTERVAL <month literal> MONTH and INTERVAL <year-to-month literal> YEAR TO MONTH [13]. Second-day intervals can also be written in different forms such as

e.g., INTERVAL <day literal> DAY and INTERVAL <day-to-second literal> DAY TO SECOND. More detailed information about the description and use of intervals can be obtained from [13].

4.2TSQL2

Although SQL-92 introduced some form of temporal support, this was still not enough with respect to managing time in databases. The database community continued to work towards improving the old and developing new mechanisms for temporal support in databases. In 1994 [2], a temporal extension to SQL2 called TSQL2 was developed. TSQL2 is based on and highly compatible with SQL-92; legacy applications in the latter can easily be supported without any statement modifications or introduction of additional codes.

The language specification inherits the DATE, TIME, TIMESTAMP and INTERVAL data types from SQL-92. A new temporal data type PERIOD was also introduced, making it easier to specify range and precision. Other results from the TSQL2 project included the introduction of the three timelines: user-defined, valid and transaction times, temporal aggregates and surrogate data. In SQL-92, only snapshot tables existed; TSQL2 provided more support for introducing temporal elements into tables enabling users to include different timestamps into snapshot tables, event tables, valid time tables, transaction time tables and bitemporal tables. Valid time and transaction time which represent the time a fact occurred in the real world and the time a fact was stored in a database respectively. The document containing the TSQL2 Language specification can be obtained from [2]. This document contains about 71 pages of information on the concepts, syntax and implementation of TSQL2.

4.3SQL:1999 and SQL/Temporal

The fourth revision of the SQL standard occurred in 1999; It was known as SQL3 or SQL:1999 [14]. An interesting change occurred in the naming convention of this standard. The hyphen which was used in the previous standards such as SQL-89, SQL-92 was replaced with a colon in SQL:1999. This change was needed for two main reasons. The first was to ensure consistency with the names of other ISO standards [15]. Secondly, unlike the previous years where shortened form of the years, for e.g '86 and '92 were used, 1999 was maintained in full when the standard was named. This decision was taken because of the year 2000 computer problem, Y2K [16] and the impact of dropping the century indicator from dates [17]. SQL:1999 consisted of eight different parts which were published in two groups [17]. Though SQL:1999 was a significant improvement on the previous SQL standards, there were no new features introduced to improve handling temporal data.

Due to the fact that SQL:1999 provided no new support for temporal data, the database community made attempts to introduce some parts of the TSQL2 into the SQL:1999 standard. This new part (part 7) was called SQL/Temporal and it was to be included as a new substandard of SQL:1999. However, this attempt was unsuccessful because the TSQL2 approach was heavily criticized by Chris Date and Hugh Darwen [18]. In their words, TSQL2 contained major "flaws" and one of the obvious ones was related to "hidden attributes" [18]. Consequently SQL/Temporal was withdrawn and the ISO project for temporal support was eventually cancelled in 2001 because people were no longer willing to dedicate more efforts to the topic [18].

The fifth revision to the SQL standard occurred in 2003 [19]. SQL:2003 made revisions to all parts of SQL:1999 and added a brand new part, Part 14: SQL/XML, XML-Related Specifications. New data types; BIGINT, MULTISET, and XML were also introduced. [19] gives a good overview of the features that were introduced in this standard. There was no temporal extension to SQL in this Standard. SQL:2006 was introduced after SQL:2003 and the major change that it contained was revision to Part 14, SQL/XML, of the SQL:2003 standard. In July, 2008, SQL:2008 was introduced but also did not offer any new form of support for temporal databases.

4.4SQL:2011

About three years after SQL:2008, a new standard, SQL:2011 was released [5]. This standard was to some extent based on some concepts from SQL/Temporal and provided extensive support for temporal data. Out of the nine parts which existed in SQL:2008, five of them were modified. The remaining four remained in effect. SQL:2011 provides substantive support for handling temporal information in databases. The new temporal features are now part of SQL:2011 Part 2, SQL/Foundation [20], instead of appearing as a new part as was in the case of SQL/Temporal. New terminologies such as system-versioned tables, application time period tables and system-versioned application time period tables have been introduced for the old concepts which existed in TSQL2 namely transaction time tables, valid time tables and bitemporal tables respectively.

SQL:2011 is the most recent of the various SQL standards. After the last major project, SQL/Temporal, which was to add temporal extensions to SQL was cancelled in 2001, it took about 10 years before new temporal extensions were added to SQL. In between the SQL/Temporal and SQL:2011, the SQL standard underwent three major revisions namely SQL:2003, SQL:2006 and SQL:2008, but the revisions to the standard did not include new features for handling temporal data aside the pre-existing datatypes DATE, TIME, TIMESTAMP and INTERVAL. Consequently, it was challenging for users

to represent the space of time between two instants which is denoted as period because no datatype existed in SQL which could serve that purpose. Users had to "define" periods in their databases by creating two table columns, for the start time and the end time. Many applications had to implement basic temporal operations like time travel by introducing extra codes in their software applications: this often led to development overhead, faulty semantics and sub-optimal performance [21].

In December 2011, SQL:2011 [22] was formally adopted, replacing SQL:2008 as the most recent version of the standard. This new standard is also referred to as ISO/IEC 9075:2011 ("Information technology – Database languages – SQL") and the official document contains more than a thousand pages. One of the main new features of this new standard is improved support for temporal databases [23]. A new set of language extensions for temporal data support are now part of SQL:2011 Part 2, SQL/Foundation instead of appearing as a new part [24]. Almost four years after the introduction of this standard, the amount of literature covering its temporal aspects is quite limited. Several informative sources gathered during our research include but are not limited to the following [23] [24] [25].

Temporal concepts in SQL:2011 are largely based on concepts and constructs from SQL/Temporal, but there are significant differences [23]. The PERIOD datatype which was proposed in TSQL2 and SQL/Temporal did not make it into SQL:2011. However, SQL:2011 provides a means of specifying distance between two instants, which is called a period. Perhaps among the reasons why there is no period data type in SQL till today is because of the costs involved: For e.g, if a period datatype was added to SQL, then it would have to also be added to the stored procedure language, to all database APIs such as JDBC, ODBC, and .NET, as well as to the surrounding technologies such as ETL products, replication solutions, and others [22].

As a means of representing periods, SQL:2011 adds *period definitions* as *metadata* to tables: A period definition is a named table component which identifies a pair of columns that capture the period start and the period end time. At any point in time, these period definitions can be created or removed using enhanced CREATE and ALTER table statements present in this new standard [23]. SQL: 2011 represents time periods using the *closed-open* model and ensures that the start time is always less than the end time. This constraint is enforced as soon as the keyword *PERIOD* is declared in a table definition.

SQL:2011 introduces new terminology for some pre-existing time dimensions from TSQL2 and SQL/Temporal: Valid time tables are referred to as *application-time period tables* and transaction time tables are now called *system-versioned tables* [23]. Bitemporal tables can also be represented in this new standard and are referred to as system-versioned application-time period tables. Whereas new syntactic expressions have been introduced for queries in system-versioned tables and updates in application-time period tables, it is the opposite case for updates in system-versioned tables and queries in application-time period tables. Another important feature which has been introduced is the possibility to specify primary key and unique constraints to ensure that no two rows with the same key value have overlapping periods.

New period predicates have also been introduced in SQL:2011 which help to formulate queries for conditions involving periods. These predicates are functionally similar to Allen operators but produce contrasting results in many cases. Period predicates existed as far back as in SQL:92, which had the OVERLAPS predicate. Both TSQL2 and SQL/Temporal also introduced the CONTAINS, OVERLAPS, MEETS, PRECEDES and SUCCEEDS operators. The above mentioned predicates have made their way into SQL:2011, along with other new period predicates. Notable among the new SQL period predicates are the IMMEDIATELY PRECEDES and IMMEDIATELY SUCCEEDS predicates. Considerable amount of success has been achieved in the area of temporal support for databases but a lot still needs to be done. Although the SQL standard has undergone major revisions since it was established in 1989, only a few versions provide some support for temporal data. The need to provide temporal support in database systems became a major concern after SQL-92 was released. Since then, the most recognized projects and efforts towards temporal support in databases occurred during TSQL2, SQL/Temporal and finally in the most current standard SQL:2011.

5.DBMSs' SUPPORT FOR TEMPORAL DATA

In order to store, manipulate and query time related data in databases, the underlying database management system must provide built-in support for temporal extensions. Over the years, quite a number of attempts have been made by the database community, including DBMS vendors and SQL standard committees, to develop temporal extensions to SQL. However, not much information is available about the research and efforts of the above mentioned groups between the years 2002 and 2011 after the SQL/Temporal project was cancelled. Nonetheless, some indications make it obvious that efforts were actually made; For e.g., DBMS vendors began adopting temporal features in their DBMS products, and also concepts proposed by researchers have been incorporated in the new standard, SQL:2011, to provide improved temporal support. The historical and current state of temporal support in DBMSs and the contribution of some major DBMS vendors towards the current state of temporal support in databases cannot be overemphasized.

One of the legacy systems which provided some form of temporal support was the Oracle 9i, a database management system developed by the Oracle Corporation in 2002. This database system had the Flashback [26] feature which permits users to query data as it existed in a previous state. Oracle 9i has a special utility package, DBMS_Flashback to handle flashback queries. To activate the flashback feature, users first have to reconfigure their databases to grant the appropriate privileges. After which the user has to decide on one of two approaches which can be used to implement flashback queries [27]. The first approach is a time based approach, where users have to explicitly specify time values from past time. The other approach is to use a SYSTEM CHANGE NUMBER to identify the point to go back to [27]. Each of these approaches employs the "AS OF" clause. For example, the conventional SQL SELECT statement is modified as: Select * from employee AS OF TIMESTAMP<specify timestamp> or Select * from employee AS OF SCN <include scn number here> [28]. Oracle has maintained this query feature over the years and its DBMS system, Oracle 12c still supports flashback [29].

Other major DBMS vendors which have provided substantive support for temporal data include IBM and Teradata. IBM DB2 database management systems implement almost all the temporal features in SQL:2011 using slightly different syntax and terminology [30]. Business time, system time and bitemporal tables in IBM DB2 are named as application time period tables, system versioned tables and system versioned application time period tables respectively in SQL:2011 [30]. Slight differences exist between the SQL:2011 specification and the implementation of the temporal features in IBM DB2. In IBM DB2 version 10.5's implementation of temporal concepts, a user does not have the possibility to define their own names for periods when creating business time tables: it is predefined as BUSINESS_TIME. This implementation contrast with the convention for naming application time period tables in SQL:2011, where period names in create table statements are user-defined.

Also, the syntax for creating system time tables in IBM DB2 10.5 differs from system versioned tables in SQL:2011. In the former, three timestamped columns are used to store system time values: two timestamped columns store information about the start and end points of the system time and one column stores information about transaction start time [30]. The transaction start time column tracks when a transaction first executes a statement which changes the table's data. Users are permitted to either manually enter or instruct the system to automatically generate the values for the three timestamped columns mentioned above. In the latter instance, the keyword GENERATED ALWAYS is used; This keyword also exists in SQL:2011 and serves the same purpose. To create business time tables in IBM DB2 10.5, one has to include two columns to represent the start and end time periods, and a PERIOD BUSINESS TIME clause [30]; This is similar to how application time period tables are specified in SQL:2011. Bitemporal tables in IBM DB2 10.5 includes specifying columns for both system time and business time; the same concept in SQL:2011.

In Teradata, the temporal features are based on the TSQL2 specification [31]. Temporal query processing in Teradata is implemented by using functional query rewrites or by implementing a native temporal support in the underlying database engine. The latter approach processes temporal queries by compiling and rewriting them into generic, non-temporal operations [31]. Although some members of the Teradata Company state that the rewrite approach is generally simpler to implement, they also admit that it adds a structural complexity to the original query, which can pose a potential challenge to query optimization [31]. The other way to implement temporal queries in Teradata is to implement temporal database operations such as scans, projections and joins directly in the DBMS internals [31].

The concepts valid time and transaction time are also "employed" by Teradata. Teradata also supports bitemporal tables, but at most one system time and at most one application time is allowed per table [31] as opposed to the specification in SQL:2011, where two values represent the application time. Temporal data types supported in Teradata include the "usual" DATE, TIME, TIMESTAMP and INTERVAL as well as the "much anticipated" PERIOD datatype. Teradata supports the PERIOD data type since V13.0; a PERIOD column can be any date or timestamp type although the beginning and end must have the same type, and the format of the date or timestamp also needs to be specified in the create table statements. The PERIOD data type uses the closed/open concept, where the start of the period is included but the end is excluded. Perhaps Teradata is a "step ahead" with regards to introducing the period data type, which still does not exist in SQL:2011. In table 1, we present an overview of the current state of temporal support for databases by three of the leading DBMS vendors.

Aside implementing their own temporal features, it is also known [30] that some of these DBMS vendors such as IBM, worked with the ANSI and ISO SQL standard committees to incorporate the new temporal extensions into the latest SQL:2011 standard. It is interesting to note that, IBM DB2 was the first database vendor, which fully supported the temporal features of SQL:2011. Defining and using tables in Teradata follows the TSQL2 specification, thereby raising the problem of incompatibility with SQL:2011 [21].

Table -1: Current state of temporal support provided by DBMS vendors

	DBMSs		
Temporal concepts in SQL	**IBM DB2**	**Oracle**	**Teradata**
DATE, TIME, TIMESTAMP data types	✓	✓	✓
INTERVAL data type	✓	✓	✓
PERIOD data type	x	x	✓
System versioned tables	ST table (3 columns for ST)	TT table (1 column for TT)	TT table (1 column for TT)
Application time period tables	BT (2 columns for BT)	VT table (2 columns for VT)	VT (1 column for VT)
System versioned application time period tables	Bitemporal (5 columns for both ST and BT)	Bitemporal (3 columns for both VT and ST)	Bitemporal (2 columns for both VT and ST)

ST –System time **BT**- Business time
TT- Transaction Time **VT** – Valid Time
✓ - available **X** - unavailable

6. CONCLUSION

SQL support for temporal features in databases has undergone reformation over the years. Some of the features had to be dropped while others are still available in the current SQL standard. In figure 2, we look at the trend of developments with regards to some temporal support introduced in the various SQL standards and projects over the years. The figure does not cover all the temporal concepts in the various projects but just a few of the major ones. It can be observed from the figure that the data types INTERVAL and PERIOD have been dropped in the most recent version of SQL, SQL:2011. Also, the main idea behind the concepts "valid time", "transaction time" and "bitemporal" have been maintained over the years; However, each of the standards and projects shown in the diagram referred to the concepts by different names. The difference between the representations of the concepts in various standards does not only lie in the naming, but also the syntax for creating, storing and manipulating data in the respective tables. Looking at the diagram, it is observable that the newer projects improve on the concepts from the preceding ones by either changing syntax or introducing new concepts.

Temporal extensions of SQL have been developed and improved over the years and the level of difficulty associated with managing data related to time in databases is on a lower level now. In spite of this achievement, there is still more which can be done, especially in the area of surveying new forms of temporal support for databases and working towards the inclusion of new temporal extensions in the subsequent SQL standards to be developed.

One major "problem" which has still not been addressed is the absence of a PERIOD data type in SQL. Proposals for PERIOD data type could not go beyond the TSQL2, SQL/Temporal era. Since SQL:2011 did not support this data type, it remains unclear what the future holds with respect to it. Many people anticipate that the introduction of PERIOD data type into SQL will be costly because a lot of the technologies which "work" with SQL would have to be modified to adapt to this data type [23]. As at the time of our research, we found virtually no information about when a new SQL standard will be released and whether or not it will include support released and whether or not it will include support for the PERIOD data type. Aside the PERIOD data type, there are other features which are still not supported in SQL and perhaps some of these features may be introduced in the next standard. These features include but are not limited to: support for multiple application time periods per table and improved support for period joins [23]. A period join is performed by joining a row from one table with a row from another table such that their application time or system time periods satisfy a condition such as overlap. Using the OVERLAPS operator in SQL:2011, inner joins can be performed but additional support is still needed in order to perform outer joins [23]. Another feature which could also be introduced is coalescing, to bring together tuples which have identical attribute values and with timestamps, which are adjacent in time or share some time periods in common [27].

Introducing temporal extensions into SQL has been beneficial to DBMS vendors and application developers. DBMS vendors have and are still making efforts to adapt their systems to support temporal data. Temporal support by DBMSs ensures consistent handling of time-related events and reducing query complexity in temporal databases. A study presented in [21], evaluated the performance of DBMS products with respect to their architectural support for temporal databases and performance when executing temporal queries. The results showed that the support for temporal data is still in its infancy as all the DBMS they considered store their data in regular, statically partitioned tables and rely on

standard indexes as well as query rewrites for their operations, leading to performance overheads [21].

As the SQL standard is revised over the years, it is likely that we will observe a continued improved support for handling temporal data in databases. The field of temporal database research is still active and there even exists an international center, the TimeCenter, which supports temporal database applications on traditional and emerging DBMS technologies [27].

Some members of this Center include very renowned people who have made significant contributions to the field of temporal databases. Temporal database research has surely paid off and although we are not certain of the timing, the database community still awaits the development of next SQL standard after SQL:2011, with the hope that it will provide improved support for temporal databases.

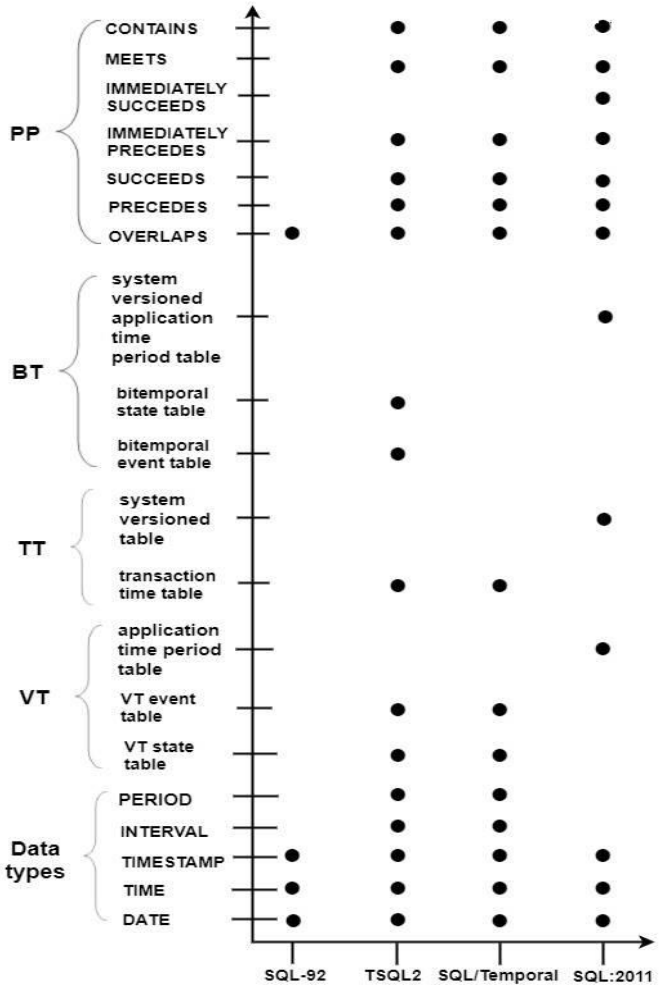

Figure 2: Some temporal features of the various SQL standards and projects. Abbreviations: **PP**- period predicates, **BT** - bitemporal table, **TT** - transaction time **VT** – Valid time

REFERENCES

[1]Snodgrass, Richard. "TSQL: A Design Approach." White paper, University of Arizona, Department of Computer Science, Tucson ,1992

[2]Snodgrass, Richard Thomas, et al. "TSQL2 language specification." Sigmod Record 23.1, 1994, pp 65-86

[3]Eisenberg A.; Melton J., SQL: 1999, formerly known as SQL3, ACM SIGMOD Record Volume 28 Issue 1, March 1999, New York, USA, pp 131 – 138

[4]Snodgrass, R et al., "Transitioning temporal support in TSQL2 to SQL3." Temporal Databases: Research and Practice, Springer Berlin Heidelberg, 1998, pp 150-194

[5]Zemke, Fred. "What's new in SQL: 2011." ACM SIGMOD Record 41.1 ,2012, pp 67-73

[6]Chamberlin, Donald D; Boyce, Raymond F."SEQUEL: A Structured English Query Language" (PDF). Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control (Association for Computing Machinery), 1974 pp 249–264

[7]Snodgrass Richard T. and Ilsoo A, "Temporal Databases," IEEE Computer 19(9), September, 1986, pp. 35–42

[8]Snodgrass, R. "The temporal query language TQuel." ACM Transactions on Database Systems (TODS) 12, no. 2,1987, pp 247-298

[9]Dyreson, C., Fabio G,, Wolfgang K., Nick K., Nikos L., Yannis M., Angelo M. et al. "A consensus glossary of temporal database concepts." ACM SIGMOD Record 23, no. 1, 1994, pp 52-64

[10]Chamberlin, Donald D. "Early history of SQL." Annals of the History of Computing, IEEE 34, no. 4 , 2012, pp 78-82

[11]"The SQL-92 Standard", Second Informal Review Draft, ISO/IEC 9075:1992, Database Language SQL, Digital Equipment Corporation, Maynard, Massachusetts, July 30, 1992

[12]Snodgrass, Richard T. "Temporal databases", In IEEE computer, 1986

[13]Representing time in SQL, Temporal Information Systems lecture notes by Prof. Dr. Rainer Manthey, Department of Computer Science, University of Bonn, Germany, Summer semester 2014

[14]ANSI/ISO/IEC International Standard (ISO/IEC 9075-2:1999) Database Language SQL — Part 2: Foundation (SQL/Foundation), Part 2, September 1999

[15]Van Der L., Rick F., SQL for MySQL Developers: a comprehensive tutorial and reference, Pearson Education, 2007

[16]Y2K bug. Encyclopædia Britannica Online

[17]Melton, J, Alan R. S., SQL: 1999: Understanding relational language components, Morgan Kaufmann, 2001

[18]Darwen, H, Date C.J. "An Overview and Analysis of Proposals Based on the TSQL2 Approach.", 2005

[19]Eisenberg, A., Jim M., Krishna K., Jan-Eike M., and Fred Z. "SQL: 2003 has been published." ACM SIGMOD Record 33, no. 1 ,2004, pp 119-126

[20]ISO/IEC 9075-2:2011, Information technology—Database languages SQL Part 2: Foundation (SQL/Foundation), 2011

[21]Kaufmann, Martin, et al .,Benchmarking Bitemporal Database Systems: Ready for the future or stuck in the past, Proc. of the 17th International Conference on Extending Database Technology (EDBT), March 24-28 2014, Athens, Greece

[22]SQL:2011, International Organization for Standardization (ISO) webshop http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53681

[23]Kulkarni, K. and Michels, Jan-E.. , Temporal features in SQL:2011, ACM SIGMOD Record, 2012, Vols. 41.3, pp 34-43

[24]ISO/IEC 9075-2:2011, Information Technology-Database languages-SQL-Part 2. Foundation (SQL/Foundation), 2011 as cited in Kulkarni K., Michels J.E, Temporal features in SQL:2011, ACM SIGMOD Record, 41.3, 2012

[25]Baumunk, C**.,** "Bitemporaldata.com: An overview of temporal features in SQL:2011", Slides on temporal features of SQL:2011, May 22, 2015

[26]Adrian Billington ,Flashback query in oracle 9i, Oracle Developer.net, November 2002,

[27]Petković, D., Temporal Data in Enterprise Database Systems, The 7th International Conference on Information Technology, 2015, pp 276-282

[28]Irfan Haq, Recovery made simple, Oracle Flashback Query ,Oracle FAQ online, October 2004

[29]Using Oracle Flashback Technology, Database development , Oracle Database Online Documentation 12c Release 1 (12.1) / Database Administration

[30]Saracco, Cynthia M., Nicola, M. and Ghandi, L. , A matter of time: Temporal data management in DB2 10, IBM, April 3, 2012

[31]Al-Kateb, Mohammed, et al., Temporal query processing in Teradata, EDBT '13: Proceedings of the 16th International Conference on Extending Database Technology, ACM, March 2013, pp 573-578