A Relative Analysis of Modern Temporal Data Models

Shailender Kumar
Computer Science & Engineering
University Institute of Engineering & Technology
Maharshi Dayanand University, India
shailenderkumar@aiactr.ac.in

Rahul Rishi
Computer Science & Engineering
University Institute of Engineering & Technology
Maharshi Dayanand University, India
rahulrishi@rediffmail.com

***Abstract*--Time oriented data is better managed through the temporal data models than by implementing them in user applications. There is an improved temporal support available in the modern commercial databases that enable the users to manipulate the temporal data easily. In this paper, we introduced some vital temporal concepts to have a better understanding of the modern temporal data models. After that, we examine few modern temporal data models with built-in temporal logic along with others that are implemented on top of some conventional databases. Finally, a comparison sheet is presented to have the operational dimensions of each model.**

Index Terms*— *bitemporal, SQL:2011, temporal data models, transaction time, valid time

I. INTRODUCTION

Recent years have seen an extensive escalation of the research interest in the field of temporal databases. Consequently, various temporal data models as well as query languages have been suggested. Research on time-varying databases primarily concentrated on the definition of a data model that could support the semantics of the temporal data, its storage, query processing and its implementation.

Time-varying data is expressed by adding timestamp to the values. The different timestamps used may be time points, intervals or set of time intervals. Majority of the existing temporal data models support time intervals. There are different aspects of data values in terms of time. The time when a value of the data becomes effective is called valid time. Whereas, the time at which a data value is recorded in the database is termed as transaction time. And when a data model supports both the times, it is termed as bitemporal [6]. In addition to this, a data model may support arbitrary time attributes with no built in semantics and is termed as user-defined time. The time-varying data in the data model can be handled either by applying the timestamp to the tuples or to the attributes [4].

Around two dozen temporal data models were introduced during 1980s and 1990s [6]. After this period, the research in this area was relatively quiet for some time. The last few years brought back the revitalization in temporal databases and new data models were introduced [5, 13, 14]. The survey of these different temporal data models have been presented in few earlier work [9] also. But none of them include the latest temporal data models [3, 5]. This paper examines most significant temporal models introduced in the last one decade. A comparison sheet is presented in this paper to have close look into the temporal support of these models.

The remainder of this paper is arranged as follows. Section 2 of the paper discusses some key features related to modern temporal data models. Section 3 summarizes various significant temporal data models introduced in the previous one decade. Section 4 evaluates the temporal data models on various parameters. Section 5 concludes the paper and discusses future scope.

II. KEY TEMPORAL CONCEPTS

Before the evaluation of various temporal data models, there are few important temporal concepts [3, 5] which we need to examine.

A. Time Dimensions

The dimension is the most important concept in temporal databases. There are three different variants of time in temporal databases: valid time, transaction time and bitemporal. Valid time is the time when a row is correctly reflecting reality by the user of the database [2]. Transaction time is the time when a row in recorded in the database. The union of valid and transaction time is called bitemporal. The majority of the temporal data models sustain valid time only. The modern temporal data models have a support for all the three dimensions.

B. Timestamping

Timestamping is a concept in temporal databases which explain how time-varying attributes are attached. It has two different variants: tuple timestamping and attribute timestamping. Tuple timestamping refers to adding timestamps to the tuple and it requires first normal form (INF) relations [8, 11]. On the other hand, attribute timestamping uses non-first normal form (NINF) relations and the timestamps are attached to the attributes [7, 10, 12]. The performance of these two approaches in BCDM data model is well taken up in Attay's work [4].

C. *Temporal Key Constraints*

In temporal databases, there are two different types of key constraints: first one relates to primary key whereas the other relates to referential integrity. The value of the primary key varies with the time dimension used in the database. If the time dimension is valid time, the temporal primary key must include the temporal attribute in addition to the relational primary key. But when it is transaction time, the temporal primary key is same as the relational primary key. The concept of temporal referential integrity is same as in relational databases.

D. *Period Data Type*

The PERIOD data type includes the combination of subsequent time units. This data type uses a closed-open period in which a time period starts from and includes the start time but exclude the end time. The data type used for the start and end time can be TIME, DATE, or TIMESTAMP. There are various functions which are used on the instance of PERIOD data type. Teradata temporal data model has the built-in support for the PERIOD data type which is absent in other modern temporal data models.

E. *Coalescing*

Coalescing is the process of merging two overlapping or adjoining value-equivalent tuples to preserve the integrity constraint. The requirement of Coalesce occurs when the UPDATE statement is executed and the non temporal attributes of the tuples are same and their timestamp overlaps or consecutive. None of the modern temporal data model supports Coalescing [3] and is only applied through hand-coded solutions.

III. TEMPORAL DATA MODELS

A temporal data model attempts to simultaneously satisfy many objectives. It should be able to represent the semantics of the application in a comprehensible manner. It must be a consistent and minimal extension of existing data models. There are numerous temporal data models introduced till date but only few of them have made an impact. Following are the few significant temporal models introduced in the last one decade.

A. *Temporal Data Model of SQL:2011*

SQL:2011 was released in December 2011 replacing its previous version SQL:2008. Before this release, users were forced to implement the time-varying support through application logic which was complex and time consuming [3]. This standard has used a closed-open period model where a time period expresses all time granules beginning from and including the start time, continuing to but excluding the end time [2, 16].

The standard incorporates three forms of tables: system-versioned tables, application-time period tables and bitemporal tables. The support for the transaction time is provided by system-versioned tables whereas valid time support is provided by application-time period tables. The system time period is specified by the keyword SYSTEM_TIME whereas the identifier for the application-time period is specified by the user. Example of application-time period table is as follows:

Example 1.

```
CREATE TABLE Employee (
E_id  INT,
E_st  DATE,
E_et  DATE,
E_dpt INT,
PERIOD FOR E_period(E_st, E_et),
PRIMARY KEY (E_id, E_period WITHOUT
OVERLAPS));
```

The data types of the period start and end attributes must be same. The integrity constraint is specified using PRIMARY KEY keyword with the option WITHOUT OVERLAPS. Although SQL: 2011 has included various important extensions for managing time-varying data yet there is certain space for further incorporations [2, 5]. Few of them are the support using PERIOD data type, Coalescing, period joins etc.

B. *Temporal Data Model of IBM DB2*

The built-in support in IBM DB2 for managing time-varying data minimizes the application logic and provides consistent handling of time-oriented events across all applications [15]. By the usage of simple declarative SQL statements the temporal data can be managed in this standard. Thus, it eliminates the need for such logic to be hand-coded using triggers and stored procedures.

The implementation of application-time period tables in IBM DB2 is identical to that in SQL: 2011 [1].The only difference is in the nomenclature of such tables which are called tables with business time in DB2. Example of table with business time is as follows:

Example 2.

```
CREATE TABLE Employee (
E_id INT,
E_st  DATE,
E_et DATE,
E_dpt INT,
PERIOD BUSINESS_TIME (E_st, E_et),
PRIMARY KEY (E_id, BUSINESS_TIME WITHOUT
OVERLAPS)
);
```

IBM DB2 does not allow users to define name for the specified time period in the PERIOD clause. Instead of user defined names, BUSINESS_TIME keyword is used.

C. *Temporal Data Model of Teradata*

Teradata Database version 13 and 14 has many new features in its SQL designed specifically for temporal support. Teradata provides support for all three time dimensions i.e. valid, transaction and bitemporal [5, 17]. Bitemporal tables contain the VALIDTIME and TRANSACTIONTIME clause as shown in the example below:

Example 3.

```
CREATE MULTISET TABLE Employee (
E_id INT,
E_vtperiod PERIOD (DATE) NOT NULL AS
VALIDTIME,
E_ttperiod PERIOD (TIMESTAMP(6)) NOT NULL AS
TRANSACTIONTIME,
E_dpt INT
) PRIMARY INDEX (E_id)
```

The distinct feature of this temporal data model is the support it provides for the PERIOD data type which is absent in other modern temporal data models. The support for coalescing of data is missing in Teradata. Teradata has the support for PRIMARY KEY option also but the support for temporal referential integrity is not incorporated.

D. *Temporal Relational Model for managing Patient Data (TempR-PDM)*

TempR-PDM [8] is a conceptual model for the management of temporal data in the relational database model with nominal extension of temporal logic. This model was implemented on patient database. The model is composed of three components: the entity, attributes and relationship. The temporal entity must have a primary key which is a combination of temporal and non-temporal attributes. The model supports tuple time-stamping. The valid time is replaced by activation_start, and activation_end time. The transaction time is also replaced by update time to take care of any change in data.

The activation time is the constituent of the primary key of the temporal entity. The time can be represented as time points and by intervals as well. The data is assumed to be in INF. The model supports multiple granularities. The default time granularity used is minute but it can be changed through conversion functions. The advantage of this model is that it retains the fundamental feature of the relational model and causes less additional cost to enable its easier implementation.

E. *Temporal Functionality In Objects with Roles Model (TF-ORM)*

TF-ORM [10] is a temporal object oriented data model. It is distinct from other similar temporal object based data models due to the use of the role feature. Several roles may be defined for each class. As an example, in the Institute Employee may be in different roles as Principal, faculty or the staff. The object is an instance of one class only but it can play distinct roles at the same time. Moreover, the object can have multiple instances of the same role.

TF-ORM environment is implemented in Oracle database through triggers, functions, stored procedures and views. This model was presented to implement a bitemporal database by the use of temporal intervals. The mapping to the relational database is done by defining a table for each class and for every role of a class. The static properties are included in this table. Each dynamic property is represented by a different table comprising of the attribute values together with the temporal timestamps explicitly represented as attributes. The implementation of the Environment is completed through the constructed tool that checks the TF-ORM query and mapped to the SQL query if found error free. The object oriented programming language Delphi was used to implement the specification tool and the query interface.

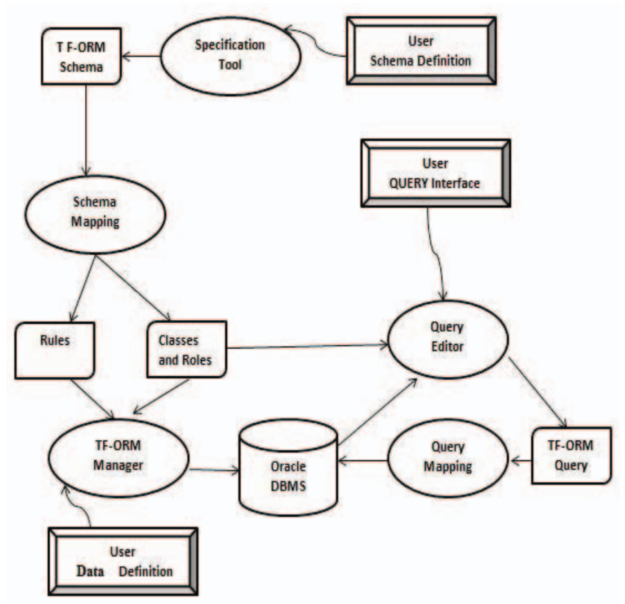

Fig. 1. TF-ORM environment [10]

IV. COMPARISON TABLE

Table 1 summaries the important operational dimensions of the various temporal models discussed in this paper. The table indicates that most of the databases used are relational. The first column of the table gives the name of the corresponding temporal data model. The second column indicates the time dimension of the model. The valid time and transaction time are represented by VT, TT respectively. The relations in tuple timestamped models are in first normal form (1NF). On the contrary, the attribute timestamped model uses non first normal form relations (N1NF). The symbol '+' indicates the presence and '-' indicates the absence of the built-in support. The last column reflects whether the corresponding temporal data model support the primary key or/and referential key constraint. The absence of any built-in support means that the same required being implemented using hand coded applications [3].

Identifier	Time Type	Timestamping	Period data type/ Coalescing [1, 3, 5]	Normal Form	Query language	Data Modeling	Primary Key Integrity/ Referential Integrity
TF-ORM [10]	Bitemporal	Attribute	-/-	NINF	Available	Object	+/+
TempR-PDM [8]	Activation time, Update time	Tuple	-/-	1NF	Available	Relational	+/+
SQL:2011 [2, 16]	VT, TT, Bitemporal	Tuple	-/-	1NF	Available	Relational	+/+
TERADATA [5, 17]	VT, TT, Bitemporal	Tuple	+/-	1NF	Available	Relational	+/-
IBM DB2 [1, 15]	VT, TT, Bitemporal	Tuple	-/-	1NF	Available	Relational	+/+

TABLE I. COMPARISON OF TEMPORAL DATA MODELS

V. CONCLUSION AND FUTURE SCOPE

We have evaluated few important temporal data models in this paper. An effective and powerful temporal information system should incorporate good temporal data model and powerful query language. Therefore, the evaluation of the models is made in relation to some key concepts discussed in section 2.

The database systems that do not support time varying feature require the users to provide the requisite support through application coding. We believe that our effort will be helpful for the researchers as they have the option to choose among the competing proposals to deal with time varying data.

In spite of so much research in temporal databases, there is still a scope for their implementation especially in the area of open source. Moreover, the temporal support using PERIOD data type, attribute timestamping, Coalescing, period joins etc. is still in an infancy stage even in modern commercial databases.

References

[1] Petković, Dušan. "MODERN TEMPORAL DATA MODELS: PROPERTIES AND DEFICIENCIES" 6th International Conference on Information Technology (ICIT) 2013.

[2] Kulkarni, Krishna, and Jan-Eike Michels. "Temporal features in SQL: 2011."*ACM Sigmod Record* 41.3 (2012): 34-43.

[3] Künzner, Florian, and Dušan Petković. "A Comparison of Different Forms of Temporal Data Management." *Beyond Databases, Architectures and Structures*. Springer International Publishing, 2015. 92-106.

[4] Atay, C. "A Comparison of Attribute and Tuple Time Stamped Bitemporal Relational Data Models." *Int. Conf. on Applied Computer Science*. 2010.

[5] Petković, Dusan. "Modern Temporal Data Models: Strengths and Weaknesses." *Beyond Databases, Architectures and Structures*. Springer International Publishing, 2015. 136-146.

[6] Jensen, Christian S., and Richard T. Snodgrass. "Temporal data management." *Knowledge and Data Engineering, IEEE Transactions on* 11.1 (1999): 36-44.

[7] Chau, Vo Thi Ngoc, and Suphamit Chittayasothorn. "A temporal compatible object relational database system." *SoutheastCon, 2007. Proceedings. IEEE*. IEEE, 2007.

[8] Burney, Aqil, Nadeem Mahmood, and Kamran Ahsan. "Tempr-pdm: a conceptual temporal relational model for managing patient data." *Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*. World Scientific and Engineering Academy and Society (WSEAS), 2010.

[9] Miao, Decheng, Jianqing Xi, and Jindian Su. "A Category Theoretic Method of Temporal Data Model." *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*. IEEE, 2013.

[10] Edelweiss, Nina, et al. "A temporal database management system implemented on top of a conventional database." *Computer Science Society, 2000. SCCC'00. Proceedings. XX International Conference of the Chilean*. IEEE, 2000.

[11] Máté, Ján, and JiYí Šafařík. "Transformation of relational databases to transaction-time temporal databases." *Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on the*. IEEE, 2011.

[12] Tansel, Abdullah Uz. "On handling time-varying data in the relational data model." *Information and Software Technology* 46.2 (2004): 119-126.

[13] Mahmood, Nasir, et al. "Fuzzy-temporal database ontology and relational database model." *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*. IEEE, 2012.

[14] Terenziani, Paolo. "Coping with Events in Temporal Relational Databases." *Knowledge and Data Engineering, IEEE Transactions on* 25.5 (2013): 1181-1185.

[15] http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata.

[16] Zemke, Fred. "What's new in SQL: 2011." *ACM SIGMOD Record* 41.1 (2012): 67-73.

[17] Snodgrass, Richard T. "A Case Study of Temporal Data." *Teradata Corporation* (2010).