ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

SURVEY REPORT

Available Online at www.ijarcs.info

A Comparative Study of Tuple Timestamped Data Models

Dr. Shailender Kumar
Department of Computer Science and Engineering
Ambedkar Institute of Advanced Communication Technologies & Research, Geeta Colony,
Delhi, India

Disha Aggarwal
Department of Computer Science and Engineering
Ambedkar Institute of Advanced Communication Technologies & Research, Geeta Colony,
Delhi, India

Abstract:Time is one of the most challenging facet to handle in real world applications. Most applications access or manipulate temporal data so such applications rely on temporal databases which records the time varying data. The objective of this paper is to provide a concise overview on concepts of temporal database and review the work in the field of tuple timestamping data models. The paper introduces the basic of conventional database and then compares it with the temporal database. The paper evaluates different time dimensions and query languages along with them. A comparative analysis of different temporal models is done based on some parameters. Finally, all the temporal models are summarized and a conclusion is given with the future work in the field of temporal database.

Keywords: database; temporal database; attribute timestamping; tuple timestamping; transaction time; valid time; data models

I. INTRODUCTION

The current interest in the relational approach to database is because of the Codd, who presented the relational model which includes the relational structure and its associated advantages. But this relational model does not consider the temporal dimension as it does not keep a track of past or future aspects of time in database but only focuses on the bivalent Boolean logic. Applications like banking, inventory management, healthcare management, reservation systems, insurance applications, weather monitoring, etc. involves the data that changes with time and it is important to keep record of all that information.

A database which stores different time aspects (past, present and/or future) of the data is known as temporal database. In this section we discuss the basic database concepts and how the conventional databases are different from the temporal databases.

A. Basic Conventional Database Concepts

A database management system (DBMS) is a collection of program that enables users to create and maintain a database. A database is a collection of related data. By data, we mean known facts that can be recorded and that have an implicit meaning [2]. So the main objective of DBMS is to provide a convenient and effective method of defining, storing, retrieving and manipulating the data contained in the database.

Data can be divided into: Past Data, Present data and Future Data. The data about the past events and circumstances which existed before in the database is referred as Past Data.

The data that is valid for current time is known as Present Data and sometimes denoted with "now" keyword. The data which is generated over a specific time range, let say, a day or week ahead of the current time is known as Future Data.

To structure this data, data models are needed. Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization. Data Model includes two types of schema: Physical Schema andLogical Schema. Physical Schema describes the database design at the physical level, while the Logical Schema describes the database design at the logical level [2].

These data model schemas must exhibit data independence. Data Independence is the ability to change the schema at one level of the database system without having to change the schema at the other levels [11]. Data Independence is of two types: Physical Data Independence and Logical Data Independence. Physical Data Independence is the ability to change the internal schema without affecting the conceptual or external schema. Logical Data Independence is the ability to change the conceptual schema without affecting the external schemas or application programs.

Conventional Database Data Models can be classified as:

i. Hierarchical Database Model: This data model organizes the data in a tree-like structure in which each child node can have one parent node only and is connected to one another through links.
ii. Relational Model: This data model maintains data in the form of tables, also known as relations, consisting of rows and columns.
iii. Network Model: This data model organizes the data in the form of graphs and all the nodes are linked to each other, without any hierarchy, using links.
iv. Entity-Relationship Model: This data model is composed of entity types (objects) and specifies relationships that can exist among those objects.

v. Object-Based Data Model: This data model extends the E-R model with concepts of encapsulation, methods and object identity.

B. *Relational Databases*

In the last two decades, the relational data model has gained popularity because of its simplicity and solid mathematical foundation. However, the relational data model as proposed by Codd does not address the temporal dimension of data. It represents the state of an endeavour at a single moment of time. The variation in the contents of the database can be seen as a modification, as new information is added, deleting the old, out-of-date data from the database.

As can be seen from Figure 1, the Employee Table consists of columns EmpID, Count, Ename, Sal from which (EmpID, Count) acts the primary key of the table. Count column stores the number of transactions performed on the Employee table. In Figure 2, the table is modified by adding new information, with the previous data in the same table, as a new tuple.This increases the size of the relational database with the increase in the number of tuples, added with each modification.

EMPLOYEE TABLE

EmpID	**Count**	**Ename**	**Sal**
E123	0	Rajiv	30000
E234	0	Simran	35000

(BEFORE MODIFICATION)
Figure. 1

EMPLOYEETABLE

EmpID	**Count**	**Ename**	**Sal**
E123	0	Rajiv	30000
E123	1	Rajiv	35000
E234	0	Simran	35000
E234	1	Simran	40000

(AFTER MODIFICATION)
Figure. 2

The current data is captured as a snapshot and discards the time aspect of past data. Also, transactions are done according to their arrival order providing no guarantee of their completion time.This is not satisfactory for applications that require past, present, and/or future data values to be dealt with by the database. This arises the need to use temporal database which can store the time-variant data without discarding past values [19].

In the following sections: Section 2 will introduce the temporal database concepts considering the time dimensions and types of timestamping. Section 3 discusses about the work done in the area of the tuple timestamped data models. Also, it gives a comparative study with the research criteria of different data models. Section 4 states the conclusion with the future work.

II. TEMPORAL DATABASE CONCEPTS

A. *Time Dimensions*

Temporal database supports three types of time dimensions, which exhibits independencefrom each other: user-defined, valid and transaction time. User-defined time is a time representation designed to meet the specific needs of users.

Valid time specifies when certain conditions in the real world are, were or will be valid. Valid time can be represented with single-chronon identifiers (event timestamps), with intervals (as interval timestamps), or as valid-time elements, which are finite set of intervals [4].

Transaction time automatically captures changes made to the state of time-variant data in a database. This time dimension represents the time period during which an instance is recorded in the database. Transaction time is mostly used to support versioning, which generally implies an object-oriented data model [14].

There is also another form of time-variant data, called bitemporal data. Bitemporal data are union of valid time and transaction time data. Several data models concerning bitemporal data are known in the literature: The Bitemporal Conceptual Data Model (BCDM) is a very simple model capturing the essential semantics of time-variant relations. Another example of a bitemporal model is Nested Bitemporal Relation Data Model (NBRDM). The model, as the name implies, is based upon a nested relational schema.

B. *Timestamping*

The temporal relational data models are categorized according to the timestamps attached and the time dimension they support. The data that have time-related information are known as temporal data.
A timestamp is a time value associated with a data value that might be a temporal element, time interval, or time point [18].
There are two types of timestamping:

1. **Attribute Timestamping:**

In this type of timestamping, value of timestamp is attached with the attributes of the database table. The various characteristics of attribute timestamping:

i. It uses non-first normal form to store time-variant data.
ii. It avoids redundancy and is more expressive because it doesn’t contain the repetition of the multiple rows in the table due to change in the attribute values over the time.
iii. It uses single relation/tuple to store time varying attributes. It doesn’t splits the time-variant and time-invariant data into several tuples.
iv. It may not be capable of efficiently using existing storage structure. Hence, it cannot use high effective relational techniques, such as Query Evaluation.
v. It performs better for queries with low levels of nesting because as the nesting increases, it will be more

complex to apply timestamp values with those attributes that are divided further in more sub-attributes.

vi. Attribute Timestamping is more suitable to execute valid time queries as it executes them faster.

vii. This timestamping is least suitable for bitemporal queries as it executes them slowly.

EMPLOYEE TABLE

EmpID	**Ename**	**Sal**		
		T Value	**T Start**	**T End**
E123	Rajiv	30000 35000	12/02/2012 15/02/2014	15/02/2014 20/02/2016
E234	Simran	35000 40000	15/03/2013 20/03/2015	20/03/2015 25/03/2016

Figure 3. Attribute Timestamping

2. Tuple Timestamping:

In this type of timestamping, value of timestamp is attached with the tuples of the database table. The various characteristics of tuple timestamping:

i. It is based upon the first normal form (1NF) to store time-variant data.

ii. It contains high data redundancy because of the repetition of multiple rows of a table due to change in attribute values over the time.

iii. It uses relational tables to represent time-invariant as well as time-variant data. It splits both time-variant and time-invariant data into several tuples.

iv. It is capable of efficiently using existing storage structure. Hence, it can use high effective relational techniques.

v. It performs better for more complex nesting structure of the query as it adds the timestamp value to the row of the table. So, it results in less complex structure after adding timestamp value.

vi. Tuple Timestamping is less suitable to execute valid time queries as it executes them slowly.

vii. This timestamping is more suitable for bitemporal queries as it executes them three times faster than attribute timestamping does.

EMPLOYEE TABLE

EmpID	**Count**	**Ename**	**Sal**	**T Start**	**T End**
E123	0	Rajiv	30000	12/02/2012	15/02/2014
E123	1	Rajiv	35000	15/02/2014	20/02/2016
E234	0	Simran	35000	15/03/2013	20/03/2015
E234	1	Simran	40000	20/03/2015	25/03/2016

Figure 4. Tuple Timestamping

III. TEMPORAL DATA MODELS

A. *Introduction*

The temporal models are used to manage both types of data: time-variant as well as non-time variant. Time can be added to any of the data model like entity-relationship model, semantic data models, knowledge based data models and deductive databases. But most of the work has been done on relational data models and object-oriented data models only. That's why in this paper, temporal aspects of relational and object-oriented data models are discussed only.

In conventional database systems, the management of time-variant data is done entirely at the level of user-defined time in which the attribute values are drawn from a temporal domain. The SQL-92 Date, which is a tuple calculus-based language, has granularity of a day; a Time has a granularity of a second, having a range of 100 hours only. A Timestamp combines the range of the Date with the granularity of a second (there are Timestamp variants with a granularity of fractions of a second) [3].

1. ***Valid Time***: Temporal Data Models can be compared on the basis of the valid time dimension. This can be done as there are three ways to represent the valid time in any temporal data model. These are: single chronon (event timestamp), intervals (interval timestamp), or as valid-time elements (finite set of intervals) [4]. In temporal data models the valid time can be associated in three ways i.e. with the values of individual attributes, attribute groups, or with the entire tuple [1]. Some of the examples of valid time models are: Ariav, HDM, Lum, Sadeghi, Jones, Navathe, Snodgrass, Sarda, and Yau.

2. ***Transaction Time***: Object oriented data models are generally identified by implementing versioning on it and transaction time supports the versioning [1]. These data models allow arbitrary, user-defined identifiers to be associated with versions. An entire version hierarchy can also be associated with a version if required by a data model. Some of the examples of transaction time models are: BCDM, Ben-Zvi, Lomet, ADM, Snodgrass, and Postgres.

3. ***Set of Temporal Data Models*:**A temporal data model should meet the certain goals, which can lead to the best possible outcomes. The application which is going to be modelled should have its semantics captured clearly and precisely. It should extend the already existing data model, like relational model, so that it will produce more consistent and accurate outcomes. It must represent all the time-variant and time-invariant behaviour of the objects logically. But, this is probably not possible to have all these features while designing a temporal data model.
   There exists an extension of SQL i.e. TSQL2 which uses Bitemporal Conceptual Data Model as its underlying data model which is used to capture the essential semantics of time-varying relations. A different model i.e. representational data model is utilized for implementation with assured high

performance. Another data model i.e. presentational data model depicts the time-varying behaviour to the user. So, no single data model can achieve all the goals that is why a set of data models should be used in order to fulfil all the desired requirements with best possible outcomes.

B. Query Languages

A data model also contains a set of operations on objects, in combination with set of objects having structures and a set of constraints on those objects. This set of operations is known as the temporal query language which is used to access, modify, and record the objects in data model [10]. Some of the query languages are: ILs, an intensional logic devised in computational linguistics [3]; Quel, defined for Ingres relational DBMS, is a tuple calculus-based query language; relational algebra is a procedural language, denoting relations as objects; SQL, a language of conventional relational database, is a tuple calculus-based language.

1. ***User- Defined Time*:**This time aspect supports both the data models and query languages. Most of the commercial relational DBMSs supports user-defined time. Time can be treated as an abstract data type with the help of its own predefined set of operations. Postgres, being an object-oriented query language, supports this approach.

2. ***Valid Time*:** As valid time supports the future aspects of certain data besides the past values of that data, it has greater influence than the user-defined time. Valid-time can be added to the data models and on the query languages with few different approaches.
   The foremost approach is to directly utilize the extensively responsive ability of relational or object-oriented data model. This approach is the easiest one as it will not require any changes to the existing data model or the query languages. But, there occurs one problem in this approach that it makes the query optimization extremely more difficult. This happens because of the lack of suggestions provided by the query language, whether the access methods that are concerned with time-varying values should be employed or not.

   In the next approach, data models and query languages can be extended with the required needs by showing their support to the time-varying figures. This approach cannot be used for relational data query languages. The problem of query optimization is still present in this approach as user-defined functions do all the manipulations related to time values.
   The former approach is used widely by temporal relational query languages. According to this, the usageof distinctive data model concept and query languages can support the information related to variation of the valid time to the great extent.

3. ***Transaction Time*:**Transaction Time defines the state of time when the fact is logically entered in the database. It does not indicate the validity of the fact in either respect of past or future time.
   It is used when one wants to roll back to the time when a particular fact was stored in the database irrespective of its validity.
   Transaction time supports the following type of versioning:

   i. Extensive Versioning
   In this type, versioning of tuples, object instances or attributes is done by themselves. In support of this versioning, there are different approaches same as with the valid time. In the first one, data models are used directly without doing any changes in them or in the query languages. Transaction time can be adapted irrespective of the time semantics in the database.The next approach extends the data model and query language so that the information related to time-varying data can be supported. In the last approach, some modifications are done in the existing data model and query language according to the required needs so that it can support time-varying data.
   Transaction time can be represented as a graph and not as a tree because it consists of branching of time whenever the objects are versioned, in addition to the feature that new version is created by merging the two old versions.

   ii. Schema Versioning
   A schema of an application can be modified according to the variation in the application's needs. So, multiple schemas can be altered logically in schema versioning. It supports the versioning of the definition of the objects in the database. Schema versioning can be applied in both relational databases and object-oriented databases.

The essential point that needs to be kept in mind is that if extensive versioning is applied then support for schema versioning may or may not be present. But, if extensive versioning is not adopted then there exists no relevance in using schema versioning there.

C. Data Models

In temporal data models, there exists some relational data models and object-oriented data models. A brief study is done on the previous data models, which get out-dated with time, just to get an overview of the work that has been done related to temporal data models so far.

i. Ariav proposed a data model named as Temporally Oriented Data Model [13]. The researcher uses relational data model as the base model, incorporating both the time dimensions that is, valid time and transaction time. This model uses the single chronon (event timestamp) representation of time. It supports the storage and retrieval of data through the use of a query language, SQL.
ii. Ben-Zvi proposed a Time Relational Model (TRM) in order to model the different aspects of time [15]. TRM uses first normal form relations and supports both type of time dimensions (valid and transaction time). This model uses intervals, which are pairs of chronons, to represent time in the relational data model used.

iii. There exists a model named as DATA, which is based on relational data model. This model uses transaction time dimension only to store the time at which data was entered in the database. This model uses a single chronon representation of time to timestamp the value. It is only a proposed model which has no implementation.

iv. Another model is DM/T which uses relational data model as a base model. Timestamping is done using one dimension only that is, transaction time. This model does not deal with the future aspects of time. Time representation is done by applying event timestamps in database. Relational Algebra is used as a language to retrieve the data from database.

v. There is another model, Historical Data Model, which stores the history of data so that user can use the previous data by rolling back it to the specific period of time. This is done using the relational data model having valid time as the time dimension to store the validity of data also. Time is represented as single chronon or event timestamp and uses a query language, ILs (intensional logic).

vi. Postgres is another model which is based on object-oriented data model. This model uses transaction time dimension to store the previous data but does not stores the future aspect of time. Time is represented in the form of intervals (pair of chronons) and uses Postquel language to retrieve and store the data, which is based on a language Quel (tuple-calculus based query language).

vii. Navathe proposed a data model named as Temporal Relational Model, based on the relational data model. This model can be expressed as temporal extension of the relational data model. This model uses valid time dimension so as to store validity of the data in near future and the representation of this valid time is done using the intervals of time. This model uses a temporal extension of SQL known as TSQL.

viii. Segev presented implementation of Temporal Data Model which is based on the relational data model [17]. It is implemented using valid time dimensions and represented as single chronon. To access the data, this model uses TDM language which is based on SQL.

ix. Weiderhold proposed a data model, Time Oriented Databank Model, having relational data model as underlying model. This model is based on valid time dimension with the representation of time as single chronon.

x. C.S. Jensen and R.T. Snodgrass explained a temporal approach named as Bitemporal Conceptual Model (BCDM) [6]. It supports both types of time dimension (valid and transaction time) and is linear and discrete in nature. The main advantage of this model is that it remains simple and understandable while storing the temporal data. But, it is not suitable to manage and implement large number of tuples.

xi. DebabrataDey, Terence M. Barron and Veda C. Storey suggested Temporal Event Entity Relationship Model (TEERM) in 1995 [7][8]. It is bitemporal in nature. This model captures the real world aspects and introduced new relationships like static relationship, dynamic relationship and quasi-static relationship. But, representation of events can sometimes introduce redundancy into tables.

xii. Nina Edelweiss, Patricia N. Hubler, Mirella M. Moro and Giovani Demartini represented implementation of Temporal Functionality in Objects with Roles (TF-ORM) data model [5]. It is an object-oriented data model but differs from other object-oriented data model in a way that it uses a role concept to represent the different behaviours of an object. There can be several roles present for each class, and each role will depict a different behaviour for the object of that class. It uses TF-ORM query language as its language to retrieve and access the data.

xiii. Jan Mate and Jiri Safarik presented an implementation which uses an automatic rule based schema transformation technique to convert relational database to temporal database [9]. It uses schema versioning and versioned table techniques. Implementation of this model is done on Postgres database with SQL as query language and PL/SQL as implementation tool language.

xiv. Vincent S. Lai, Jean-Pierre Kuilboer and Jan L. Guynes suggested a model as an extension of Enhanced Entity Relationship (EER) [12][8]. This model is named as Temporal Enhanced Entity Relationship (TEER), which uses both time dimensions (valid and transaction time) and stores the full history of every entity and its associated relationships. This model can easily be mapped to an extension relational model.

xv. Gadia-3 identified a temporal extension to SQL model, which was proposed on the basis of convention database model given by Navathe [4] [16]. It is a valid time model which allows both the attributes that is, time-varying and non-time varying but they must all be of the same type. It can cope with the multiple occurrences of value equivalent events in the same partition of the timeline.

D. Comparative Analysis

The tuple timestamped data models are compared as per the following parameters:

i. Time Dimension: This characteristicdefines the different time aspects of models whether they use valid time, transaction time or both of them.

ii. Implementation of Model: This characteristicsdefines that whether the model has implementation or not.

iii. New Proposed Language: This depicts if any new language is proposed with the model.

iv. Implementation Language: This specifies a language that was used to implement that model.

v. Extended Non-Temporal Model: This criteria will depict that on which convention database that temporal model is proposed.

vi. Language used to Query: This depicts a language that is used to access and manipulate the data in the particular model.

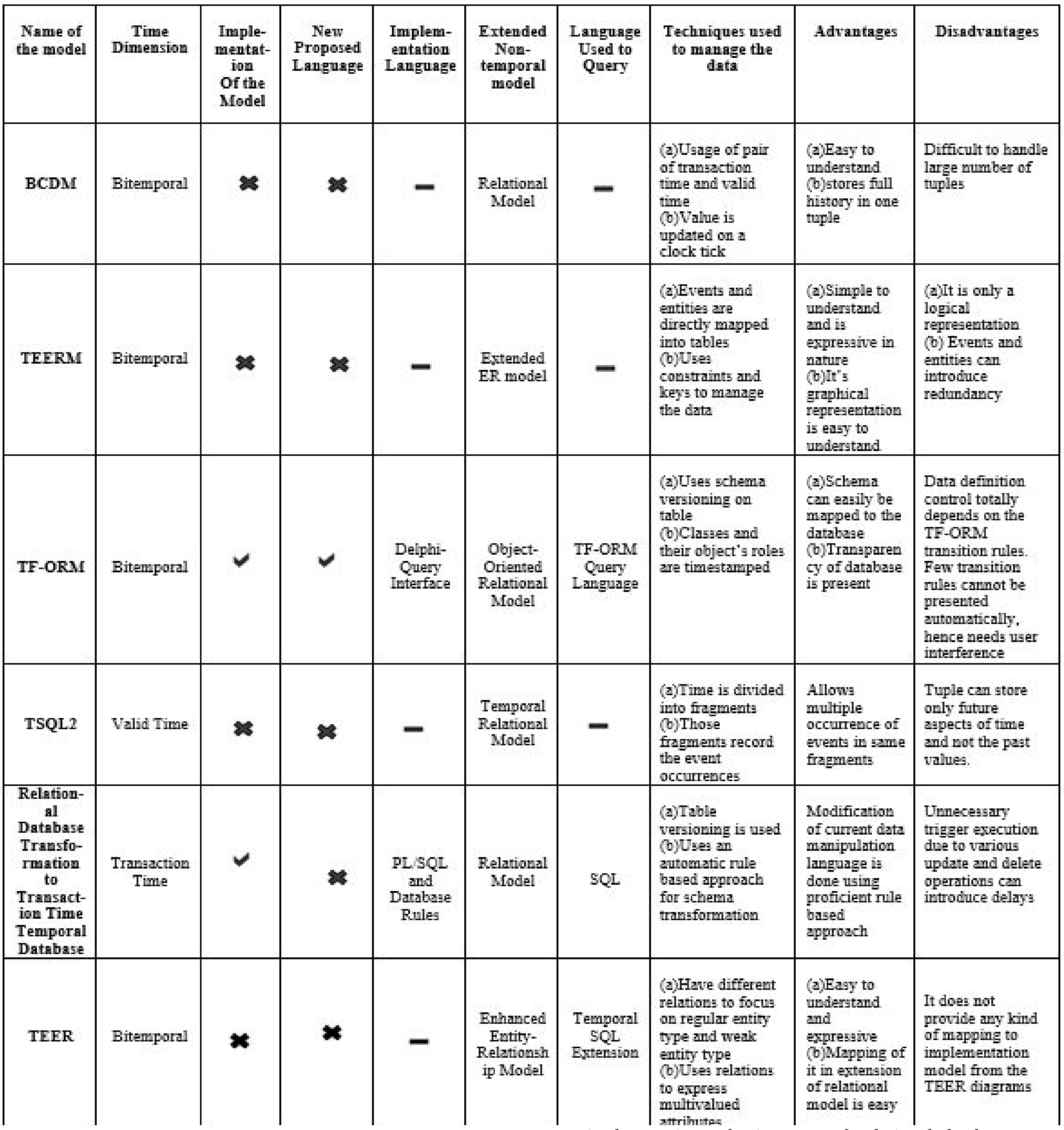
vii. Techniques used to manage the Data: This field enumerates the various data management techniques used to manage the data in the models.

viii. Advantages: This specifies the various advantages that the model persists.

ix. Disadvantages: This specifies the different disadvantages that the model has.

A comparison of some prominent tuple timestamped temporal data models is done as shown in the Table I.

TABLE 1. COMPARISON TABLE



Name of the model	Time Dimension	Imple-mentat-ion Of the Model	New Proposed Language	Implem-entation Language	Extended Non-temporal model	Language Used to Query	Techniques used to manage the data	Advantages	Disadvantages
BCDM	Bitemporal	✖	✖	—	Relational Model	—	(a)Usage of pair of transaction time and valid time (b)Value is updated on a clock tick	(a)Easy to understand (b)stores full history in one tuple	Difficult to handle large number of tuples
TEERM	Bitemporal	✖	✖	—	Extended ER model	—	(a)Events and entities are directly mapped into tables (b)Uses constraints and keys to manage the data	(a)Simple to understand and is expressive in nature (b)It's graphical representation is easy to understand	(a)It is only a logical representation (b) Events and entities can introduce redundancy
TF-ORM	Bitemporal	✓	✓	Delphi-Query Interface	Object-Oriented Relational Model	TF-ORM Query Language	(a)Uses schema versioning on table (b)Classes and their object's roles are timestamped	(a)Schema can easily be mapped to the database (b)Transparen cy of database is present	Data definition control totally depends on the TF-ORM transition rules. Few transition rules cannot be presented automatically, hence needs user interference
TSQL2	Valid Time	✖	✖	—	Temporal Relational Model	—	(a)Time is divided into fragments (b)Those fragments record the event occurrences	Allows multiple occurrence of events in same fragments	Tuple can store only future aspects of time and not the past values.
Relation-al Database Transfo-rmation to Transact-ion Time Temporal Database	Transaction Time	✓	✖	PL/SQL and Database Rules	Relational Model	SQL	(a)Table versioning is used (b)Uses an automatic rule based approach for schema transformation	Modification of current data manipulation language is done using proficient rule based approach	Unnecessary trigger execution due to various update and delete operations can introduce delays
TEER	Bitemporal	✖	✖	—	Enhanced Entity-Relationsh ip Model	Temporal SQL Extension	(a)Have different relations to focus on regular entity type and weak entity type (b)Uses relations to express multivalued attributes	(a)Easy to understand and expressive (b)Mapping of it in extension of relational model is easy	It does not provide any kind of mapping to implementation model from the TEER diagrams

IV. CONCLUSION AND FUTURE WORK

The research in the area of temporal database has been going on from 20 years or more. In near 70's Codd proposed the relational model with its relational structure, due to which the main focus was only on the relational model. With time when it was required to add time with the data in an organized way, it results in temporal relational databases. After some decades, need arises to capture real world entities with this time attribute which results in temporal object-oriented data models. A discussion is done on the following points:

- A brief discussion is done on the basic conventional databases including relational database, which is widely used almost everywhere.
- A need of temporal database is reviewed as what problems were prevailing in result of which we require temporal database.
- Various time dimensions or time aspects of the temporal models are discussed and well understood.
- A lot of research has been done on both types of temporal data models: relational and object oriented models, discussing about their complex nature and structural design. But, none of the models satisfies all the desired objectives as an individual model. So, it is recommended to use a suite of data models for the best outcomes.
- There are various kinds of query languages exist that can be used to access or manipulate these data models. All the languages are easy to understand and implement.
- Many temporal data models are discussed from which some are just conceptual models but are equally important as they form a basis for many complex models. But there are some models also, which has there implementation done.

Areas that need to be addressed in near future:

- Attaining a suitable level of performance is a big task that needs to be considered in temporal databases as it records thevoluminous amount of data, and sometimes heterogeneous in nature, which will take plenty of time to get processed. There is a need to do analysis of the temporal indexing and temporal join algorithms in order to increase performance.
- As tuple-timestamping increases redundancy in the table, it is needed to manage the increasing size of the table because it will become impractical to store all the data in main memory of one machine. Other storage options can be explored like a distributed environment can be implemented to resolve this.

V. REFERENCES

[1] S. Jensen, J.Clifford, R.Elmasri, S.K. Gadia, P. Hayes, and S.Jajodia “A consensus glossaryof temporal database concepts”, eds., TechnicalReport R 93-2035, Dept. of Mathematics and Computer Science, Inst. for Electronic Systems, Denmark, Nov. 1993.

[2] R. Elmasri and S.Navathe, "Fundamentals of Database Systems", Benjamin/Cummings 1994.

[3] GultekinOzsoyoglu, Richard T. Snodgrass, “Temporal and Real-Time Databases: A Survey”, IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 4, August 1995.

[4] S.K. Gadia, “A homogeneous relational model and query languages for temporal databases,” ACM Trans. Database Systems, vol. 13, no. 4, pp.418-448, Dec. 1988.

[5] Nina Edelweiss, Patricia N. Hubler , Mirella M, Moro, and Giovani Demartini, “A Temporal Database Management System Implemented on Top of a Conventional Database”, IEEE International Conference of the Chilean, pp. 58-67, November 2000.

[6] C.S. Jenson and R. T. Snodgrass, “Temporal Data Management”, IEEE Transactions on Knowledge and Data Engineering, 11(1):36-44, January/ February 1999.

[7] DebabrataDey, Terence M. Barron, and Veda C. Storey, “A conceptual model for the logical design of temporal databases”, Decision Support Systems 15 (1995), pp. 305-321, Elsevier Science B.V 1995.

[8] H. Gregersen and C. S. Jensen, “Temporal Entity-Relationship Models-A Survey”, IEEE Transactions on Knowledge and Data Engineering, Vol. II, Issue 3, pp. 464-497, 1999.

[9] J’anM'at'e and Ji’r’i ' Safa'r'lk, “Transformation of Relational Databases to Transaction-Time Temporal Databases”, IEEE Second Eastern European Regional Conference on the Engineering of Computer Based Systems, 2011.

[10] S.B. Navathe and R. Ahmed, “A temporal relational model and a query language;’ Information Sciences, vol. 49, pp. 147-175, 1989.

[11] AviSilberschatz, Henry F. Korth, and S.Sudarshan, “Database System Concepts, Sixth Edition”, January 2010.

[12] Vincent S. Lai, Jean-Pierre Kuilboer and Jan L. Guynes, “Temporal Databases: Model Design and Commercialization Prospects”, DATABASE, Vol. 25, No. 3, August 1994.

[13] G. Ariav, “A temporally oriented data model,” ACM Trans. Database Systems, vol. 11, no. 4, pp. 499-527, Dec. 1986.

[14] Shailender Kumar, Rahul Rishi, “Retrieval of Meteorological Data using Temporal Data Modeling”, Indian Journal of Science and Technology, Vol 9(37), October 2016.

[15] J. Ben-Zvi, “The Time Relational Model,” PhD thesis, Computer Science Dept., UCLA, 1982.

[16] Christian S. Jensen, Richard T. Snodgrass, Michael D. Soo, “The TSQL2 Data Model”, IEEE Transaction on Knowledge and Data Engineering, 1994.

[17] A. Segev and A. Shoshani, “Logical modeling of temporal data,” U. Dayal and I. Traiger, eds., Proc. ACM SIGMOD International Conf. Management Data, pp. 454-466, San Francisco, May 1987.

[18] CananAtay, “An attribute or tuple timestamping in bitemporal relational databases”, Turk J Elec Eng& Comp Sci (2016) 24: 4305 – 4321.

[19] Shailender Kumar, Rahul Rishi, “A relative analysis of modern temporal data models”, 3rd International Conference on Computing for Sustainable Global Development, pp. 2851-2855, March 2016.