Temporal Databases – A Review

Ekta[1], Ankit Sharma[2], Chinu[3]

[1]Assistant Professor, C.S.E, Baba Farid College of Engineering and Technology, Bathinda, Punjab.
[2]Assistant Professor, C.S.E, Baba Farid College of Engineering and Technology, Bathinda, Punjab.
[3]Assistant Professor, C.S.E, Baba Farid College of Engineering and Technology, Bathinda, Punjab.

Received: April 08, 2018 Accepted: May 02,2018

ABSTRACT *Temporal databases contain data varying corresponding to time. In the real time database, every transaction have deadlines or constraints wrt time. Organizations are now store achieve of data to help them in making important business decisions. The Commercial Database Management system that is available in the I.T. sector do not provide any considerable and functional method of storing and manipulate this type of data. This paper reviews the extensive research in these two formerly separate areas. The paper aims to discuss the Temporal Database Management System (TDBMS) that provide way to manipulate temporal data*

Keywords – *Temporal databases, temporal data.*

INTRODUCTION

Temporal database stores the data involving to time constraint. It provides a wide range of temporal data types and stores information relating to time variations - past, present and future. for example -the history of the stock market or the movement of employees within an organization. Thus, a temporal database stores a collection of time related data.

Temporal database is created by collecting and storing temporal data. The main difference between the temporal and non-temporal data is a time constraint is appended with data representing when it is applicable or stored in the database. Data stored in conventional databases believe data should be valid at current time as for time instance "now". When the data in temporal databases is updated, deleted or inserted, the state of temporal database will be overwritten to outline a new state. The state aforementioned to any changes to the database will no longer available after updating the data. Thus, by correlating time with data, it is feasible to store the dissimilar database states [2].

Temporal data is created by time-stamping the normal data. In relational data model, tuples (rows) are time stamped and in the object oriented data model, objects (attributes values) are time stamped. In the relational data model, relations (Tables) are extended for two additional attributes in which one is for start time and second is for end time [4].

NEED FOR TEMPORAL DATABASE

In the last decades, the relational data model is becoming popular because of its ease and concrete mathematical foundation. Though, the relational data model was proposed by E. F. Codd does not address the temporal element of data. Variation of data over time is treated in the same way as normal data. This is not acceptable for applications that need present, past or future data which is to be dealt by the database, In real time such applications abound. In fact, most application needs temporal data to a certain level [1].

DIFFERENT FORMS OF TEMPORAL DATABASE

Time can be considered as valid time or transaction time .

A. Historical Database

A historical database keeps data values with respect to applicable time. The history of the records updated time to time is maintained in the historical database [2].

B. Rollback Database

A Rollback database keeps data values corresponding to transaction time. It maintain the regularity if the transaction failed and system roll backs to the initial state [2].

C. Bitemporal Database

A bitemporal database keeps data corresponding to both applicable and transaction time. Bitemporal database design is a method of keeping data values dependent on time to represents the history of data values and the history of change to the values in the database [2].

TABLE FORMS

A. Snapshot Table

A Snapshot Table is the non temporal relation showing only current status of the database.

TABLE I
SNAPSHOT TABLE

Emp_no	Salary	Name
Em_52	20000	Smith
Em_53	15000	Jones

B. Valid Time Table

The valid time of a database object is the time in which the object is efficient or hold (is true) in reality. The time in which the event occurred, took place in reality. For example, in a company the Salary of the employees have a valid time and end time [1].

TABLE II :VALID TIME TABLE

Emp_no	Salary	Name	Valid Start Time	Valid End Time
Em_52	11000	Smith	01-01-1990	31-03-1995
Em_52	15000	Smith	1-04-1995	31-12-2000
Em_52	20000	Smith	01-01-2000	31-06-2005
Em_53	10000	Jones	01-01-1990	31-12-1995
Em_53	15000	Jones	01-01-1996	31-12-2000

C. Transaction Time

Database object is stored in a database at numerous point of time. The transaction time for an object is the time in which the object is stored in the database, the time that it is present in the database. Transaction time is the time period during which a fact is stored in the database. Imagine a temporal database storing data about the 18th century. The valid time of these facts is somewhere between 1701 and 1800, whereas the transaction time starts when we insert the facts into the database, say, January 21, 1998. For example, In a banking system, the transaction time of a withdrawal would be form the time the clerk entered the payment of withdrawal into the database to the time that it was made invalid in the database [3].

TABLE III
TRANSACTION TIME TABLE

Customer	Amount	Name	Transaction Start Time	Transaction End Time
Cu_52	11000	Smith	01-01-1990	31-03-1995
Cu_52	15000	Smith	1-04-1995	31-12-2000
Cu_52	20000	Smith	01-01-2000	31-06-2005
Cu_53	10000	Jones	01-01-1990	31-12-1995
Cu_53	15000	Jones	01-01-1996	31-12-2000

BENEFITS OF TEMPORAL DATABASE

Temporal database has wide spread application area [3]:

1) It's easy to deal with temporal data.
2) Record the data changed with time is more convenient.
3) An object description can be well defined without fragmentation.
4) Having relation model to describe temporal data.
5) Having query algebra to deal with temporal data.
6) It works that to handle static data (without time dimension) in temporal database.
7) The traditional database algebra still work in temporal database.

8) The new query algebra to control time dimension is similar to traditional database algebra.
9) SQL queries are much simpler.
10) Much less procedural code is needed.
11) Schemas are less clustered.

APPLICATION DOMAINS OF TEMPORAL DATA

Examples of application domains dealing with temporal data are [2] :

1) Financial Applications – history of stock markets, share prices.
2) Reservation Systems – e.g. when was a flight booked.
3) Medical Systems – e.g. patient records.
4) Computer Applications – e.g. history of file back ups.
5) Archive Management Systems – e.g. sporting events, publications and journals.

GOALS OF TEMPORAL DATABASES

1) Identification of an appropriate data type for time.
2) Prevent fragmentation of an object description.
3) Provide query algebra to deal with temporal data.
4) Compatiable with old database without temporal data.

CASE STUDY

Personnel management of data in a traditional database, say table employee is saved in the database with the following attributes.

Employee(Name, Salary, Title, BirthDate DATE)

TABLE IV

EMPLOYEE TABLE

Name	Salary	DOB	DATE

It is easy to know the salary of an employee as:
SELECT Salary FROM Employee WHERE Name = 'John'
It is also easy to know the date of birth of an employee SELECT BirthDate FROM Employee WHERE Name = 'John'.
But what if want to know the history of the employee's salary. Then the existing database is converted into Temporal database.table in the temporal database is stored as:

TABLE V

EMPLOYEE TABLE WITH TIME STAMP

Name	Salary	DOB	FROM_DATE	TO_DATE
john	60,000	12/0380	01/01/95	30/06/95
john	70,000	12/03/0	01/07/95	31/12/95
john	80,000	12/03/80	01/01/96	30/06/96
john	90,0000	12/03/80	01/07/96	31/12/96

MULTIPLE TEMPORAL DIMENSIONS

Till 1995 there was only single-dimensional temporal databases. Temporal relations were allowed only a single temporal attribute. To motivate the introduction of multiple temporal dimensions there are two categories.

A. Bitemporal databases

Each tuple in a relation two kinds of time are stored—the valid time (when a particular tuple is true) and the transaction time (when the particular tuple was inserted/deleted in the database) [4].

B. Spatial databases

Multiple dimensions over an interpreted domain can be used for representing spatial data where multiple dimensions serve as coordinates of points in a k dimensional space [5].

Most of the data modeling techniques require only fixed-dimensional data. However, the true need for arbitrarily large dimensionality of data models originates in the requirement of having a first-order complete query language. Thus, there are two cases to consider : Temporal models with a fixed number of dimensions, and temporal models with a varying number of temporal dimensions without an upper bound.

The representation of multiple temporal dimensions in abstract temporal databases is quite straightforward. Here simply index relational databases by the elements of an appropriate self-product of the temporal domain (in the case of snapshot temporal databases), or add the appropriate number of temporal attributes (in the case of timestamp temporal databases).

DESIGNING TEMPORAL DATABASES

The design of appropriate database schemas is critical to the effective use of database technology and the construction of effective information systems that exploit this technology. Database schemas capturing time-referenced data are often particularly complex and thus difficult to design. The first of the two traditional contexts of database design is the data model of the DBMS to be used for managing the data. This data model, generally a variant of the relational model, is assumed to conform to the ANSI/X3/SPARC three-level architecture. In this context, database design must thus be considered at each of the view, logical, and physical levels. In the second context, a database is modeled using a high-level, conceptual design model, typically the Entity-Relationship model. This model is independent of the particular implementation data model that is eventually to be used for managing the database, and it is designed specifically with data modeling as its purpose, rather than implementation or data manipulation, making it more attractive for data modeling than the variants of the relational model. Mappings are assumed available that bring a conceptual design into a schema that conforms to the specific implementation data model of the DBMS to be used [5].

A. Logical Design

In temporal databases, there is an even greater need for database design guidelines. However, the conventional normalization concepts are not applicable to temporal relational data models because these models employ relational structures different from conventional relations. New temporal normal forms and underlying concepts that may serve as guidelines during temporal database design are needed. In response to this need, a range of temporal normalization concepts have been proposed, including temporal dependencies, keys, and normal forms. Consider the Checked Out relation schema, Applying conventional dependencies directly, the answer to both questions is no. (The possible answers are 'no' and 'perhaps' as we are considering relation instances.) The second representation is so different from a regular relation that it makes little sense to directly apply conventional dependencies. The relation rules out any of the dependencies when we apply regular dependencies directly. Considering that the different representations of the Checked Out relation model the same mini world and are capable of recording the same information, it may reasonably be assumed that these different representations would satisfy the same dependencies. At any point in time, a customer may have checked out several tapes. In contrast, a tape can only be checked out by a single customer at a single point in time. With this view, TapeNum temporally determines CustomerID, but the reverse does not hold [3].

B. Conceptual Design

It is widely known that the temporal aspects of the mini-world are very important in a broad range of applications, but are also difficult to capture using the ER model. Put simply, diagrams that would be intuitive and easy to comprehend without the temporal aspects become obscure and cluttered when an attempt is made to capture the temporal aspects. Modeling the mini-world at a single point in time. Attempting to capture the temporal aspects that are essential for this application complicates matters. It is necessary to capture the time wen a customer has checked out a video tape. And since it is possible for the same customer to have checked out the same tape at different times, the CustomerID and TapeNum attributes do not identify a single instance ofCheckedOut. Instead, it is necessary to make CheckedOut a ternary relationship type, introducing a new, somewhat artificial, entity type that captures the times of rentals. Including start and end time attributes on this entity type, the CustomerID, TapeNum, andStartTime attributes identify instances ofCheckedOut [3].

But simply requiring that these three attributes be a key of a relation representing rentals does not ensure the integrity of this relation—it remains possible for the same tape to be checked out more than once at the same point in time. As another issue, rental prices may vary over time, e.g., due to promotions and films getting old. Finally, including transaction time leads to further complications. As a result, some industrial users simply choose to ignore all temporal aspects in their ER diagrams and supplement the diagrams with textual phrases to indicate that a temporal dimension to data exists, "full temporal support." The result is that the mapping of ER diagrams to relations must be performed by hand and the ER diagrams do not document fully the temporally extended relational database schemas used by the application programmers.

TEMPORAL DBMS IMPLEMENTATION

There has been a vast amount of work in storage structures and access methods for temporal data and a dozen-odd temporal DBMS prototypes have been reported .

Two basic approaches may be discerned. Traditionally, an integrated approach has been assumed, in which the internal modules of a DBMS are modified or extended to support time varying data. More recently, a layered approach has also received attention. Here, a software layer interposed between the user-applications and a conventional DBMS effectively serves as an advanced application that converts temporal query language statements into conventional statements that are subsequently executed by the underlying DBMS, which is
itself not altered. While the former approach ensures maximum efficiency, the latter approach is more realistic in the short and medium term. Consistent with the vast majority of temporal DBMS implementation, this section assumes an
integrated approach utilizing time stamping of tuples with time intervals, unless explicitly stated otherwise [5].

A. Querying Temporal Database

In order to query a temporal database, the introduction of the temporal relational algebra (TRA) is considered. Defining new temporal operators such as Until and Since or using current, existing operators of the relational algebra (RA), need to be either defined or revised to manipulate temporal data (especially taking care of the time intervals) and extract correct information from a temporal database. When querying a relation R in the temporal database to find tuples that holds for time q, we need only ask the relational database (to which the temporal structure has been applied the relational query.

$\sigma_{start \le q \le end}\, R(a1,\ldots,an,start,end)$ where σ is the select operator [5].

CONCLUSION

This paper has briefly introduced to temporal data management, describing the challenges faced. A great amount of research has been conducted on temporal data models and query languages, which has shown itself to be a complex challenge issues.

The semantics of standard temporal relational schemas are well understood, as are the implications for database design. Many languages have been proposed for querying temporal databases, half of which have a formal basis.

REFERENCES

[1] H. Gregersen and C. S. Jensen, "Temporal Entity-Relationship Models—A Survey" *IEEE Transactions on Knowledge and Data Engineering*, vol.3, pp.464–497, 1999.

[2] R. Elmasri and S. Navathe,*Fundamentals of Database Systems,* 5th ed. pp.413-434, 1994.

[3] J. Celko. Joe Celko, *Data and Databases: Concepts in Practice*, pp.567-597, Morgan Kaufman Publishers,1999.

[4] J. Clifford and A. Tuzhilin, "Recent Advances in Temporal Databases" Proceedings of the International Workshop on Temporal Databases., Workshops in Computing Series. Springer-Verlag, 1995.

[5] C. S. Jensen and R. T. Snodgrass, *Temporally Enhanced Database Design,*1995.