Querying Temporal Database System for University Staff Database

Su Lei Phyu
University of computer studies, Yangon
suleiphyu.ucsy@gmail.com
starfish.su63 @gmail.com

Abstract

A temporal database is a database with built-in time aspects or a temporal data model and a temporal version of structured query language. More specifically the temporal aspects usually include valid time and transaction time. These attributes go together to form bi-temporal data. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact is stored in the database. Bi-temporal data combines both Valid and Transaction Time.

In this system, university staff database management system using temporal database is used for university staff transfer, promotion, education, training, experience, salary and position history need to be maintained. The main task of the system is to record the time varying data about university staffs database. University has different departments and different roles of staff. This staff has owned different positions and different salary on time varying. This system can maintain and analyze the staff's history, position status and salary. Thus, the executive of the University can review the staff's condition accordingly with the staff transfer and promotion.

Keyword: Bi-temporal data, temporal database, valid time, transaction time.

1. Introduction

A database is an organized repository for data. A database that contains current data only is termed as a snapshot database in which data is deleted or updated when the facts represented by that data cease to be true, For example, consider the University Staff Database, in which case, one of the staff is currently Lecturer. From the view of temporal version, by contrast, it might show not only that the rank is currently Lecturer, but also that he has been that rank ever since May 5th 2006.

A temporal database is formed by compiling, storing temporal data. The difference between temporal data and non-temporal data is that a time period is appended to data expressing when it was valid or stored in the database. The data stored by conventional databases consider data to be valid at present time as in the time instance "now". When data in such a database is modified, removed or inserted, the state of the database is overwritten to form a new state. The state prior to any changes to the database is no longer available. Thus, by associate time with data, it is possible to store the different database states.

In essence, temporal data is formed by time-stamping ordinary data (type of data we associate and store in conventional databases). Each ordinary data has two time values attached to it, a start time and an end time to establish the time interval of the data. In a relational data model, relations are extended to have two additional attributes, one for start time and another for end time. Time can be interpreted as valid time or transaction time. A *historical database* stores data with respect to valid time. A *rollback database* stores data with respect to transaction time. A *bitemporal database* stores data with respect to both valid and transaction time they store the history of data with respect to valid time and transaction time.

2. Related Works

In [3], this paper shows a model for representing changes in semi structured data (DOEM) and a language for querying changes. The querying languages were translated to a language for querying semi structured data, and can therefore be viewed as a stratum approach. In [1], the research area of temporal databases aims to change this state of affairs by characterizing the semantics of temporal data and providing expressive and efficient ways to model, store, and query temporal data.

In [8], temporal Wikipedia explains the temporal data nature and temporal data aspect. It record not only changes in what happened at different times, but also changes in what was officially recorded at different times. A particularly challenging issue is the support of temporal queries in a transaction time database under evolving schema. In [6], this paper shows the historical database with a chronology, touching briefly on all work that awareness of this area. It discuss in some detail what the system consider to be the ten most important papers and events in terms of their impact on the discipline of temporal database.

3. Temporal Database

Temporal data stored in a temporal database is different from the data stored in non-temporal database in that a time period attached to the data expresses when it was valid or stored in the database. As mentioned above, conventional databases consider the data stored in it to be valid at time instant now, they do not keep track of past or future database states. By attaching a time period to the data, it becomes possible to store different database states.

There are mainly two different notions of time which are relevant for temporal databases. One is called the *valid time*, the other one is the *transaction time*. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact is stored in the database. Note that these two time periods do not have to be the same for a single fact. Imagine that we come up with a temporal database storing data about the 18th century. The valid time of these facts is somewhere between 1700 and 1799, where as the transaction time starts when we insert the facts into the database, for example January 21, 1998.

Then, the following table could result:

Emp ID	Name	Dept	Salary	VTstart	VTEnd
10	U Hla	SW	11000	1985	1990
10	U Hla	APP	11000	1990	1993
10	U Hla	APP	12000	1993	Now
11	Daw Nu	HW	10000	1988	1995
12	Daw Mya	HW	10500	1991	Now

The above valid-time table stores the history of the employees with respect to the real world. The attributes *ValidTimeStart* and *ValidTimeEnd* actually represent a time interval which is closed at its lower and open at its upper bound. Thus, we see that during the time period [1985 - 1990), employee U Hla was working in the Software department, having a salary of 11000. Then he changed to the Application department, still earning 11000. In 1993, he got a salary raise to 12000. The upper bound Now denotes that the tuple is valid until further notice. Note that it is now possible to store information about past states. We see that Daw Nu was employed from 1988 until 1995. In the corresponding non-temporal table, this information was deleted when Daw Nu left the company.

3.1 Temporal Database Forms and Data Model

Another important issue relating to the representation of time is the role played by the time (what the time signifies) that is being supported by the temporal database system. There are two common perceptions of time, transaction time and valid time.

3.1.1 Transaction time

A database object is stored in a database at some point in time. *The transaction time of an object is the time when the object is stored in the database, the time that is present in the database.* For example, in 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. Another example would be, in a company situation, an employee receives a pay rise but it comes into effect when the payroll clerk enters this salary rise into the database. Transaction time values cannot be after the current time.

Often, data cannot be recorded in a database in real time, for example, due to a delay in the processing of information. So there might be a time gap between data being valid in the real world and recording the data in a database. Sometimes, it is necessary to keep track during which time periods facts are stored in a database. This notion of time is called transaction time. A database object is stored in a database at some point in time. The transaction time of an object is the time when the object is stored in the database, the time that is present in the database. It is maintained entirely by the system, and no user is allowed to change them.

Valid time, by contrast, can be viewed as the history of value changes. Recording transaction time can also be viewed as recording the history of corrections. This means that data timestamped with transaction time provides for querying the history of data manipulations and errors, whereas data timestamped with valid time allows the querying of value changes. Values for transaction time cannot be later than the current time, since transaction time reflects the time when a database operation is actually executed. The DBMS itself records transaction time. It also does not make sense to update transaction time, since a database operation of a committed transaction can never be undone. The only way to change a committed database operation is to do an inverse transaction, which, however, is executed at a later time point and thus leads to another transaction time record.

3.1.2 Valid time

The valid time of a database object is the time when the object is effective or holds (is true) in reality. The time when the event occurred, took place in reality. For example, in a banking system, the payments and withdrawals made by a customer have a valid time associated with the time the customer performs the transaction at the bank. Objects in the temporal database system will have a time component associated to it; this will hold either the valid time or the transaction time.

Objects in the temporal database system will have a time component associated to it; this will hold either the valid time or the transaction time. Valid time concerns the time when a fact is true in reality. Valid times can be in the future, if it is known that some fact will become true at a specified time in the future. This notion of time, recording data with respect to when it was, is or will be valid in the real world, is called valid time. A valid time interval thus records the time period when a fact is true. Valid time must be supplied by the user when adding or modifying data. Valid time values can be updated.

3.1.3 User defined time

User defined times are drawn from a domain of dates and times with an identity relation and a total ordering, i.e., it has an associated less than relation. User defined times may be manually supplied or computed by an application program.

A distinction can be made whether or not the timestamp added to data is interpreted by the DBMS (for example, during query evaluation). An un-interpreted timestamp, for example, a value in an attribute birthday, is called user-defined time, because the user himself interprets the given time information, whereas the DBMS treats this temporal data as just another attribute. Values can be any time instant referring to past, present or future time points. User defined time values are supplied by the user and may be updated.

3.2. Bitemporal Operation

A temporal database (or a bitemporal database) combines both a historical database and a rollback database, so it has both valid-times and transaction times. This allows you to make queries about historical dates as they were believed to be at some point in the past.

A bi-temporal database stores the history of data with respect to both valid time and transaction time. It is noted that the history of when data was stored in the database (transaction time) is limited to past and present database states, since it is managed by the system directly which does not know anything about future states. So, a bi-temporal database is a combination of a historical and a rollback database. A bi-temporal database thus has the properties of both historical and rollback databases. It is now possible to record updates of valid time in this kind of database. A diagram of bi-temporal database is shown in Figure 1.

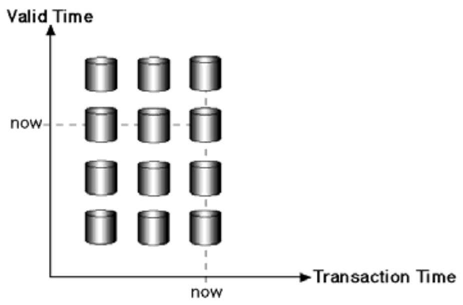

Figure 1. A bi-termporal database in context of valid time and transaction time

3.3 Time Interval Operations

Temporal modal logic represents time by means of modal operators. It is an extension to classical logic that provides the logic of possibilities. Temporal intervals are at the core of the system. Intervals are fundamental and at base not further sub-dividable. Events occur in one temporal interval. When we talk about time in databases, we mean a system that can be linear, branching (linear in the past with multiple paths in the future) or cyclic.

Time can be continuous, analogous to the real numbers, dense (analogous to, e.g. the rational numbers) or discrete. Time can be aggregated into temporal sets, intervals and periods, which are sets of intervals. The building blocks of any time system are the elements that make up the time line. On these time lines, we define an Instant as a time point on an underlying time axis. Likewise, a Time Interval (TI) is the time between two instants.

Since intervals are represented as pairs of time-points, comparisons between intervals are based on time-point comparisons of the upper and lower bounds. The interval comparison operators are BEFORE, AFTER, DURING, CONTAINS, OVERLAPS, MEETS, STARTS, FINISHES, and EQUAL.

Let *I1, I2* be two intervals, and *begin(I), end(I)* be respectively the lower bound and upper bound of the interval I. The definitions of 13 interval comparisons are given in Table 1.

Table 1. Comparison Operators Meaning

1	I1 before I2	$I1_E < I2_S$
2	I1 after I2	$I2_E<I1_S$
3 4	I1 during I2 I1 contains I2	$(I1_S>I2_S \wedge I1_E \leq I2_E) \vee$ $(I1_S \geq I2_S \wedge I1_E < I2_E)$ $(I2_S>I1_S \wedge I2_E \leq I1_E) \vee$ $(I2_S \geq I1_S \wedge I2_E < I1_E)$
5 6	I1 overlaps I2 I1 overlapped_by I2	$I1_S < I2_S \wedge I1_E > I2_S \wedge$ $I1_E < I2_E$ $I2_S < I1_S \wedge I2_E > I1_S \wedge$ $I2_E < I1_E$

7
8
	I1 meets I2
	I1 met_by I2
		I1E = I2S
		I2E = I1S

9
10
	I1 starts I2
	I1 started_by I2
		I1S = I2S $\wedge$ I1E<I2E
		I1S = I2S $\wedge$ I2E<I1E

11 12	I1 finishes I2 I1 finished_by I2	I1S > I2S $\wedge$ I1E=I2E I2S > I1S $\wedge$ I1E=I2E
13	I1 equivalent I2	I1S = I2S $\wedge$ I1E=I2E

4. Overview of the proposed system

This system is the development of university staff information system for computer universities. In computer universities, all of staffs are different positions, different department, different salary by time varying. The universities staffs are attend the different training, teach the different subjects, accept the different transfer university. Thus, this system can view the current information of the required staff and can analyze the different places and roles of staff.

This system mainly maintains the personal data in current condition. The user can view the staff personal detailed data and current university, current rank, current department and current pay scale. In addition, this system also maintains the training, transfer and subject operations. In these operations include valid-start and valid-end attributes. These attributes are actually represent a time interval which is closed at its lower and open at its upper bound.

In our proposed system, user can do insert new staff, search by staff number for valid staff and searching staff information by user request. In insert new staff information, if the inserted staff information is not completed, the system shows the page with required field for user input. When the user input completes, the insertion of new staff case is successful and will show these staff information.

In search by staff number for only valid staff, if the searched staff ID is valid until now then the system will show that staff, otherwise the system will show no selected staff. In searching staff information by user request, the system will show the staff information by user requested. If the staff is valid in the system, that valid staff can do update or delete operation. All these insert, update, delete and select operations of information are processed to temporal database and then display the result of staff detailed information. Final, user can get the historical information of staff such as the movement, promotion, change salary rate and retire of staff form University.

This system can use temporal operators such as before, after, during, contains, meets, starts, finishes, equivalent. If user wants to see the staff training data before specified date, the system shows the required staff information. The user can see all staff information about staff training, transfer and teaching subject by user requested time.

The method and apparatus also intercepts other commands to change a temporal database, such as those similar to conventional SQL "DELETE" and "INSERT" commands, and generates conventional SQL commands to perform delete and insert functions using a subset of the techniques used to perform updates described above.

Our temporal application was developed as the ways describe above, by using a relational database system to model and store temporal relations and hence, produces a temporal database. This database application will demonstrate temporal features and aspects to users.

Figure 2 shows the system flow diagram of our proposed system.

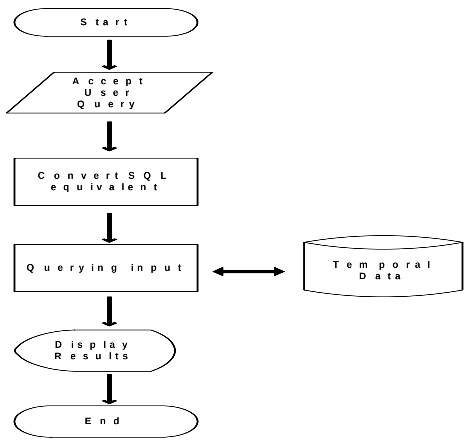

Figure 2: System flow diagram

4.1 Database Design Of the proposed system

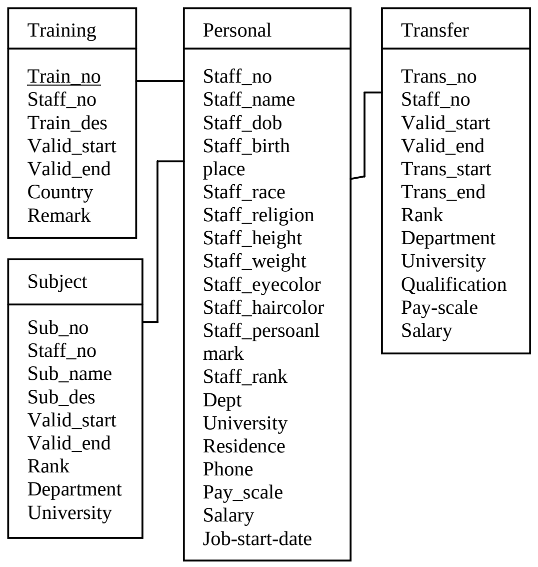

Figure 3. Staff Database Design

In our database, we have four tables, one is personal which is to record staff's personal data and the other three is staff transaction tables. There are Training, Transfer and Subject. Figure 1 shows Staff database design. Since Personal table has no temporal operation, common query statements can be used for this table.

The temporal operation will be done on three transaction tables. The Training transaction table has seven attributes such as Train_no, Staff_no, Train_des, Valid_start, Valid_end, Country and Remark. Among these attributes, Train_desc and Country are distant transactional attributes for our application. The department attribute has nine dimensions such as Myanmar, English, Information Science, Computational Mathematic, Computer Software, Computer Hardware, Computer Application, Research and Development and Operation and rank attribute has six dimensions such as Demonstrator, Tutor, Assistant Lecturer, Lecturer, Associate Professor, Professor.

As a first step in introducing the topic of the paper, we briefly describe bi-temporal data. This type of data has associated a valid time, indicating when the data was true in the modeled reality, and a transaction time, indicating when the data along with its valid time was stored as current in the database.

The valid time of a tuple, a period, may be recorded using the two attributes Valid start and Valid end.

When database designers actually understand the core temporal database concepts, perhaps most prominently valid and transaction time, they are able to design better databases using existing models and tools. A central challenge is to provide complete conceptual models, with associated design tools, that cover all aspects of designing a temporal database; empirical evaluation of these by real users is needed to provide essential insights.

The cardinality (number of specific values) of an attribute is less useful than the average cardinality at a point in time. Another useful statistic is the number of *long-lived tuples*, the presence of which is the bane of some index structures and temporal algebraic operators. The area of *active databases*, rules responding to database changes and external events are a focus. These may be extended to take into account prior history and temporal trends.

4.2 Querying Staff Temporal Database

In this system, the user can use the temporal operators such as before, after and during. For example, the user wants to see the all of the staff's transfer before 1.1.2000, than the system will show the transfer staff before 1.1.2000.

Before

Q1 : Get subject name for Daw Ni Ni who are teaching at IS department before2009.01.02.

SQL:

```
SELECT subject.sub_name
FROM subject LEFT OUTER JOIN personal ON subject.staff_no = personal.staff_no
WHERE subject.valid_end <= '2009-01-02' AND personal.staff_name LIKE 'Daw Ni Ni' AND subject.department LIKE 'IS'
```

After

Q2 : Get staff name who are transfer with Assistant Lecturer position to UCSM after 2000-09-16.

SQL:

```
SELECT personal.staff_name
FROM transfer LEFT OUTER JOIN personal ON transfer.staff_no = personal.staff_no
WHERE transfer.valid_start >= '2000-09-16' AND transfer.staff_rank LIKE 'AL' AND transfer.university LIKE 'UCSM'
```

During

Q3 : Get transfer university for Daw Mi Mi Khine during 5 Years in 1999-05-21 to 2004-05-21.

SQL:

```
SELECT transfer.university
FROM transfer LEFT OUTER JOIN personal ON transfer.staff_no = personal.staff_no
WHERE transfer.valid_start > '1999-05-21' AND transfer.valid_end <= '2004-05-21' AND personal.staff_name LIKE 'Daw Mi Mi'
```

5. Conclusion

Finally, since this system implements the temporal operation on MySql database server, the temporal query statements will be needed to substitute with non_temporal query statements which are supported by MySql.

In this system, we can make operation of transfer, promotion, education, training, experience, salary and position and view the required operations of the staff. We can't use real temporal operators and query language. We only implement temporal aspect by using relational database architecture. This system can use user required operators like temporal nature.

Reference:

[1] Christian S, Jensen, Introduction to Temporal Database Research

[2] G. Hamilton and R. Cattell. JDBC: A Java SQL API version 1.20.JavaSoft, 1997.

[3] I. Ahn and R. T. Snodgrass. Partitioned Storage for Temporal Databases. Information Systems, 13(4):369–391, 1988.

[4] J. Gray and A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, 1993.

[5] J. Patel, Department of Computing, Imperial College, University of London, Temporal Database Systeem, 2003.

[6] M. H. Böhlen. Temporal Database System Implementations. SIGMOD Record, 24(4):53–60, 1995.

[7] R. T. Snodgrass. The Temporal Query Language TQuel. ACM TODS, 12(2):247–298 (1987).

[8] Temporal database-Wikipedia, the free encyclopedia