Temporal Database: An Approach for Modeling and Implementation in Relational Data Model

Ab Rahman Ahmad[1], Nashwan AlRomema[2], Mohd Shafry Mohd Rahim[3], Ibrahim Albidewi[4]

[1,4]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
[2,3]UTM-IRDA Digital Media Center, Faculty of Computing, Universiti Teknologi Malaysia, Malaysia.
[1]hayinsuh@yahoo.com, [2]nashwan.alromema@gmail.com

Abstract: Conventional relational data models deals with current data and can only represent the knowledge in static sense, while temporal data representations have a dynamic domain and resources on the database systems which are well described in literature. This study introduces an approach for implementing temporal database applications with interval-based timestamp in conventional Non-temporal Database Management Systems (DBMS). The proposed approach can be easily implemented in relational framework as well as having the representational power of the modeling and querying power of 1NF relational data model.
[Ahmad A, Alromema N, Rahim MS, Albidewi I. **Temporal Database: An Approach for Modeling and Implementation in Relational Data Model.** *Life Sci J* 2015;12(3):105-109]. (ISSN:1097-8135). http://www.lifesciencesite.com. 14

Keywords: Temporal Database, valid-time data model, transaction time data model.

1. Introduction

Temporal Database is considered as repositories of time-dependent data. There has been a vast amount of work regarding developing temporal database applications starting from the 1970s (Findler, 1971). Some of these works deal with storage structure, and query processing as well as a dozen-odd temporal DBMS prototype (Date et al., 2003), (Snodgrass, 2000), (Novikov and Gorshkova 2008), (Jensen et al., 1994), (Tansel, 2004), (Elmasri and Navathe, 2000), and (Jensen et al., 1994). Some of temporal database applications have been used in spatiotemporal database for managing movement data in GIS as in (Rahim et al., 2007), (Rahim et al., 2008), and (Man et al., 2011). Conventional relational database is used to store and process the data that refer to the current time (Date et al., 2003). There are two basic approaches in developing temporal database application, the first one is an integrated approach where the internal models of DBMS are modified or extended to support time-varying aspects of data, and the second approach would be the stratum approach in which a layer over DBMS converts temporal statements in to conventional DBMS and converts the result from the DBMS to be in the temporal form (Patel, 2003). While the first approach ensures the maximum efficiency, the second approach is more realistic and more popular. The major contributions of this research study can be formulated as follows:-

- To describe the meaning and use of temporal features in the framework of relational data model.
- To incorporate temporal aspects need to minor modifications without affecting the performance of the parts of the system that do not use temporal data.
- To represent the temporal database in a data model that has expressive power and less storage memory comparing to other works especially in (Novikov and Gorshkova, 2008), and efficient query processing.
- To provide an implementation technique that is easy, does not cost much, and based on relational database not on XML files as in (Wang et al., 2006).

The discrete time model is considered as the time model for representing temporal database because of the simplicity and relative ease of implementation (Patel 2003). The time aspect used in temporal database can be interpreted as **User-defined time,** which is defined as the column that just happens to be of a date/time data type (an example the Birth Date column), and does not indicate anything related to the validity of other columns, or **Temporal time,** in which the column(s) that are of date/time data type are used to indicate time aspects of the associated tuple. **Temporal time** is categorized into **Valid-time:** in which the associated time, is used to indicate when certain fact occur or when it is considered true in the real world (Bohlen et al, 1998)**. Transaction-time:** the associated time refers to the time when the information was actually stored in the database. **bitemporal-time:** Associated time refers to both valid-time and transaction-time yield in bitemporal data model. Rollback database views tuples as begin valid at sometimes as of that time (Snodgrass, 2000), (Jensen et al., 1994).

2. Material and Methods

The appropriate design of database schemas is critical to the effective use of database technology, and the construction of effective information systems that exploit this technology. Designing database

systems is typically considered in three contexts (1) Conceptual design using high-level conceptual data model, (2) Logical design using implementation (representational) data model like relational data model, and finally (3) Physical design using the appropriate Database Management System (DBMS) to ensure the desired performance. The temporal aspects of database schemas are often complex and thus difficult and error-prone to design. In designing temporal database, the same steps as the mentioned above can be followed, in addition to that, defining new features concerning the time aspects, because both conventional conceptual model, and relational data model do not fully support time-varying aspects. The following steps summarize the proposed methodology for designing temporal database in relational database.

- Designing the conceptual model for the business logic of the system and map it into conventional relational data model using the mapping methodology described in (Snodgrass, 2000) and (Elmasri and Navathe, 2000), where all temporal aspects that need to be modeled are ignored at this step. All conventional methods which are used to construct good relational database schema by analyzing the design and applying different forms of normalization should take place in this step.
- Adding the temporal aspect for all the database objects that need to keep the historical changes of the entities' data.

3. Results

To make the process of our methodology clear, an example of the conceptual model shown in Figure 1 for *STUDENT* and *COURSE* relations are mapped to relational data model shown in Figure 2. Our methodology for representing temporal database is accomplished by, firstly, defining the database object (entity/relation) for which we want to track the historical of changes of the stored data, then we add for each such relations two additional columns Lifespan Start Time (*LSST*) and Lifespan End Time (*LSET*), which indicate the beginning and the end of the time interval within which the database object exists in the modeled reality (Jensen et al., 1994). Secondly, for each such entity/relation, we create an additional relation with the same name as in the basic schema with the suffix *VT*. We use *VT* to indicate the valid time data model.

As an example, the relational database table *STUDENT* in Figure 2 is represented into temporal database (Figure 3) by adding two additional columns *LSST* and *LSET*, after that we create a new table *Table_VT*, for example *STUDENT_VT* as $STUDENT\_VT = (St\_no, index, Update\_A, VST, VET)$.

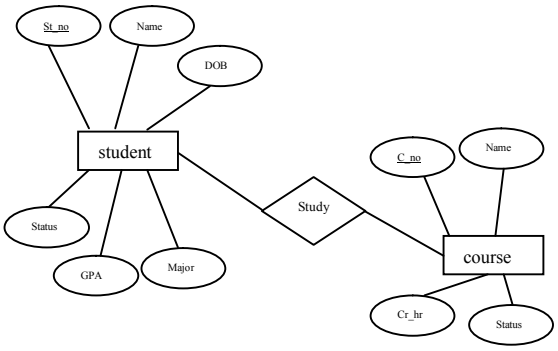

Figure 1: An example of non-temporal conceptual schema.

The conventional mapping of conceptual data model in Figure 1 to logical data model is shown below.

STUDENT

St_no	Name	DOB	Status	GPA	Major

COURSE

C_no	Name	C_ hours	Status

STUDY

St_no	C_no

Figure 2: Relational Schemas for Student and course

Adding the temporal aspects to these two relations is shown below as in Figure 3.

STUDENT 3 4 5

St_no	*Name*	*DOB*	*Status*	*GPA*	*Major*	*LSST*	*LSET*
8090	Jon	1/1/90	Active	3.4	CS	201302	300001
8091	Alex	2/1/89	Graduate	3.90	IS	201001	201403

STUDENT_VT

St_no	*Index*	*Updated_V*	*VST*	*VET*
8090	4	3.56	201001	201003
8090	4	3.80	201101	201401
8090	3	Active	201001	201403

COURSE 1 2 3

C_no	*Name*	*Cr_hour*	*Status*	*LSST*	*LSET*
CS201	Programming I	3	Active	201001	300001
CT222	Database	4	Active	200901	300001

STUDENT_VT

C_no	*Index*	*Updated_V*	*VST*	*VET*
CS201	1	C++ prog.	201001	201201
CT222	2	3	200901	201101
CS201	2	4	201001	201201

Figure 3: Relational Schemas for Student and Course.

The columns are identified as follows: *St_no* is the key attributes in the basic schema, *index* attributes is used to identify the updated attributes, *upadated_v* is used to store the old value of the updated attributes in the basic table, and *VST, VET* is used to represent the beginning and the end of the time interval within which the values in the specific updated attribute were valid.

As shown in Figure 3, *Table_VT* has primary key consists of the primary key of the basic table, *index* column, and the *VST* column. For *STUDENT_VT* table, the primary key is (*St_no, index,VST*). The primary key of the basic table in *Table_VT* serves as the foreign key in the *Table_VT*, Foreign key(St_no)references STUDENT(St_no). The data in the basic table keeps the latest updated data (current data), whereas *Table_VT* stores the historical changes of the validity of the updated attributes in basic table.

4. Discussions

Modification operations are considered as challenges when applied to time-varying data because of the time dimension attached to this data (Tansel, 2006). In our representational data model we consider the insertion, deletion, and update of records in the table of the basic schema, the data in the *Table_VT* are updated automatically using database triggers or application's function. The following are the rules of data modification operations:

- Insertion Operation: inserting a new record into a table of the basic schema is accomplished as in conventional database, in addition to that the value of *LSST* field is set to the current date, and the value of the *LSST* field is set to a very far future time, for example, 300001. This date is always greater than the current date for the lifespan of the application. Inserting data into *Table_VT* is accomplished as consequences of updating any attribute in the table of the basic schema as it will be explained in updating operation. Thus, the data in the table of the basic schema always represents the latest current valid data.
- Updating Operation: updating a record in a table of the basic schema results into the following actions: 1- If the updated data is an indexed attribute(s) as shown in Figure 3, then the old value of this attribute and its index with the same value of the primary key and *VST* and *VET* fields are inserted into *Table_VT*, the values of *VST* and *VST* can be calculated as follows: (a) if this is the first time to update this attribute (this attribute has not been updated before or no record for this attribute is found in *Table_VT*), then *VST* value will have the same value as *LSST* in the table of the basic schema, and *VET* will be having the value of the current time. (b) If this attribute has been updated before, then *VST* will be having the value of *VET* plus one time granule of the latest update of this attribute. An example of this case is shown in Figure 3, when the value of the GPA attribute indexed by 4 has been updated (at time point 201003) for *Jon*, and then we look at *Table_VT* at that time point, since no record has been found for this attribute and for this object, thus a new record for the updated value of this attribute and corresponding database object has been inserted into *Table_VT* table with these values $(St\_no:8090, index:4, Update\_A:3.56, VST:201001, VET:201003)$ 2- If the updated data is *LSET* attribute with instance time not equal to 300001, then this action is considered as logical delete of this record and this record stops to be a life or valid, as if one student has finished/graduated or terminated from the university.
- Delete Operation: delete a record form the basic schema is accomplished by setting the value of *LSET* to current time as explained in update operation.

In our proposed schema representation *Table_VT* tables keep the historical changes of the validity of the updated attributes in the basic table. Each record in *Table_VT* represents the validity of the changed attributes in the basic table during the time interval [*VST*, *VET*]. The historical changes of the validity is continuous, the timestamp in *VST* field coincides with the value of *VET* field of the preceding record with the same primary key. Figure 3 shows the schema representation and the update operations on the basic schema tables (*STUDENT* and COURSE) and the temporal tables (*STUDENT_VT*, *COURSE_VT*).

Although the historical changes of data are in temporal schema and the latest current valid data available from the basic schema, our approach is useful for the following reasons:

- Integrity constraints in the basic schema as well as temporal schema can be defined and implemented in DBMS easily without any major update to the existing applications. The purpose of this implementation is to ensure the creation of highly reliable databases.
- The proposed implementation removes data redundancy and satisfied high level of memory storage saving comparing to other implementation techniques discussed by Halawani and Alromema (2010), reducing the redundant data will help to facilitate efficient query execution.
- The tables in the temporal schema is updated only by insert operation when specific attribute in the basic schema table updated, thus the growth of this table depends on the frequency of attributes updates.
- The current valid data in basic schema table helps in efficient query execution because some queries do not need to have temporal data and

temporal-joins involving data from the temporal schema are less efficient than joins of the tables in the basic schema.

4.2 Querying Temporal Data Model

Querying temporal databases can be classified according to the provided time slice into current query, sequenced query, and non-sequenced. Current query provide the current valid data which is located in the basic schema, while temporal sequenced query provide the data that were valid during a certain interval of time, where this data can be obtained from basic schema, temporal schema, or both, depends on the complexity of the query. Non-sequenced queries provide the historical changes of an entities' data regardless of the time constraints.

Current query is an ordinary query which provides current values of the data regardless of the time dimension. We project current queries on the basic table schema where the latest current values are stored for example the query that selects the current *GPA* and Status of a Student is

```
SELECT S.GPA, S.Status
FROM STUDENT S
WHERE S.St_no = 8090;
```

Sequenced query provide the data that were valid during a certain interval of time, and the result of the query is valid-time table unlike current query which returns snapshot state. For example the query that returns the GPA of a Student in a certain point/semester of time or in a certain interval of time is

```
Q1 for point of time t
SELECT SV.St_no, SV.upadated_v
FROM STUDENT_VT SV
WHERE SV.index = 4 and
SV.VST <= t and
SV.VET > t and
SV.St_no = 8090;
```

non-sequenced query provide the historical changes of an entities' data during their lifespan time. The complexity of non-sequenced queries depends on number of tables involved because the intervals in which the selected records were valid must be overlap for different tables. Above queries can be applied for any other temporal information in Student or Course tables. With time, the tracking log query that retains a data for a certain time interval might have a different data in other time interval.

Acknowledgements:

This distinct research was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah under grant no 830-016-D1434. The authors, therefore, acknowledge with thanks DSR technical and financial support.

Corresponding Author:
Dr. Ab Rahman Ahmad
Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: hayinsuh@yahoo.com

References

1. Findler N, Chen D. On the Problems of Time Retrieval, Temporal Relations, Causality, and Coexistence, 1971, Proc. Second Int. Joint Conf. on Artificial Intelligence, London.
2. Date C, Darwen H, Lorentzos N. Temporal data and the relational data model. 2003, San Francisco: Morgan Kaufmann.
3. Wang F, Zhou X, Zaniolo C. Using XML to Build Efficient Transaction-Time Temporal Database Systems on Relational Databases, 2006, Proceedings of the 22nd International Conference on Data Engineering (ICDE'06) 8-7695-2570-9/06, IEEE.
4. Snodgrass R. Developing Time-Oriented Database Applications in SQL, 2000, 1st edition, Morgan Kaufmann Publishers, Inc., San Francisco.
5. Patel J. Temporal Database System Individual Project. 2003, Department of Computing, Imperial College, University of London, Individual Project, 18-June-2003, http://www.doc.ic.ac.uk/~pjm/teaching/student_projects/ jaymin_patel.pdf.
6. Novikov B, Gorshkova E. Temporal Databases: From Theory to Applications, Programming and Computer Software, 2008, Vol. 34, No. 1, pp. 1–6. © Pleiades Publishing, Ltd., 2008.
7. Halawani S, Alromema N. Memory Storage Issues of Temporal Database Applications on Relational Database Management Systems, 2010, Journal of Computer Science 6, (3): 296-304.
8. Tansel A. On handling time-varying data in the relational model. 2004, Journal of Information and Software Technology, Elsevier, 46(2), 119-126.
9. Jensen C, Snodgrass R, Soo M. The TSQL2 Data Model. 1994, Chapter 12: 361-395.
10. Elmasri R, Navathe . Fundamentals of Database Systems. 2000, 3rd edition. Addison Wesley.
11. Jensen C, Clifford J, Elmasri R, Gadia S, Hayes P, Jajodia S. A Glossary of Temporal Database Concepts. March 1994, SIGMOD Record, 23(1).
12. Tansel A. Modeling and Querying Temporal Data, Copyright © 2006, Idea Group Inc.

13. B¨ohlen M, Busatto R, Jensen C, Point- Versus Interval-based Temporal Data Models, ©1998 IEEE.
14. Jensen C, Clifford J, Elmasri R, Gadia S, Hayes P, Snodgrass R, Soo M. A Consensus Glossary of Temporal Database Concepts. March 1994, SIGMOD RECORD, Vol. 23, No.1.
15. Rahim MS, Shariff A, Mansor S, Mahmud A, Daman D. A spatiotemporal database prototype for managing volumetric surface movement data in virtual GIS Lecture Notes in Computer Science. 2007, including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 4707.
16. Rahim MS, Othman N, Daman D. Visualization of surface movement data using tin-based temporal modeling approach, 2008, Proceedings of the 4th IASTED International Conference on Advances in Computer Science and Technology, ACST.2008:339-343.
17. Man M, Jusuh J, Rahim MS, Zakaria M. Formal specification for spatial information databases, 2011, integration framework (SIDIF) Telkomnika, 9(1):81-88.
18. Man M, Rahim MS, Zakaria M, Aezwani W, Bakar. Integration model for multiple types of spatial and non spatial databases. 2012, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. 62 LNICST:95-101. 0.

3/4/2015