Journal of Computer Science 6 (3): 296-304, 2010
ISSN 1549-3636
© 2010 Science Publications

Memory Storage Issues of Temporal Database Applications on Relational Database Management Systems

Sami M. Halawani and Nashwan A. Al-Romema
Faculty of Computing and Information Technology in Rabigh,
King Abdulaziz University, P.O. Box 344, Rabigh 21911, Saudi Arabia

Abstract: Problem statement: Many existing database applications manage time-varying data. These database applications are referred to as temporal databases or time-oriented database applications that are considered as repositories of time-dependent data. Many proposals have been introduced for developing time-oriented database applications, some of which suggest building support for Temporal Database Management Systems (TDBMS) on top of existing non-temporal DBMSs, while others suggest modifying the models of exiting DBMSs or building TDBMS from scratch. **Approach:** This study addressed several issues concerning developing a technique that enables database designers to understand the way in which time-varying behavior can be modeled and mapped into tabular form. **Results:** Conventional DBMSs do not have the capability to record and process time-varying aspects of the real world. With growing sophistication of DBMS applications, the lack of temporal support in conventional DBMS raises serious problems when used to develop temporal database. The technique of understanding how to think about time and represent it in formal systems is the topic of this study. We examined how to implement time-varying application in the SQL structured query language by introducing temporal data concepts that need to be simulated in DBMSs which lack temporal supports. We proposed a temporal data model that combines the features of previous temporal models and that reduces the cost of memory storage. **Conclusion:** We proposed a technique for implementing temporal database on top of exiting non-temporal DBMS. This technique includes five main areas. These areas are temporal database conceptual design, temporal database logical design, integrity constraints preventions in temporal database, modifying and querying temporal database. We proposed a data model for the temporal database based on the data models which are discussed in literature.

Key words: Temporal database, temporal database models, time-oriented database, valid-time data model, transaction time data model, bitemporal data model

INTRODUCTION

Database Management Systems (DBMS) are supposed to model and record part of the real-world in a well-defined format. The stored data helps many organizations to make important business decisions. Conventional DBMS is used to store and to process the data which refer to the information that is valid at the current time. Temporal database is a modeling technique in database technology that deals with storing time related data (Kostenko, 2007). This modeling technique offers temporal data types and stores information related to the past, the present and to the future. This modeling technique provides expressive and efficient ways to model, store and query different time-state of the stored data.

Time model: Time is represented in the real-world as a line where each point in the line is called an instance and the time between two instances is called period, the length or unanchored segment of time-line is an interval.

Instant of time, period and interval in temporal database are known as temporal data type (Snodgrass, 2000).

Views of time can be considered as:

- A continuous time model, which is considered to be similar to represent time with real numbers and new time point can be defined between two existing time points. As a result, we have an infinite set of time points
- A discrete time model in which time is viewed as natural numbers (integer numbers)

Corresponding Author: Sami M. Halawani, Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, P.O. Box 344, Rabigh 21911, Saudi Arabia

There is a concept of some atomic unit of time, known as a chronon (Patel, 2003). The chronon is the shortest duration of time which is non-decomposable unit of time; it cannot be further divided or broken to generate new time points. Chronon used to build all units of discrete time.

Conceptually, since time may extend to the infinite future of infinite past, so adding some aspect of time into the relational database model should be bounded to indicate the assigned time. In the time line the time is read as time line clock termed as time-line chronons. Every tick of the clock represents a time instance. The calendars relate times on the time line clock to be more familiar in temporal description. For example, the Gregorian calendar defined the time line clock chronon as day, month and year, for example, "22nd of June 2008", this time point known as granules and the partitioning schema that partitioned the time line into finite set of time segment known as granularity, which is a common feature of all temporal data (Snodgrass, 2000).

The discrete time model is considered as the time model for representing temporal database because of the simplicity and relative ease of implementation. If a continuous time model were used to represent time in a temporal database, there would be many problems in providing arithmetic support since there is an infinite precision of time (Patel, 2003).

Taxonomy of time in temporal database has been developed, concerning when a certain event occurs or when a certain fact is considered to be true (Elmasri and Navathe, 2000). The time aspect used in temporal database can be interpreted as the following:

- User-defined time: Which is defined as the column that just happens to be of a date/time data type and does not indicate anything related to the validity of other columns (Snodgrass, 2000)
- Temporal time: In which the column(s) that are of date/time data type are used to indicate time aspects of the associated tuple

MATERIALS AND METHODS

The time models in temporal database systems can be categorized into the following:

Valid-state time: In which the associated time, is used to indicate when certain fact (event) occur or when certain fact is considered to be true in the real world. Databases that support valid-time state is termed as historical database (Snodgrass, 2000). These databases can be represented as three dimensional database as shown below in Fig. 1.

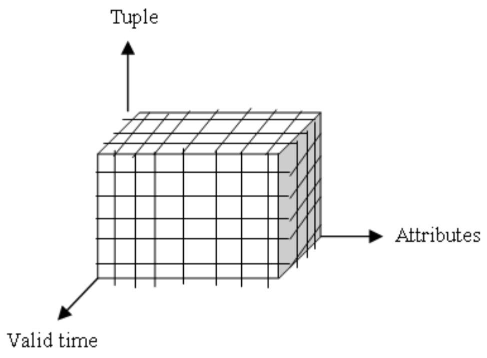


Fig. 1: Three dimensional views of valid-time relations (Snodgrass, 1987)

Valid-state time incorporated in relational database system to become temporal database by adding date/time column(s) into the relation with some granularity to indicate the validity of the desired fact which can be:

- Point time event or fact: Which is typically associated in database with single time point in some granularity
- Duration point or fact: It is associated with specific time period in some granularity

Valid-state time used in temporal database systems to model and record the history of the validity, several different applications prefer this kind for the flexibility that can be gained by recording and processing historical data which can categorized as:

- Proactive update**:** It is applied to the database before it becomes effective in the real world
- Retroactive update**:** The update is applied to the database after it became effective in the real world
- Simultaneous update: An update that is applied at the same time when certain fact or event becomes effective in the real world

Transaction-state time: The associated time refers to the time when the information was actually stored in the database. Transaction-state time is used in temporal database systems to model and record the history of changing state of the transaction-state database tables (Snodgrass, 2000). It is also called Rollback database. The data in transaction-time table is indexed by the transaction time, where the relation can be viewed as cubic to capture the time dimension as shown in Fig. 2.

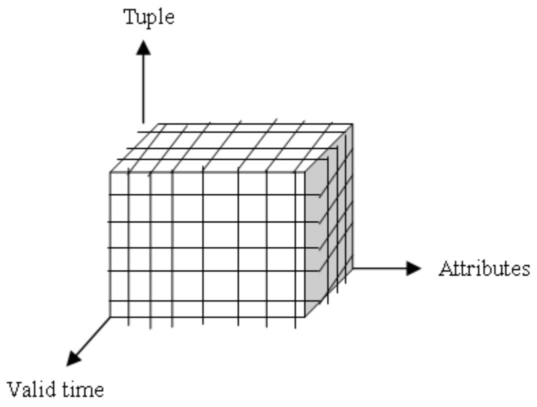


Fig. 2: Three dimensional views of transaction-time relations

Valid-state time and transaction-state time are considered to be the most common time models in temporal database, and they are referred to as time dimensions, in some applications only one of the dimensions is needed and in other cases both time dimensions are required, yielding to bitemporal-state time.

Bitemporal-state time: Associated time refers to both Valid-state time and transaction-state time yield in bitemporal data model. Rollback database views tuples as begin valid at sometimes as of that time (Snodgrass, 2000). Such database can be viewed conceptually as a collection of cubes, one at each transaction time as shown in Fig. 3.

Data models for temporal database: Temporal database models and schemes have been discussed by Segev and Shoshani (1998); Delaney *et al*. (1992); Elmasri and Navathe (2000) and Gadia and Yeung (1988a). The various temporal features that characterize temporal data models are outlined, more explicitly, they concerned with:

- The semantics of time representation (valid time/transaction time)
- Whether timestamp is applied to a tuple or to individual attributes (tuple timestamp/attribute timestamp)
- Whether attribute values are defined for the same or different time period in the same tuple (homogeneous tuple/heterogeneous tuple) (Gadia and Yeung, 1988a; 1988b)
- Whether time is represented as points or intervals (single chronons/intervals/temporal elements)
- 1NF or N1NF relations

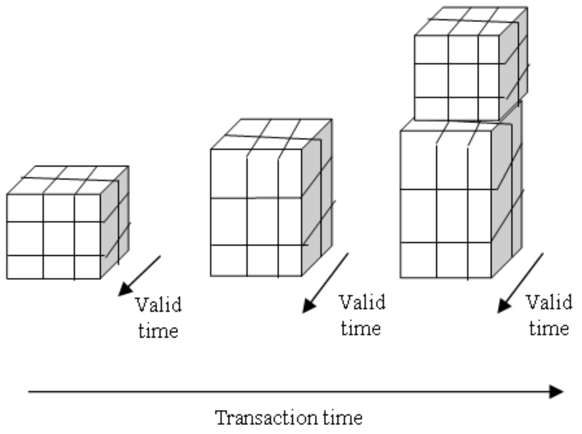


Fig. 3: Bitemporal relations

Table 1: N1NF employee relation

E-name	Dept	Ph. No.
Bill	{<[6/2004, 6/2005), loading>, <[6/2005, now), sales>}	{<[6/2004, 6/2005), 0598877>, <[6/2005, now), 0684477>}
Pat	{<[6/2003, 1/2005), loading>, <[1/2006,now), research>}	{<[6/2000, 3/2005), 059878877>, <[3/2005, now), 0684477>}

In addition to that, other concepts are used to describe the temporal data model, we can list them as the following:

- The associated time or the time model can be either discrete or continues time model
- The time model is bounded and finite, which means that there is start time and end time point to indicate the temporal aspect of the database object
- A linear model of time means that only one version of data is available at any time

Linear time model as oppose to a branching model of time, the branching model of time allows alternative versions of data to hold at any given time.

So, discrete, bounded, finite and linear data model approach is used in modeling temporal database (Patel, 2003)

In general, there are two main approaches for modeling temporal relational database Goralwalla *et al*. (1995) and Ahn and Snodgrass (1986). They are as follows:

Attribute time stamp: The time is attached to attribute values of a relation and the histories of an attribute are included in a set of triplet-valued, as shown below in Table 1.

The triplet of the form <[l, u), v> means "l" represents lower time bound, "u" represents upper time bound and "v" represent the value of the attribute, this approach violates 1NF since it does not contain single or indivisible value, a temporal data model using nested relation is based on this approach which is discussed in (Garani, 2003). We saw in our research that this approach needs more work for query optimization, thus it is excluded from our study.

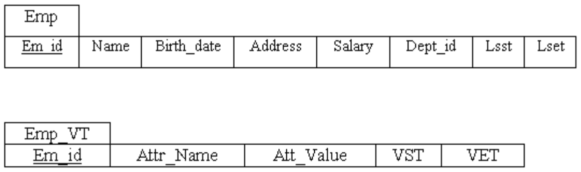
Tuple time stamp: Where the time stamp can be one of the following:

- Tuple Timestamp Single Relation (TTSR) that holds all its pertaining time varying attributes along with non-temporal attribute so time stamp is represented as two additional time attributes named "From" and "To" fields, this approach is not efficient since if a relation has many attributes, a whole new tuple version is created whenever any one of the attributes is updated. If the attributes are updated asynchronously, each new version may differ in only one of the attributes, thus needlessly repeating the other attribute values leading to hug space needed
- Tuple Timestamp Multiple Relation (TTMR) where the temporal relation is decomposed as the following:
    - Time varying attributes are distributed over multiple relations and non-temporal attributes are gathered into separate relation

RESULTS

Based on the two main approaches for modeling temporal relational DB discussed above we proposed a third data model, we named this model a Tuple Timestamp Historical Relation (TTHR) in which the relation that needs to capture temporal time aspects decomposed in two relations, one represents the current state relation and the other recodes the changes in all the time varying attributes. The approach is as the following:

- Keeping non-temporal attribute and time-varying attribute in the original classical relation as it is and add additional date/time attribute to the relation according to the desired temporal type will be developed in some granularity, this relation will represent the current state of modeled reality, with the added date/time column to indicate the valid-time DB, Transaction-time DB and Bi-temporal-time DB, respectively, the absence of VET or TET in this relation will reduce the memory usage and these data can be extracted implicitly since this relation represents the current state



Emp

Em_id	Name	Birth_date	Address	Salary	Dept_id	Lsst	Lset

Emp_VT

Em_id	Attr_Name	Att_Value	VST	VET

Fig. 4: Temporal relational data model of employee relation

- Creating new relation with these columns (1) key(s) attribute, (2) time-varying attribute name, (3) time-varying value, (4) timestamp start and (5) timestamp end. This relation will be referred to as sequenced changed table that will hold all the historical changes of all time-varying attribute

Example: The employee relation Emp become temporal as shown in Fig. 4.

In Emp_VT relation, the field (Attr_Name) will hold the name of columns that have been changed and (Att_Value**)** field will store the changed value, this field size should be of the size of the largest field in the (Emp**)** of variant data type to hold the value of others temporal fields.

Cost model: we introduce the cost of the memory usage when different temporal database models are used. Fortunately, the results with regards to performance of the different approaches for temporal database model are already available in (Goralwalla *et al.,* 1995). Different queries that may cover most combinations of possible requirement were tested against TTMR and TTSR. The tests cover current status data and historical data, it has been found that TTMR surpass TTSR order of magnitudes in performance for both current status and historical data, with regards to execution time. But the comparison with regards to used memory is not available and carried out in this paper for TTMR and TTSR. We extend this comparison to include the proposed model.

Definitions and axioms:

Definition 1 (valid-time database relation): Valid-time database relation is a set of attributes that construct the relation and can be grouped into 4 subsets, key attributes, time-invariant attributes (s) (unchangeable), time-varying (changeable) attributes and timestamp attributes. They are represented by K, U, C and T respectively, so:

$$R = \{\{A_{K1}, A_{K2}, \ldots, A_{Kn}\}, \{A_{U1}, A_{U2}, \ldots, A_{Un}\}, \{A_{C1}, A_{C2}, \ldots, A_{Cn}\}, \{A_{T1}, A_{T2}\}\}$$

Where:

$A_K = \{A_{K1}, A_{K2}, A_{K3}, \ldots, A_{Kn}\}$

$A_U = \{A_{U1}, A_{U2}, \ldots, A_{Un}\}$
$A_C = \{A_{C1}, A_{C2}, \ldots, A_{Cn}\}$
$A_T = \{A_{T1}, A_{T2}\}$
$R \ = \{A_K, A_U, A_C, A_T\}$

Definition 2 (time-invariant attribute): Time-invariant attribute is an attribute whose values are not changed with a time, Time-invariant attributes can be updated as in the case of an error, but a database does not keep a history of it.

Definition 3 (time-varying attribute): Time-varying attribute is an attribute whose values are associated with timestamps.

Definition 4 (timestamp): A timestamp is a time value associated with a Time-stamped object (i.e., an attribute value or tuple).

Definition 5 (lifespan): The lifespan of a database objects is the time through which the object is defined.

Definition 6 (frequency of time-varying attribute): is the number of times this attribute to be updated (changed) within a specific interval of time.

$F(A_{Ci})$: Frequency of times A_{Ci} changing within an interval of time where i in $(_1\ldots\ldots_{Cn})$

Definition 7 ($S(A_{ji})$): Size of field/attribute A_{ji} in bytes where j in {K,U,C,T} and i in $(_1, _2, _3,\ldots, _n)$.

Definition 8 Cost (A_j): The cost of a subset A_j, which is the summation of all attributes size in A_j in bytes where j in {K, U, C, T}.

Definition 9 Cost(r): The cost of a tuple (row) r in R is the summation of all the cost of subsets attributes = $Cost(A_k)+Cost(A_u)+Cost(A_c)+Cost(A_T)$.

Axioms 1: The cost of different attribute type is defined as:

$$Cost(A_K) = \sum_{i=1}^{Kn} S(A_{Ki}) = K \text{ byte} \quad (1)$$

$$Cost(A_U) = \sum_{i=1}^{Un} S(A_{Ui}) = U \text{ byte} \quad (2)$$

$$Cost(A_C) = \sum_{i=1}^{Cn} S(A_{Ci}) = C \text{ byte} \quad (3)$$

$$Cost(A_T) = \sum_{i=1}^{2} S(A_{Ti}) = T \text{ byte} \quad (4)$$

Axioms 2:

$$F(A_C) = \sum_{i=1}^{Cn} F(A_{Ci}) = \delta \text{ times} \quad (5)$$

in interval of time = λ say 3 months, 6 months, one year, or two years, or any interval of time depends on the nature of the developed system.

The frequency of changing of the time-varying attribute (A_C) in interval of time λ can be calculated as:

Calculation of memory cost needed for different models: In comparison with regards to used memory in different model a fixed length not spanning records for the database file structure design is assumed to be applied in our study. Time stamps are represented in “VST” and “VET” i.e., we will take valid-time model for representing temporal database.

TTSR model:

TTSR relation:

$A_{K1}\ldots A_{Kn}$	$A_{U1}\ldots A_{Un}$	$A_{C1}\ldots A_{Cn}$	$A_{T1,}A_{T2}$

The cost of representing one row can be calculated as:

$Cost(r) = Cost(A_K)+Cost(A_U)+Cost(A_C)+Cost(A_T)$
$\quad = K+U+C+T$ byte as formulas above 1-4

The cost of representing history data of one row with $F(A_C) = \delta$(delta) times in λ(lamda) interval of time is:

$$= (K+U+C+T)*\delta \quad (6)$$

Since each changing in any A_C require insert new row with all attributes.

TTHR model (proposed):

TTHR-snapshot relation:

$A_{K1}\ldots A_{Kn}$	$A_{U1}\ldots A_{Un}$	$A_{C1}\ldots A_{Cn}$	A_{T1}

TTHR-history relation:

$A_{K1}\ldots A_{Kn}$	Index	α	A_T

The cost of representing one row can be calculated as:

$Cost(r) = Cost(A_K)+Cost(A_U)+Cost(A_C)+Cost(A_{T1})$
$\quad = K+U+C+T/2$ byte as formulas above 1-4

The cost of representing the history data of one row with:

$F(A_C) = \delta$ times in λ interval of time is:

$$= (K+index+\alpha+T)*\delta$$
$$= (K+1+\alpha+T)*\delta \quad (7)$$

Where:

Index =	New attribute to index the time-varying attributes with one byte size
α =	New added attribute of variant data type to hold data from different type, it's size assumed to be as the size of the largest field size in A_C

Since each changing in any A_C requires inserting new row in the second relation for the old value of the effected (changed) time-varying attributes, the total saves in memory (space) for TTHR over TTSR for a certain interval of time δ can be calculated as the following:

$$\text{Cost(TTSR)} = (K+U+C+T)*\delta \text{ as in 6}$$

$$\text{Cost(TTHR)} = (K+1+\alpha+T)*\delta \text{ as in 7}$$

$$\text{Cost (improvement)} = \frac{\text{Cost}(\text{TTSR})-\text{Cost}(\text{TTHR})}{\text{Cost}(\text{TTSR})} = \frac{(K+U+C+T)*\delta-(K+1+\alpha+T)*\delta}{(K+U+C+T)*\delta} = \frac{U+C-(1+\alpha)}{K+U+C+T}$$

as (1+ α) represent small value so:

$$\text{Cost (improvement)} \approx \frac{U+C}{K+U+C+T}$$

In case $\alpha \rightarrow C$ the improvement will be:

$$\approx \frac{U}{K+U+C+T}$$

The save in memory would be directly proportional to δ and to the number and size of attributes in C set.

TTMR model:

TTMR-non-temporal relation:

$A_{K1}\ldots A_{Kn}$ $A_{U1}\ldots A_{Un}$

TTMR- A_{C1} relation:

$A_{K1}\ldots A_{Kn}$ A_{C1} A_T

TTMR- A_{C2} relation:

$A_{K1}\ldots A_{Kn}$ A_{C2} A_T

TTMR- A_{C3} relation:

$A_{K1}\ldots A_{Kn}$ A_{C3} A_T

TTMR-A_{Cn} relation:

$A_{K1}\ldots A_{Kn}$ A_{Cn} A_T

The cost of represent one row can be calculated as:

$$\text{Cost}(r) = K+U+\sum_{i=1}^{Cn}(S(A_{Ci})+K+U) \quad (8)$$

The cost of representing history data of one row with $F(A_C) = \delta$ times in λ interval of time with:

$$\text{Cost}(r) = \sum_{i=1}^{Cn}(S(A_{Ci})+K+U)*\delta i \quad (9)$$

where, δi is the number of times A_{Ci} may be updated in interval of time = λ.

Since each changing in any A_{Ci} require insert new row in the A_{Ci} relation for the new value of the effected (changed) time-varying attributes.

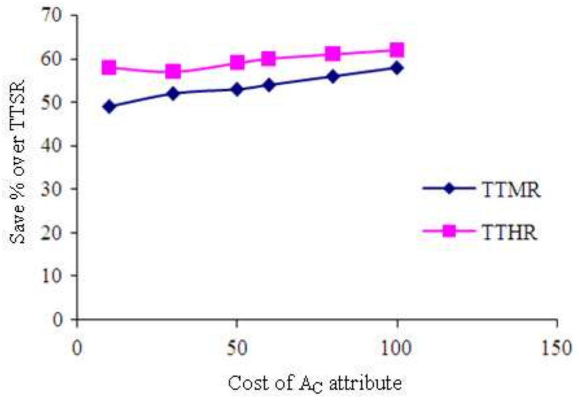
The total save in memory (space) for TTMR over TTSR for a certain interval of time δ can be calculated as the following:

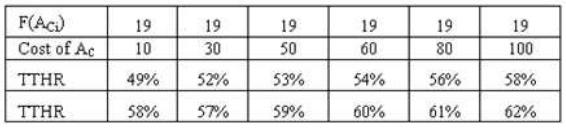
$$\text{Cost(TTSR)} = (K+U+C+T)*\delta \text{ as in 6}$$

$$\text{Cost(TTMR)} = \sum_{i=1}^{Cn}(S(A_{Ci})+K+U)*\delta i \text{ as in 9}$$

$$\text{Cost (improvement)} = \frac{\text{Cost}(\text{TTSR})-C(\text{TTHR})}{\text{Cost (TTSR)}} = \frac{(K+U+C+T)*\delta-\sum_{i=1}^{Cn}(S(A_{Ci})+K+U)*\delta i}{(K+U+C+T)*\delta}$$

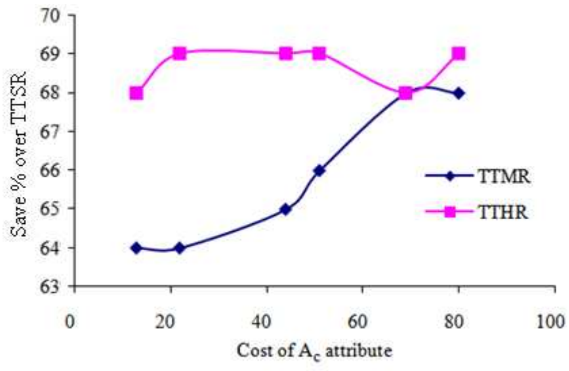
The cost of each A_{Ci} and δ of each A_{Ci} should be known to calculate the Cost improvement. But by experiment we found that the memory save improvement in this model over TTSR range from 20-71% and several parameters may affect the improvement like number of A_C set and δ where the saving in space may exceed 100% with much higher δ.

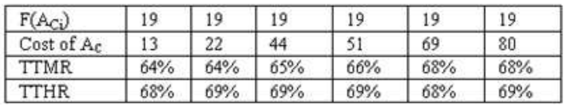




$F(A_{Ci})$	19	19	19	19	19	19
Cost of A_C	10	30	50	60	80	100
TTHR	49%	52%	53%	54%	56%	58%
TTHR	58%	57%	59%	60%	61%	62%

Fig. 5: The memory saves of TTMR and TTHR over TTSR (Experiment 1)



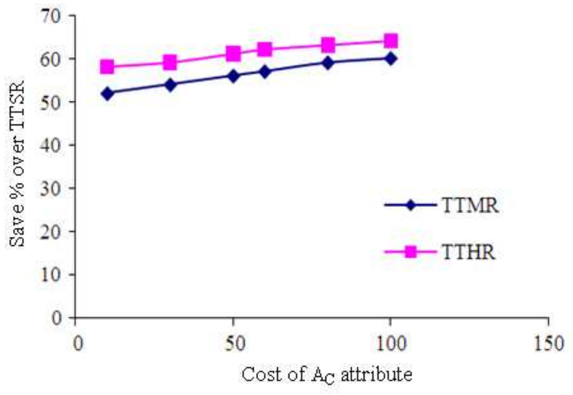


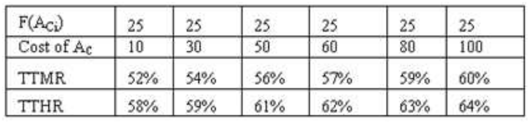
$F(A_{Ci})$	19	19	19	19	19	19
Cost of A_c	13	22	44	51	69	80
TTMR	64%	64%	65%	66%	68%	68%
TTHR	68%	69%	69%	69%	68%	69%

Fig. 6: The memory saves of TTMR and TTHR over TTSR (Experiment 2)

Using this model for representing temporal database model satisfied memory save but in contrast, it costs too much in query. Since decomposing the relation in to C_n relations and combine information from separate relations, temporal intersection join would be needed, which is generally expensive to implement.

Detailed evaluations for different temporal databases are discussed by (Ahn and Snodgrass, 1986). We carried out the experiment several times with varying the cost of A_C and freezing δ to make it equal 19 and 25.





$F(A_{Ci})$	25	25	25	25	25	25
Cost of A_c	10	30	50	60	80	100
TTMR	52%	54%	56%	57%	59%	60%
TTHR	58%	59%	61%	62%	63%	64%

Fig. 7: The memory saves of TTMR and TTHR over TTSR (Experiment 3)

Experiment 1: As shown in the table below the graph, we got the results as shown in the graph in Fig. 5.

Improving the memory usage in TTMR or in TTHR depends on δ and the cost of A_C as seen in Fig. 5 we do the experiment by freezing the $F(A_{Ci})$ at 19 and varying A_C from 10-100 bytes.

Experiment 2: In Fig. 6 we do the experiment by freezing the $F(A_{Ci})$ at 19 and varying A_C from 13-80 byte, but with different values of Experiment 1. We can conclude that the proposed temporal data model achieves memory space save that is roughly equal or greater than that in TTMR and in our case study we preferred to use TTHR for its simplicity.

Experiment 3: In Fig. 7 we do the experiment by freezing the $F(A_{Ci})$ at 25 and varying A_C from 10-100 bytes, we got the same result as in Experiment 1.

DISCUSSION

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. While the first approach ensures the maximum efficiency, the second approach is more realistic and more popular. Wang *et al*. (2006) proposed transaction-time extensions for database systems that require no modification of the existing standards of database using XML. Where, XML provides excellent support for temporally grouped data models, which have long been advocated as the most natural and effective representations of temporal information.

Anyi (2006) adopted a practitioner's approach for compute temporal aggregation and temporal universal quantification in standard SQL (not-temporal SQL). This provides a solution for users that need such time-varying facilities in their applications. Which are based on DBMS's that do not support time-varying facilities Snodgrass (2000) in his book "Developing Time-Oriented Database Applications on SQL' covered the different aspect of temporal database and proposed a technique for developing this kind of application on different DBMS using a set of assertions or triggers to satisfy the temporal aspects, features and constrains.

CONCLUSION

We have proposed a data model for the temporal database based on the data models which are discussed in (Gregersen and Jensen, 1998; Segev and Shoshani, 1998; Ahn and Snodgrass, 1986). Tests in (Ahn and Snodgrass, 1986) have shown that tuple time stamping temporal data model that involves multiple relations for every time-varying attributes have a better overall performance and efficiency in both the processing time and used space. Our proposed data model is based on tuple time stamping with two relations, one relation is for the current snapshot data and the other one is the auxiliary relation that holds the temporal aspects of whole time-varying attributes, the proposed temporal data model achieves saving in memory usage range from 70-90% over the temporal data model discussed in (Novikov and Gorshkova, 2008), where a framework for temporal database implementation is discussed.

REFERENCES

Ahn, I. and R. Snodgrass, 1986. Performance evaluation of a temporal database management system. Proceedings of the 1986 ACM SIGMOD International Conference Management of Data, May 28-30, ACM Press, Washington DC., United States, pp: 96-107. DOI: 10.1145/16894.16864

Anyi, E., 2006. Temporal aggregates and temporal universal quantification in standard SQL. ACM SIGMOD Rec., 35: 16-21.

Delaney, C., D. Rama and P. Srinivasan, 1992. Design of a temporal database for phlebitis. Proceedings of the 1992, ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's, (ACTA'92), ACM Press, Kansas City, Missouri, United States, pp: 204-209. DOI: 10.1145/143559.143642

Elmasri, R. and Navathe, 2000. Fundamentals of Database Systems. 3rd Edn., Addison Wesley, USA., ISBN: 0-201-54263-3, pp: 744-754.

Gadia, S. and C. Yeung, 1988a. A generalized model for a relational temporal database. Proceedings of the 1988 ACM SIGMOD International Conference on Management of data, June 1-3, ACM Press, Chicago, Illinois, United States, pp: 251-259. DOI: 10.1145/50202.50233

Gadia, S., 1988b. Homogeneous relational model and query languages for temporal databases. ACM Trans. Database Syst. 13: 418-448.

Garani, G., 2003. A temporal database model using nested relations. Doctor of Philosophy in the University of London, School of Computer Science and Information Systems, Birkbeck College, UK, http://www.dcs.bbk.ac.uk/research/recentphds/garani.pdf

Goralwalla, I., A. Tansel and M. Ozsu, 1995. Experimenting with temporal relational databases. Conference on Information and Knowledge Management, Proceedings of the fourth International Conference on Information and Knowledge Management, Nov. 29- Dec. 2, ACM Press, Baltimore, Maryland, United States, pp: 296-303. DOI: 10.1145/221270.221597

Gregersen, H. and C. Jensen, 1998. Conceptual modeling of time-varying information. Department of Computer Science, Aalborg University, http://www.cs.aau.dk/~csj/ Thesis/pdf/chapter32.pdf

Kostenko, B., 2007. Temporal preprocessor: Towards temporal applications development. Proceedings of SYRCODIS'07, the 4th Spring Young Researchers Colloquium on Databases and Information Systems, June 21-21, CEUR-WS.org, Moscow, Russia, pp: 1-3. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-256/submission_14.pdf

Novikov, B. and E. Gorshkova, 2008. Temporal databases: From theory to applications. Programm. Comput. Software, 34: 1-6.

Patel, J., 2003. Temporal Database System Individual Project. Department of Computing, Imperial College, University of London, Individual Project. http://www.doc.ic.ac.uk/~pjm/teaching/student_projects/ jaymin_patel.pdf

Segev, A. and A. Shoshani, 1998. Functionality of temporal data models and physical design implementations. IEEE Database Eng., 11: 38-45.

Snodgrass, R., 1987. query language TQuel. ACM Trans. Database Syst., 13: 418-448.

Snodgrass, R., 2000. Developing Time-Oriented Database Applications in SQL. 1st Edn., Morgan Kaufmann Publishers, Inc., San Francisco, pp: 504.

Wang, F., X. Zhou and C. Zaniolo, 2006. Using XML to build efficient transaction-time temporal database systems on relational databases. Proceedings of the 22nd International Conference on Data Engineering, Apr. 3-7, IEEE Computer Society, Washington DC., USA., pp: 131. DOI: 10.1109/ICDE.2006.168