Journal of Engineering and Applied Sciences 13 (1): 222-230, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

Performance Evaluation of Tuple Timestamp Multiple Historical Relation Data Model

Shailender Kumar
Department of Computer Science and Engineering, AIACTR, Delhi, India

Abstract: Most of the prevalent database applications in the contemporary environment maneuvers with the temporal data. Temporal records characteristically vary over a due course of time while conventional databases don't facilitate any feature to manage multiple snapshots of these records captured at discrete intervals. Consequently, it becomes difficult to cope up with this varying nature of temporal records by using traditional database approach. This restraint has been resolved by various data models which are implemented as an extension of the conventional database systems. This study presents a Tuple timestamp multiple historical relation data model which possesses features like minimal storage and minimal query execution time unlike the earlier historical relation data models. Tuple timestamp multiple historical relation data model lays emphasis upon storage of the preceding instances of the records into multiple historical relations. B-Tree Indexing is applied for scaling up the efficiency of the data model.

Key words: Indexing, open source database, temporal database, triggers, Tuple time-stamping, valid time

INTRODUCTION

A database management system is expected to record and architect real world data in an organized format. The data stored in these databases help in making crucial decisions which in turn, formulates the long term strategies. Time oriented database is a technique in database technology to store data that embrace time aspects. It provides a seamless way to store, model and query data which evolves over time. An instance of time is represented as a point on a time line and the time between any two instances is called time period.

Over the years, researchers (Edelweiss *et al*., 2000; Halawani *et al*., 2012; Pilev and Georgieva, 2012; Kunzner and Petkovic, 2015) have put extensive efforts to advance novel methods for effectively managing temporal data. The most of the developments in this field primarily focuses on enforcement of the unique trades over the orthodox databases. The contemporary studies focus upon the employment of time-oriented architecture with the help of openly available databases. Time oriented applications provision numerous versions of time (Galante *et al*., 2005; Petkovic, 2014; Atay, 2016) of schemas: machine-oriented schemas, valid-time range schemas and bi-temporal schemas. Machine-oriented schemas follows machine time to seizure variations applied to the data values. Valid-time range schemas imprison the instant in which the data values remains intact and legal in the materialistic applications. A bi-temporal schema (Garani, 2012; Atay, 2016) exercises the merger of legal and machine time to put a whole mark of time upon its values. A temporal data type which can be applied in the schemas is completely dependent over implication grassland. The present and past instances of the records put a restrain to the system time whereas the future instances are decisive for the valid time. Usually two different tactics are followed to attach the whole mark of time in the schema: one is tuple-stamping where normal form schema in its first form is applied while the other one is attribute-stamping where normal form schemas which are not in the first form are applied.

Although, there has been large inclination of researchers towards this research area in recent time, yet no standard temporal database have been developed till date. The existing tuple time-stamping models are not capable enough to optimize the query execution time, memory cost and circumlocution arisen due to asynchronous update of the vibrant traits. The researchers present Tuple-stamped heterogeneous archived schemas which addresses these problems here. It uses valid time as one of the parameter to put a whole mark of time over the values stored in the database. The rational start-time and rational end-time of this varying record is provided by the user. The imposition of integrity constraints over thetime-oriented values ensures its exactness and efficacy. In Tuple timestamp heterogeneous archived schemas, numerous time-oriented schemas are created for each of the vibrant traits provided their values static in nature. Else, the basic vibrant schema will persist in their original form. All the changes made within the temporal attributes are captured separately and a log is created in the form of history relation. The proposed temporal database model is implemented using Postgres an open source database system. The temporal relations used in this model are heterogeneous in nature. The results in this paper show that the proposed model is capable of optimizing the query execution time and the memory cost as compared to the other existing historical relation database models. The efficiency of the proposed model increases with the increase in frequency of update operations.

The researcher have detailed the contribution of this article as follows: in this study, the researcher seek to minimize the query execution time and memory cost associated with the data model by considering the degree of heterogeneity of the time-varying attributes. Specifically, the researcher have proposed a novel method which involves indexing as well as multiple relations for the dynamic attributes to remove the existing redundancy, so that, the performance of the proposed system enhances significantly. The researcher have taken special care in capturing the evolutions in the value of the attributes in the separate relations. This is due to the fact that value of all the attributes doesn't necessarily change at the same instant. Another significant feature of the researchers work is that they have designed their system in such a manner that it remains suitable even for the large scale database applications. In fact when the data set is very large then the probability of value of each attribute being distinct increases steeply. As a consequence, the window for overlapping region of values of different attributes shrinks significantly which helps in enhancing the efficiency of proposed system. To sum up this discussion it can be stated that the overall cost of the data model which is primarily dependent over the query execution time and memory cost.

Literature review: Some of the former developments provisioned the exploration of certain contemporary time-oriented data schema (Edelweiss *et al*., 1993; Kunzner and Petkovic, 2015; Kvet *et al*., 2015). A numerous time-oriented data models are available which have been implemented (Goralwalla *et al*., 1998; Tansel, 2004; Chau and Chittayasothorn, 2007; Yang *et al*., 2015) on the top of conventional databases. The logical architecture of time-oriented schemas substantially concentrates on the connotations (Jensen and Snodgrass, 1996; Anselma *et al*., 2013), storage, query processing and implementation.

A number of papers surveyed the performance aspect of the existing time-oriented data models. The research by Kunzner and Petkovic (2015) equated the utility of the employments designed by applying in-built concepts and the manually programed employments. Study by Atay (2016) contrasted the attribute-stamping and tuple-stamping tactics in bi-temporal data schemas. The constraints regarding the utility of time-oriented data schemas using attribute-stamping and tuple-stamping approach are analyzed too.

There are a number of time-oriented data models which are based on the existing relational data models (Jensen and Snodgrass, 1999; Safarik, 2011). The two discrete approaches for associating the timestamps are tuple time-stamping and attribute time-stamping. Tuple time-stamping approach (Bohlen *et al*., 1998; Safarik, 2011; Halawani *et al*., 2012; Kumar and Rishi, 2016) was used in abundance in the majority of earlier work. Now a days, tuple time-stamping where first normal form relations are applied is used to append the timestamp. The logical architecture of time-oriented schemas substantially concentrates on the connotations (Jensen and Snodgrass, 1996; Anselma *et al*., 2013; Khatri *et al*., 2014), storage, query processing and implementation.

Time dimension is an essential feature which is considered while designing the time oriented databases. The primary classifications of time dimension are application time, system time and bi-temporal (Mahmood *et al*., 2012; Atay, 2016). The time dimension which has been abundantly used in temporal research papers is application time (Anselma *et al*., 2013). It encompasses all of the current, archived and imminent specimen of accounts while the machine time covers solely current and archived specimen. Methods have been developed to implement the time-oriented logic in the grassland of machine-learning and cybernetics also. Time-oriented data also plays a key role in these fields. Study by Kaufmann *et al*. (2015) proposed an indexing approach for executing the queries in bi-temporal databases. Data models have also been proposed to entertain the subjects related to indeterminacy (Terenziani, 2016) in time-oriented relational databases. Several studies (Halawani *et al*., 2012) explained that tuple timestamp historical relational model applies customary SQL for enforcement of the schema.

The researcher states that in this model they have used heterogeneous contemporary and archived schema for the temporal traits. They eliminate the redundancy caused by the bogus addition of temporal attributes while updating the time-varying parameters. Query execution time is optimized by further indexing of the non key attributes. Research carried out in temporal database field so far, rarely used the open source database. The researcher proposed work applies the largely prevailing openly available database PostgreSQL for employment of the tuple timestamp multiple historical relation data model.

Preliminiaries: A time oriented data model consists of conceptual and logical aids for elaborating time-oriented relation and its integrity constraints.

Temporal relational schema: Time-oriented schema is an assemblage of keys along with time-oriented and time-invariant traits. Tuple time-stamping is used in temporal relations to imprison the validity of the attribute. Temporal relational schema has been demonstrated mathematically in Eq. 1:

$$R^t = (K_1, K_2, K_3, ..., K_n, S_1, S_2, S_3, ..., S_n, D_1, D_2, D_3, ..., D_n | T) \quad (1)$$

where, R^t is a time-oriented relation with fixed range of keys, invariant-traits, variant-traits and timestamp (K_1, K_2, K_3, ... , K_n), (S_1, S_2, S_3 ... , S_n), (D_1, D_2, D_3... , D_n) and variable T denotes the associated timestamp.

Temporal and non-temporal attributes: The value of non-temporal attributes does not rely on time which means their value does not changes with time. Therefore, these attributes are said to be static in nature. On the other hand, the value of temporal attributes evolves with time and the timestamp is associated with them. Hence, these attributes fall under the category of dynamic attributes.

Temporal relational algebra: Temporal relational algebraic expression contains temporal operators as well as predicate symbols for querying time oriented database. Time-oriented operators applied in this domain are differs from the traditional algebra, since, they provision time-dimension. The time-oriented algebra is employed in a careful manner, so that, it does not breach certain integrity constraints.

Memory cost of temporal relation attributes:

$$\text{Cost of Key attributes} = \sum_{i=1}^{K_n} K_i = K \text{ bytes} \quad (2)$$

$$\text{Cost of Static attributes} = \sum_{i=1}^{S_n} S_i = S \text{bytes} \quad (3)$$

$$\text{Cost of Dynamic attributes} = \sum_{i=1}^{D_n} D_i = D \text{ bytes} \quad (4)$$

$$\text{Cost of Timestamps } (T) = T \text{ bytes} \quad (5)$$

Frequency of time varying attributes: Frequency of dynamic attributes (time varying) at any interval of time say 1 day, 1 week, 1 month or 1 year:

$$= \sum_{i=1}^{D_n} (D)_i = f \quad (6)$$

MATERIALS AND METHODS

Temporal data model: Tuple timestamp multiple historical relation data model is designed over the PostgreSQL database platform and it effectually manages temporal data. The consumer may interoperate with the time-oriented database through user interaction portal designed under Java NetBeans IDE Version 8.0.2. This developments empowers the consumer to modify current and archived data. The catalogue is beheld as an assemblage of contemporary table and the archived table. The contemporary table stockpiles the up-to-date snapshots of data.

Tuple time-stamping approach (Halawani and Al-Romema, 2010; Kumar and Rishi, 2017) is applied to store all the currently valid tuples of the real world into current table. New records are introduced into the contemporary tables only. The archived table stores the precedent varieties of the values. On applying a streamline assignment over the contemporary table, the previous data of the schema gets mechanically insinuated into archived table by the use of temporal functions. The queries submitted for the retrieval of previous instances will be replied by the history tables.

Whenever there is a need for carrying out update operation over the present table, triggers are taken into use and are evoked as per requirement. In above mentioned scenarios, the archived data of the row to be streamlined are moved to the archived relation. The maximum limit of the time-period traits of archived table is set identical to the minimum limit of the contemporary table streamlined time-oriented traits. In actual terms, data is never deleted from the temporal database on applying delete operation, it just moves from the current relation to the history relation.

The researcher have also implied tuple-stamping into the archived table. It applies time-range data structure for capturing the validity period of an attribute. Upper bound of the time-range section is identical to the minimum limit of the time-range of the freshly streamlined table. The primary key of archived relation is obtained by unification of the primary key of contemporary table and the time-range trait of the archived table. A timestamp is associated with tuple and this is done through many ways like by using single relation data model, historical relation data model and multiple historical relation data model. A tuple timestamp single relation data model holds both its temporal attributes as well as non-temporal attributes. Time stamp is represented by using two additional temporal attributes named as valid from and valid to. A complete new tuple is added whenever the value of any attribute gets updated. During asynchronous update, each new tuple may differ with the others in only one of the attributes.

In historical relation data model, the relation that captures time varying attributes is decomposed into two relations. First of them represent the current values of an attributes and it is also known as snapshot relation as it contains only current aspects of an object while the second one records the changes made in all of the time varying attributes. This relation can be grouped into four subsets: key, static attributes, dynamic attributes and timestamp attributes.

In multiple historical relation data model, the relation which has to capture time-varying time aspects is decomposed into a number of relations which are equivalent to a number of dynamic attributes. An identical history relation for every dynamic attribute is maintained automatically by the use of triggers. A snapshot relation of dynamic attribute captures only current values of an attribute and the past values are automatically transferred into a history relation of that attribute. Temporal relations are represented by key attributes, static attributes, dynamic attributes and timestamp:

$$R\{key_{attr}, static_{attr}, dynamic_{attr}, ts_{attr}\} \quad (7)$$

Where:

$Key_{attr} = \{K_1, K_2, K_3, \dots, K_n\}$
$Static_{attr} = \{S_1, S_2, S_2, \dots, K_n\}$
$Dynamic_{attr} = \{D_1, D_2, D_3, \dots, D_n\}$
$Timestamp = \{T\}$

Tuple timestamp historical relation data model: A current relation is used to capture the existing aspects of an object and a history relation is used to capture the previous values of an object. The previous values show the temporal evolution of an object. The value of a dynamic attribute varies with time as the state or role of an entity varies but the magnitude of an invariant trait remains same. So, we need to capture these changes in the value of an object. Here, we use a separate function named as history relation for capturing the past values of an object. We just need to capture these time varying attributes as it is beneficial from the memory storage point of view. The amount of memory needed is less as compared to single relation data model in which static attributes as well as dynamic attributes are stored in a single relation. Complexity of this historical data model is a little bit high as compared to single relation data model due to the involvement of various triggers and integrity constraints.

A current relation or snapshot relation have four fields: key attributes, static attributes, dynamic attributes and timestamps. Snapshot relation of historical data model:

- Key
- Static attributes
- Dynamic attributes
- Timestamps

It possesses only current aspects of an object and rest of the past values are automatically transferred into a history relation by the use of triggers. A trigger executes automatically on the occurring of any event like insert, delete or update. Here, triggers are used for managing transfer of data between current and history relation. A history relation have three fields: key attributes, dynamic attributes (time varying) and timestamps. History relation of historical data model:

- Key
- Dynamic attributes
- Timestamps

These timestamps reflect the validity of instance of an object by using two fields “valid from” and “valid to”.

Tuple Timestamp Multiple Historical Relation data model (TTMHR): In this data model, multiple current or history relations are used for capturing the values of an object. A current or snapshot relation is used to capture current aspects of an object where a history relation is used to capture the previous values of an object which shows the temporal evolution of an object. The value of a temporal attributes vary with time as the state or role of an object will change where the value of a static attributes remain same. Here, we need to capture the changes occurred in the attribute value of an object. Thus, we use a separate relation named history relation for storing the previous values of an object and we just need to capture time varying attributes which is beneficial as we are thinking in terms of memory storage. An identical history relation for every dynamic attribute is maintained automatically by the use of triggers. A snapshot relation of dynamic attribute captures only current values of an attribute and the previous values are automatically transferred into a history relation of that attribute. By doing this, we will eliminate the redundancy caused by the bogus addition of temporal attributes while updating the time-varying parameters. So, the memory cost of tuple timestamp multiple historical relation data model is low as compared to historical relation data model. Query execution time is also optimized by further indexing of the non key attributes. A non-temporal relation have two fields: key attributes and static attributes. Non-temporal relation of TTMHR data model:

- Key
- Static attributes

It holds only static aspects of an object and rest of the temporal values captured in the separate relations. A trigger execute automatically on the occurrence of any temporal operations like insert or update.

More than one current or history relations are formed for capturing the evolution of time varying attributes where time varying attributes are distributed over multiple relations. A separate relation is formed for every time varying attribute for optimizing the memory cost of data model. It is done by removing redundancy from the data model. Current or history relation have three fields: key attributes, dynamic attributes (time varying) and timestamps. Current or history relation of TTMHR data model:

- Key
- Dynamic attribute
- Timestamps

The validity period of an object instance is shown by using time-range which shows the validity start time and validity end time.

RESULTS AND DISCUSSION

The proposed tuple timestamp multiple historical relation data model primarily consists of these elements: time-oriented data structure and range of time-oriented algebraic operators which are applied to provide the time-oriented feasibility to the model. In postgres "tsrange" of time-range data structure is applied for putting a whole mark of time over the values stored in the database. It shows the validity period during which the attribute value of the tuple is valid in the real world.

System pre-requisites: The experiments are performed on the system configured with features like 2 GB of RAM and Intel(R) Core(TM) i3-3217U 1.8 GHz. The proposed model is implemented on an open source platform named as PostgreSQL of Version 9.5 and the database application interface is created by using Java NetBeans IDE Version 8.0.2.

Memory cost

Historical relation data model

Current or snapshot relation:

$$[(K]_1,(K_2, K_3,...,K_n, S_1, S_2, S_3,..., S_n, D_1, D_2, D_3, ..., D_n, \text{Timestamp})$$

History relation:

$$[(K]_1, K_2, K_3,...,K_n, D_1, D_2, D_3,..., D_n, \text{Timestamp})$$

In current or snapshot relation, there are so, many attributes like static (non-temporal) attributes, dynamic (temporal) attributes, key attributes and timestamp. In history relation only dynamic attributes, key attributes and timestamps are present. A timestamp in current relation shows the beginning of validity of an attribute. It doesn't show the end of validity time because a current or snapshot relation holds an attribute value which is presently valid in the real world. Recording of changes in attribute value is significant in many ways like for analysis purpose, graphical representation and decision making process. Memory required to represent a single row can be calculated as:

$$\text{Cost}(R) = \text{Cost(key attributes)} + \text{Cost(Static attributes)} + \text{Cost(Dynamic attributes)} + \text{Cost(Timestamps)} \quad (8)$$

$$= (K+S+D+T)\text{bytes} \quad (9)$$

Memory required to represent history data of a single row having time-varying attributes of frequency f is:

$$= (K+D+T)\times f \quad (10)$$

History data doesn't involve any static attributes and for every changes in dynamic attribute value insertion of a new tuple in the history relation is required. In terms of memory cost, this technique is more efficient as compared to single relation technique for capturing changes in attributes.

Tuple timestamp multiple historical relation data model

Non-temporal relation:

$$[\![(K_1]\!], K_2, K_3,...,K_n, S_1, S_2, S_3,..., S_n)$$

Current temporal relations: D_1 relation (time varying attribute):

$$[\![(K_1]\!], K_2, K_3,...,K_n, D_1, \text{Timestamp})$$

D_2 relation (time varying attribute):

$$[\![(K_1]\!], K_2, K_3,...,K_n, D_2, \text{Timestamp})$$

D_3 relation (time varying attribute):

$$(K_1, K_2, K_3,...,K_n, D_3, \text{Timestamp})$$

$D_n$relation (time varying attribute):

$$(K_1, K_2, K_3, ..., K_n, D_n, \text{Timestamp})$$

Memory required to represent a single row in temporal relation can be calculated as:

$$\text{Cost}(R) = \text{Cost}(\text{key attributes}) + \text{Cost}(\text{Static attributes}) = K+S \text{ bytes} \quad (11)$$

Memory required to represent a single row in temporal relation can be calculated as:

$$\text{Cost}(R) = \text{Cost}(\text{key attributes}) + \text{Cost}(\text{Dynamic attributes}) + \text{Cost}(\text{Times tamps}) = (K+D_i+T) \text{ bytes where } 1 \leq 1 \leq n \quad (12)$$

Total cost of temporal relation can be calculated as:

$$\text{Cost}(R) = (\text{Number of Dynamic attributes}) \times \text{Cost}(\text{key attributes}) + \text{Number of Dynamic attributes} = (K+T)*j + D \text{ bytes where j represents the number of dynamic attributes} \quad (13)$$

Memory required to represent history data of a single row having time-varying attributes of frequency f is:

$$= (K + D_i + T) \times f \text{ where } 1 \leq i \leq n \quad (14)$$

Improvement: Cost of historical relation data model with frequency (f) = (K+D+K)× f as in Eq. 10. Cost of TTMHR with frequency (f) = (K+D+T)×f as in Eq. 14:

$$= \frac{\cos t(\text{historical relation}) - \cos t(\text{TTMHR})}{\cos t(\text{historical relation})} \quad (15)$$

$$= \frac{(K+D+T)\times f - \{K+D_i+T)\times f}{(K+D+T)\times f} \quad (16)$$

$$= \frac{D - D_1}{K+D+T} \quad (17)$$

Improvement in memory cost lies on many parameters of time varying attributes like frequency, number of attributes and size of each attribute.

Result 1: Freezing the frequency parameter at 20 value of dynamic attributes vary from 12-44 bytes. We got the results as shown in Fig. 1. We did the experiment by freezing the frequency parameter at 20 and varying cost of dynamic attribute from 12-44 bytes.

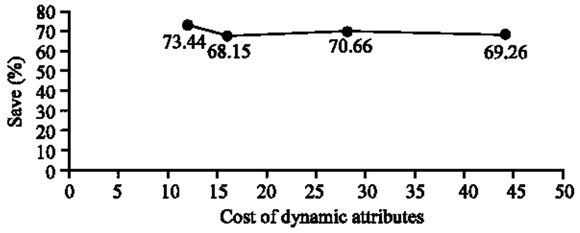

Fig. 1: Percentage of memory saved with 20 frequency

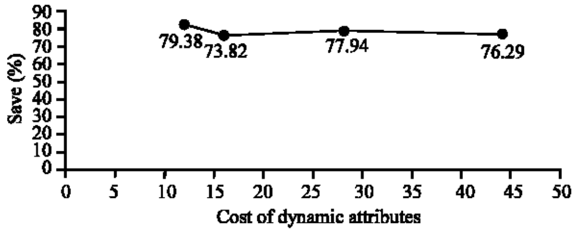

Fig. 2: Percentage of memory saved with frequency 55

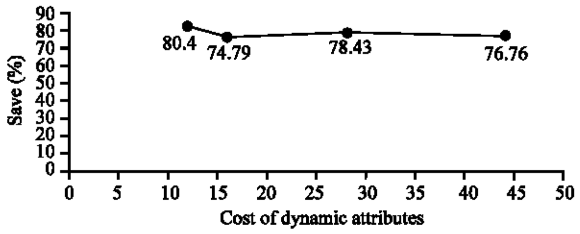

Fig. 3: Percentage of memory saved with frequency 78

Result 2: Freezing the frequency parameter at 55 value of dynamic attributes vary from 12-44 bytes. Figure 2 shows the experiment carried by freezing the frequency parameter at 55 and varying cost of dynamic attribute from 12-44 bytes.

Result 3: Freezing the frequency parameter at 78 value of dynamic attributes vary from 12-44 bytes. Figure 3 shows the experiment carried by freezing the frequency parameter at 78 and varying cost of dynamic attribute from 12-44 bytes.

Net cost of the data model: In this sub-segment, the researchers have analogized the TTMHR data model with the archived relation data model. The archived relation data model applies a single relation for holding the value of previous specimen of data. The analogy between the two models is derived with respect to the time elapsed

Table 1: Efficiency of the data model

Frequency (f)	Query	M_c of historical model	M_c of proposed model	Q_E of historical model	Q_E of proposed model	N_c of historical model	N_c of proposed model	Efficiency (save%)
20	Q1	1435	381	0.366	0.280	525.21	106.68	79.68
20	Q2	1435	457	0.351	0.286	503.68	130.70	74.05
20	Q3	1435	421	0.228	0.212	327.18	89.25	72.72
20	Q4	1435	441	0.322	0.309	462.07	136.26	70.50
55	Q5	3885	801	0.384	0.342	1491.84	273.94	81.63
55	Q6	3885	1017	0.398	0.377	1546.23	383.40	75.20
55	Q7	3885	857	0.360	0.344	1398.60	294.80	78.92
55	Q8	3885	921	0.385	0.362	1495.72	333.40	77.70
78	Q9	5495	1077	0.408	0.380	2241.96	409.26	81.74
78	Q10	5495	1385	0.344	0.332	1890.28	459.82	75.67
78	Q11	5495	1185	0.389	0.374	2137.55	443.19	79.26
78	Q12	5495	1277	0.428	0.396	2351.86	505.69	78.49

Table 2: Frequency with 20 values

Frequency (f)	Cost of dynamic attribute	Cost of historical relation data model (f)	Cost of TTMHR with (f)	Percentage save
20	12.00	16.00	28.00	44.00
20	1435.00	1435.00	1435.00	1435.00
20				
20	381.00	457.00	421.00	441.00
20	73.44	68.15	70.66	69.26

Table 3: Frequency with 50 values

Frequency (f)	Cost of dynamic attribute	Cost of historical relation data model (f)	Cost of TTMHR with (f)	Percentage save
55	12	16.00	28.00	44.00
55	3885	3885.00	3885.00	3885.00
55	801	1017.00	857.00	921.00
55	79.38	73.82	77.94	76.29

while executing a query (Kunzner and Petkovic, 2015; Atay, 2016) or storage cost. The query execution time may be defined as the time involved in executing the query processing plan. The range queries are applied to compare the data models. Each query is enforced exactly ten times and the mean value of the results obtained is weighed for comparison. The query execution time is calculated with the help of pgAdmin tool of Postgres.

Table 1-3 reflects the mean query execution-time of both the models under varying frequency of dynamic attributes. The time involved in executing most of the queries is optimized in the proposed tuple time-stamped multiple historical relation data model. The net cost of the data model depends on the parameters like query execution time or memory cost of the data models:

$$N_c = Q_E \times M_c \quad (18)$$

Where,

N_c = The Net cost of the data model,

Q_E = The mean Query execution time

M_c = The Memory cost of the data models

$$\text{Efficiency}(\text{save\%}) = \frac{N_c(\text{Historical model}) - N_c(\text{Proposed model})}{N_c(\text{Historicalmodel})} \times 100 \quad (19)$$

The queries Q1, Q6, Q8, and Q12 are used for equi-join operation on current or history relation. For inner join operation Q2, Q5, Q7, Q9 queries are used and queries Q3, Q4, Q10 and Q11 are used for union operation. It is clearly visible from the outcomes that the net cost of the proposed data model is less as compared to the historical relation data model and these results shows that the proposed model is efficient in terms of query execution time and memory cost. Query execution time of the proposed model is outperforms the conventional model when only the contemporary or archived table is considered for executing the query.

CONCLUSION

We proposed a data model which is proficient in handling time varying data for the time-oriented database. Multiple relations are used for the dynamic attributes that varies at different instants. This is done to remove redundancy from the database. The quintessential aspect as described by the authors in this study is 'indexing' which is applied on the frequently accessed non key attributes. The performance of the proposed temporal data model purely depends on the degree of heterogeneity of the time-varying attributes. The final outcome achieved after the empirical evaluation indicates that the proposed data model outperforms the historical relation data model as it diminishes the memory cost up to 81% as compared to the latter. The results prove that in the proposed model, there is substantial improvement of about 69-82% in the net cost of the data model. Moreover, it is also suitable for the large scale temporal databases. In fact, when the data set is very large then the probability of value of each attribute being distinct increases steeply. As a consequence, the window for overlapping region of values of different attributes shrinks significantly which helps in enhancing the efficiency of proposed system.

RECOMMENDATION

As a future direction, temporal aggregate functions may be applied to enhance the performance of the model.

REFERENCES

Anselma, L., P. Terenziani and R.T. Snodgrass, 2013. Valid-time indeterminacy in temporal relational databases: Semantics and representations. IEEE. Trans. Knowl. Data Eng., 25: 2880-2894.

Atay, C., 2016. An attribute or Tuple Timestamping in Bitemporal relational databases. Turk. J. Electr. Eng. Comput. Sci., 24: 4305-4321.

Bohlen, M.H., R. Busatto and C.S. Jensen, 1998. Point-versus interval-based temporal data models. Proceedings of the 14th International Conference on Data Engineering, February 23-27, 1998, IEEE, Orlando, Florida, pp: 192-200.

Chau, V.T.N. and S. Chittayasothorn, 2007. A temporal compatible object relational database system. Proceedings of the 2007 IEEE SoutheastCon, March 22-25, 2007, IEEE, Richmond, Virginia, pp: 93-98.

Edelweiss, N., J.P.M.D. Oliveira and B. Pernici, 1993. An Object-Oriented Temporal Model. In: Advanced Information Systems Engineering, Rolland, C., F. Bodart and C. Cauvet (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-56777-6, pp: 397-415.

Edelweiss, N., P.N. Hubler, M.M. Moro and G. Demartini, 2000. A temporal database management system implemented on top of a conventional database. Proceedings of the 20th International Conference on Chilean Computer Science Society (SCCC'00), November 16-18, 2000, IEEE, Santiago, Chile, pp: 58-67.

Galante, D.M.R., C.S.D. Santos, N. Edelweiss and A.F. Moreira, 2005. Temporal and versioning model for schema evolution in object-oriented databases. Data Knowl. Eng., 53: 99-128.

Garani, G., 2012. An algebra for the bitemporal nested data model. Proceedings of the 3rd International Conference on Advances in Information and Communication Technologies (ICT'12), November 23-25, 2012, ACEEE, Amsterdam, Netherlands, pp: 22-23.

Goralwalla, I.A., M.T. Ozsu and D. Szafron, 1998. A framework for temporal data models: Exploiting object-oriented technology. Proceedings of the 1998 Conference on Technology of Object-Oriented Languages and Systems, August 1, 1997, IEEE, Santa Barbara, California, pp: 16-30.

Halawani, S.M. and N.A. Al-Romema, 2010. Memory storage issues of temporal database applications on relational database management systems. J. Comput. Sci., 6: 296-304.

Halawani, S.M., I. Al-Bidewi, A.R. Ahmad and N.A. Al-Romema, 2012. Retrieval optimization technique for TUPLE timestamp historical relation temporal data model. J. Comput. Sci., 8: 243-250.

Jensen, C.S. and R.T. Snodgrass, 1996. Semantics of time-varying information. Inf. Syst., 21: 311-352.

Jensen, C.S. and R.T. Snodgrass, 1999. Temporal data management. Knowledge Data Eng., 11: 36-44.

Kaufmann, M., P.M. Fischer, N. May, C. Ge and A.K. Goel *et al*., 2015. Bi-temporal timeline index: A data structure for processing queries on bi-temporal data. Proceedings of the 2015 IEEE 31st International Conference on Data Engineering (ICDE'15), April 13-17, 2015, IEEE, Seoul, South Korea, ISBN:978-1-4799-7965-3, pp: 471-482.

Khatri, V., S. Ram, R.T. Snodgrass and P. Terenziani, 2014. Capturing telic/atelic temporal data semantics: Generalizing conventional conceptual models. IEEE. Trans. Knowl. Data Eng., 26: 528-548.

Kumar, S. and R. Rishi, 2016. Retrieval of meteorological data using temporal data modeling. Indian J. Sci. Technol., 9: 1-10.

Kumar, S. and R. Rishi, 2017. A new optimized model to handle temporal data using open source database. Adv. Electr. Comput. Eng., 17: 55-61.

Kunzner, F. and D. Petkovic, 2015. A Comparison of Different Forms of Temporal Data Management. In: Beyond Databases, Architectures and Structures, Kozielski, S., D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek and D. Kostrzewa (Eds.). Springer, Cham, Switzerland, ISBN:978-3-319-18421-0, pp: 92-106.

Kvet, M., K. Matiasko and M. Vajsova, 2015. Sensor based transaction temporal database architecture. Proceedings of the 2015 IEEE World Conference on Factory Communication Systems (WFCS'15), May 27-29, 2015, IEEE, Palma de Mallorca, Spain, ISBN: 978-1-4799-8244-8, pp: 1-8.

Mahmood, N., S.A. Burney, S.A. Ali, K. Rizwan and S.A.K. Bari, 2012. Fuzzy-temporal database ontology and relational database model. Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'12), May 29-31, 2012, IEEE, Sichuan, China, ISBN:978-1-4673-0025-4, pp: 573-577.

Petkovic, D., 2014. Performance issues concerning storage of time-variant data. Egyptian Comput. Sci. J., 38: 1-11.

Pilev, D. and A. Georgieva, 2012. Effective time temporal database model. Intl. J. Inf. Technol. Secur. N, 2: 33-46.

Safarik, J., 2011. Transformation of relational databases to transaction-time temporal databases. Proceedings of the 2nd Eastern European Regional Conference on Engineering of Computer Based Systems (ECBS-EERC'11), September 5-6, 2011, IEEE, Bratislava, Slovakia, ISBN:978-1-4577-0683-7, pp: 27-34.

Tansel, A.U., 2004. On handling time-varying data in the relational data model. Inf. Software Technol., 46: 119-126.

Terenziani, P., 2016. Irregular indeterminate repeated facts in temporal relational databases. IEEE. Trans. Knowl. Data Eng., 28: 1075-1079.

Yang, C., X. Wang, M. Zhang, R. Zheng and W. Wei *et al*., 2015. Standardization on bitemporal information representation in BCDM. Proceedings of the 2015 IEEE International Conference on Information and Automation, August 8-10, 2015, IEEE, Lijiang, China, ISBN:978-1-4673-9103-0, pp: 2052-2057.