(IJCSIS) International Journal of Computer Science and Information Security,
Vol. 14, No. 09, September 2016

Performance Evaluation of Attribute and Tuple Timestamping In Temporal Relational Database

Nashwan Alromema[1], Mohd Shafry Mohd Rahim[2]
[1]PhD Scholar, [2]Associate Professor
[1,2]Dept. of Software Engineering, Faculty of Computing
Universiti Teknologi Malaysia,
Johor, Malaysia.
[1]Nashwan.alromema@gmail.com

Ibrahim Albidewi[3]
[3]Professor
[3]Dept. of Information Science, Faculty of Computing
and Information Technology,
Jeddah, Saudi.
[3]Ialbidewi@kau.edu.sa

***Abstract*— Modeling temporal database over relational database using 1NF model is considered the most popular approach. This is because of the easy implementation as well as the modeling and querying power of 1NF model. In this paper, we compare a new approach for representing valid-time temporal database (in terms of structure and performance) to the main models in literature with attribute and tuple timestamping. The measurement of the performance is represented by the processing time to get the required temporal data as well as the size of the whole stored temporal data. A test has been performed by running sample queries for the same data in the represented models. Based on the tests, we have found that the new proposed model required less time and used less disk space. Therefore, it is more appropriate for modeling 1NF with interval-based timestamping in relational data model.**

Keywords-component; *Time-oriented data model; Time-varying data; interval-based event; point-time event; valid-time data; transaction time data;*

I. INTRODUCTION

Temporal Database (TDB) is a well-known database modeling technique for managing time-varying data. This modeling technique is considered as repository of time-dependent data. There has been a vast amount of work regarding developing temporal database applications starting from the 1970s [1]. Some of these works deal with storage structure and query processing as well as dozen-odd temporal DBMS prototypes [2 - 6]. The research work in [7] treats the problems of temporal databases models, integrity constraints, storage structures, and implementation techniques using different DBMS. A debate within the last three decades has been on how to model, implement and query temporal database in an efficient way [8]. Conventional relational databases store, and process the current valid-time data [2], commercial DBMS and standards for the query language do not fully support temporal features [3, 21].

Much of temporal data models have been proposed since the 1980s. These data models are categorized into temporal relational databases and temporal object-oriented databases. A study in [11, 19] demonstrate a new approach of implementing temporal database in XML platform. Modeling temporal database over relational model is one of the most popular approaches [24]. Such approaches are based on schema extension of Conventional Relational Model (CRM). There are two common approaches for these extensions. The most frequently stated approaches are tuple timestamping with First Normal Form (1NF), and attribute timestamping with Non-First Normal Form (N1NF). The study in [17] generalizes the models under 1NF approach into Tuple Timestamping Single Relation (TTSR), and Tuple Timestamping Multiple Relations (TTMR). TTSR approach introduces redundancy, where attribute values that change at different time are repeated in multiple tuples. However, TTMR approach has solved the problem of data redundancy in TTSR. The problem with this approach is that the fact about a real world entity is spread over several tuples in several relations. And combining the information for an object a variation of join known as temporal intersection join would be needed, which is generally expensive to be implemented. For N1NF, as stated in Jensen [6], there are some difficulties of temporal data models capturing an object in a single tuple such that "the models may not be capable of directly using existing relational storage structures or query evaluation techniques that depend on atomic attribute values". The study in [3] shows an approach of partial implementation of temporal database capabilities in top of widely used commercial DBMS. The model in this study is categorized under TTSR. This study also lacks most of temporal features as well as data redundancy of the proposed representational data model. The study in [10] shows an approach of temporal database representation in standard SQL under TTMR approach. The study explains a number of examples of temporal data and how temporal manipulations of such data can be effected by using standard SQL. A Column Level Temporal System (CLTS) proposed by Kvet in [20] is TTMR approach. The main issue of this model is to keep the duplicity of data minimal. As reducing the duplicities of the data is considered one of the important factors which improves processing speed to get a current snapshot and all data during the life cycle of the database object [22]. Atay and Tansel in [18] proposed the Nested Bitemporal Relational Data Model (NBRDM) under N1NF approach [18], NBRDM model attached bitemporal data to attributes and defined a bitemporal relational algebra and a bitemporal relational calculus language for the proposed data model [15, 16].

In this paper, we compare a new approach for representing valid-time temporal database with the main models in literatures (TTSR and TTMR). A comparison study is with

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

respect to the structure and performance. The simplicity as well as the ease of use are considered as other measures. The measurement of the performance is represented by the processing time to get the required data as well as the size of the whole stored temporal data. The implementations of the data models are performed in the most widely used commercial DBMSs (Oracle RDBMS). This paper utilizes the following concepts on temporal database theory: The representation of real world time is as a line. Every point in the line is referred to as an instance, a period is the time separating two instances, and an interval is the duration of loose segment of the time-line. Temporal data types in a temporal database can be identified as an instant of time, period and interval [7]. It is conceivable that time extends infinitely into the past or the future, as such when the relational database model has time introduced to it, it should be limited to delineate a particular time.

Granules are time points and the dividing scheme that splits the time line into a measurable collection of time segments is referred to as granularity, and is an aspect of all temporal information [12]. Temporal databases are depicted by the discrete time model because it is easy and comparatively simple to use [9]. Temporal databases have formulated a taxonomy of time which identifies when a particular event happens or when a given statement can be regarded at factual. User-defined time is one interpretation of the time feature employed in temporal databases. It is expressed in the data that is of the date/time kind (the birth date column for example), and does not suggest anything correlated to the validity of the other columns or temporal time; wherein the column(s) that contain date/time information types are employed to mark the related tuple's time aspects. There are three categories of temporal time: Valid time: Where in the related time is employed to determine when a particular statement (event-based) happened or when a particular statement (interval based) is regarded as being factual in the real world [13, 27]. Transaction time: The related time is in reference to the period when the data was actually retained inside the database. Bitemporal-time: The related time is connected to the yield of valid-time and transaction time in the model of bitemporal data. Tuples are regarded as valid at instances of that time by rollback databases [7, 8].

II. EXTENDED TEMPORAL DATABASE MODELS

In this section, the general technique of schema extension approach for modeling interval-based temporal database models in relational database will be discussed. The TTHR and TTMR will be used for the attribute-timestamped approach, whereas TTSR will be used for the tuple-timestamped approach.

A. TTHR Data Model

The methodology of representing temporal database in this paper is accomplished by using Schema extension approach of CRM. This approach is referenced as Tuple Timestamp Historical Relational (TTHR) data model. The proposed approach does not significantly change the procedures of designing and developing information systems. Figure 1 shows the conceptual structure of TTHR model. The database applications are directly connected to the main tables which hold the current valid time data. This feature gives the advantages that TTHR can be adapted to any functioning database systems without any changes to the infrastructure. The historical changes of each time-varying attributes in any table are stored in corresponding temporal database table (auxiliary tables) as shown in Figure 1. The data representation of temporal database in TTHR is accomplished by two steps: Firstly, defining the database object (entities /relations) for which we want to track the historical changes of the stored data, and 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 [14]. 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 model.

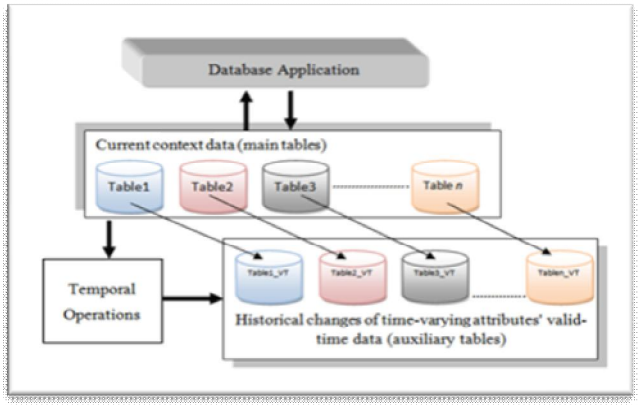

Figure 1. A Conceptual structure of TTHR data model

Figure 2(a) shows the schema structure of conventional non-temporal database for *EMPLOYEE* and *DEPARTMENT* database objects. Incorporating temporal aspects using TTHR (in this paper we will consider valid-time aspects [13]) to this database schema is shown in Figure 2(b).

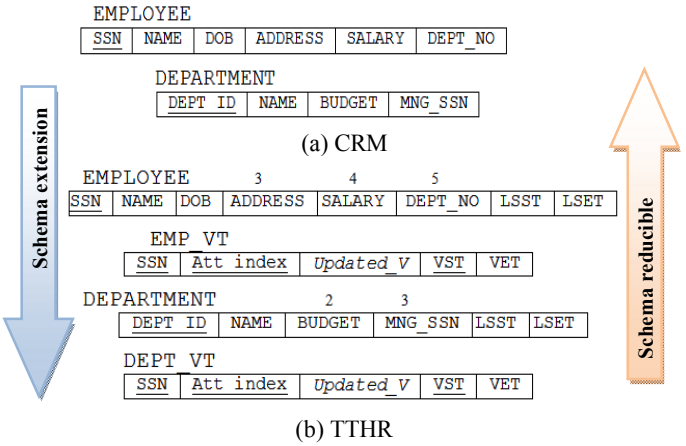

Figure 2. Schema structure of CRM and TTHR data model

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

The *EMPLOYEE* and *DEPARTMENT* tables in Figure 2(a), are extended to represent the temporal database using the proposed temporal data model (TTHR) as shown in Figure 2(b), Whereas the new table *Emp_VT* for the basic schema *EMPLOYEE* has the schema structure *Emp_VT (SSN, Att_index, upadated_v, VST, VET)*, these columns are identified as follows:

- *SSN* is the key attribute in the basic schema,
- *Att_index* attributes is used to identify the updated attributes in *EMPLOYEE* table, the value of this attribute is a function of time-varying attributes i.e. *Att_index(ADDRESS)=3*,*Att_index(SALARY)=4*, and *Att_index(DEPT_NO)=5*.
- Attribute *upadated_v* is used to store the old value of the updated attributes in the basic table.
- *VST* and *VET* are 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 2(b), *Emp_VT* has a primary key consists of the primary key of the basic table, *Att_index* column, and the *VST* column. For *EMP_VT* table, the primary key is (SSN,Att_index,VST). The data in the basic table (*EMPLOYEE*) keeps the latest updated data (current data), whereas *EMP_VT* stores the historical changes of the validity of the updated attributes in basic table.

B. TTSR Data Model

Temporal data modelling in the relational data model is accomplished by adding extra time attributes to the schema structure of CRM. The representation in Figure 3 incorporates time in a relational data model by adding extra timestamp columns to the conventional relation. These columns are **V**alid **S**tart **T**ime *(VST)* and **V**alid **E**nd **T**ime *(VET)* and indicate when the information in the corresponding tuple is valid, e.g. the period of time during which an employee *e* is affiliated to a specific department *d* (for simplicity in representation, the domains of *VST* and *VET* are considered as orders and isomorphic to the domain of natural numbers). Attributes *ADDRESS*, *SALARY*, and *DEPT_NO* in *EMPLOYEE* relation are considered as time-varying attributes, whilst the rest (SSN, NAME, and DOB) are considered as time-invariant attributes.

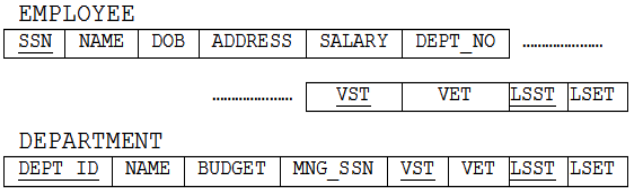

Figure 3. Schema structure of TTSR data model

This model has been used by many researchers such as in [7, 8]. We will refer to this representation by Tuple Timestamping Single Relation (TTSR) henceforth.

C. TTMR Data Model

Because of the data redundancy in TTSR, a trend by database researchers has been to normalize the relations in TTSR by decomposing the temporal relation as follows: time varying attributes are distributed over multiple relations, and time-invariant attributes are gathered into separate relations as in Figure 4. This model has been used in [10]. Figure 4 shows this temporal database representation. We will refer to this representation as Tuple Timestamping Multiple Relations (TTMR) henceforth.

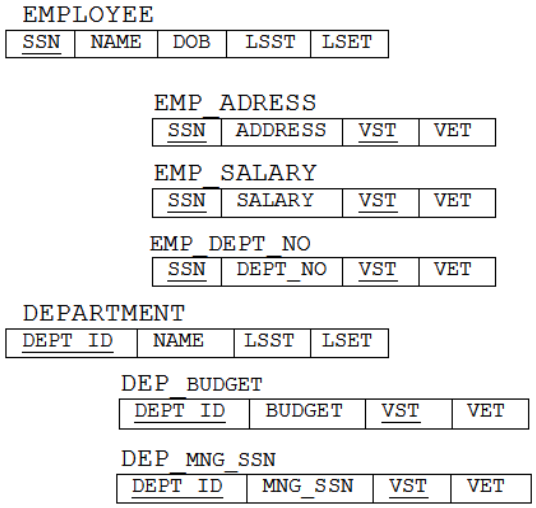

Figure 4. Schema structure of TTSR data model

Figure 5 shows an example of a temporal database (for *EMPLOYEE* table) represented by TTSR, TTMR, and TTHR models, respectively. A closed-closed representation for periods of validity is used in the three models, e.g. the time of validity of the first row in Fig 3(a) is [1, 5]. The natural numbers 1 and 5 are isomorphic to timestamp, i.e. 1 represents *'01/01/2001'* and so on [12]. To represent *Now* temporal variable, we have chosen a very large number such as 3000 to indicate the current validity of a specific row, e.g. the last row in Fig 3(a).

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

EMPLOYEE

SSN	NAME	DOB	ADDRESS	SALARY
89	Jon	01/04/1989	Houston	2,000
89	Jon	01/04/1989	Houston	4,000
89	Jon	01/04/1989	Houston	4,000
89	Jon	01/04/1989	Bolok	4,000
99	Richard	16/03/1990	FireOak	5,000
99	Richard	16/03/1990	FireOak	8,000
99	Richard	16/03/1990	FireOak	9,000

DEPT_NO	VST	VET	LSST	LSET
10	1	5	1	3000
10	5	15	1	3000
20	16	20	1	3000
20	21	3000	1	3000
20	5	9	5	3000
20	10	23	5	3000
20	24	3000	5	3000

(TTSR)

EMPLOYEE

SSN	NAME	DOB	LSST	LSET
89	Jon	01/04/1989	1	3000
99	Richard	16/03/1990	5	3000

EMP_ADRESS

SSN	ADDRESS	VST	VET
89	Houston	1	20
89	Bolok	12	3000
99	FireOak	5	3000

EMP_SALARY

SSN	SALARY	VST	VET
89	2,000	1	5
89	4,000	5	3000
99	5,000	5	9
99	8,000	10	23
99	9,000	24	3000

EMP_DEPT_NO

SSN	DEPT_NO	VST	VET
89	10	1	15
89	20	16	3000
99	20	5	3000

(TTMR)

EMPLOYEE

SSN	NAME	DOB	ADDRESS	SALARY	DEPT_NO	LSST	LSET
89	Jon	01/04/1989	Houston	6,000	10	1	3000
99	Richard	16/03/1990	FireOak	9,000	20	5	3000

EMP_VT

SSN	Att_index	*Updated_V*	VST	VET
89	4	2,000	1	5
89	5	10	1	15
89	3	Houston	1	20
99	4	5,000	5	9
99	4	8,000	10	23

EMP_Att_index_look_up

Att_index	Attribute_DDL
3	ADDRESS
4	SALARY
5	DEPT_NO

(TTHR)

Figure 5. The temporal data representaion of Employee relation in the different modeling approcahes

III. QUERY PROCESSING

Querying temporal database (using standard SQL2) is evaluated according to the supplying time tick to the query [24]. As stated by Snodgrass [7], querying temporal databases is classified into current query, sequenced query and non-sequenced query. Current query provides the current valid data which is in the basic schema table, while sequenced query provides the data that were valid during a certain interval of time where these data can be obtained from basic schema, temporal schema, or both depending on the complexity of the query. Non-sequenced query provides the historical changes of database objects' data. In this section, Query processing and evaluation of the proposed data model compared to the main models in literature (TTSR, and TTMR) will be demonstrated. In this study, Oracle SQL developer suite is used to evaluate and compare the performance of the queries in the three models. The query comparison consists of the 10 queries that are introduced in chapter 4 section 4.5.2.

A. Schemas of the three models

The schemas of the three models used in these benchmarks are as follows:

TTHR

Emp (SSN , *name* , *Birth _date* , *Address*, *Tel_no*, *Supervssn*, *Dno*, *Salary*, *Rank* , *LSST* , *LSET*)
Emp _VT (SSN , *Att _index* , *updated _V* , VST , VET)
Dept(D_numer, D_name, Mng_ssn, location,Budget, LSST, LSET)
Dept _VT (D_no , *Att _index* , *updated _V* , VST , VET)

TTSR

Emp (SSN , *name* , *Birth _date* , *Address*, *Tel_no*, *Supervssn*, *Dno*, *Salary*, *Rank* , VST , VET , LSST , LSET)
Dept(D_numer, D_name,Mng_ssn, location,Budget, VST, VET, LSST, LSET)

TTMR

Emp (SSN , *name* , *Birth _date*)
Emp _address (SSN , *address* , VST , VET)
Emp _Tel _no(SSN , *Tel _no*, VST , VET)
Emp _supervssn(SSN , *supervssn*, VST , VET)
Emp _Dno(SSN , *Dno*, VST , VET)
Emp _salary (SSN , *salary* , VST , VET)
Emp _Rank (SSN , *Rank* , VST , VET)
Emp _LS (SSN , LSST , LSET)
Dept (D_numer , *location*)
Dept _name(D_numer, D_name, VST, VET)
Dept _Mngssn(D_numer, Mng_ssn, VST, VET)
Dept _budget(D_numer, Budget, VST, VET)
Dept _Ls(D_numer, LSST, LSET)

The temporal variable *"Now or ∞"* is assumed as a very large value and isomorphic to 3000 (some authors assumed the temporal variable *"Now"* as equivalent to *Null* and some others assumed it as very large number).Three functions for time interval manipulations are used (Overlap, Min_P, Max_P) Algorithm 1, 2, 3. A coalesce function (Algorithm 4) for temporal query results manipulations also used for the queries under study.

Algorithm 1 Max time point Algorithm
Input: Two-time points t_1, t_2
Output: the maximum time point.

Begin
If ($t_1 > t_2$) then
t_1 ; ELSE t_2 ;
END if;
End Max time point;

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Algorithm 2 Min time point Algorithm.

Input: Two-time points t_1, t_2

Output: the minimum time point.

Begin

If ($t_1 < t_2$) then

t_1;

ELSE

t_2;

END if;

End Max time point;

Algorithm 3 overlap Algorithm

Input: Two-time intervals $[t_1, t_2]$, $[t_3, t_4]$.

Output: overlap or not overlap.

Begin

If $Max\_p(t_1, t_3) < Min\_p(t_2, t_4)$ then

Overlap;

ELSE

Not-Overlap;

END if;

End Overlap;

Algorithm 4 Coalesce Algorithm

Input : non-coalesce relation R (Query result)

Output: coalesced relation R_c.

Begin

$R \leftarrow R\ orderby\ (a_1, a_2, ...a_n, T_{vs}, T_{ve})$

$R_c \leftarrow \varnothing$

$Size \leftarrow Count(R)$;

$j = 1$;

Repeat

$R_j = (a_1, a_2, ...a_n, T_{vs}, T_{ve})$;

WHILE

$R_{j+1} = (a'_1, a'_2, ...a'_n, T'_{vs}, T'_{ve})$ AND

$(a_1 = a'_1, ...a_n = a'_n)$ AND $(T'_{vs} <= T_{ve} + 1)$

$T_{ve} = \max(T_{ve}, T'_{ve})$;

$j = j + 1$;

END WHILE;

$R_c \leftarrow R_c \cup \{a_1, a_2, ...a_n, T_{vs}, T_{ve}\}$;

$j = j + 1$;

WHILE $j < Size$;

RETURN R_c;

End Coalesce;

B. Non-sequenced query

The query suite in this paper consists of 10 English queries that cover the three categories of temporal database queries (Current, sequenced and non-sequenced queries)

Query 1: What are the descriptions of the current data of employee with SSN=2019?

Fragment SQL Code 1

TTHR	TTSR
`SELECT * FROM EMP WHERE SSN= 2019;`	`SELECT * FROM EMP WHERE SSN= 2019;`

TTMR

```
SELECT E.SSN, E.NAME, E.BIRTH_DATE,
AD.ADDRESS, TL.TEL_NO, SU.SUPPERVSSN ,
DN.DNO, S.SALARY, R.RANK, LS.LSST, LS.LSET
FROM EMP E, EMP_ADDRESS AD, EMP_TELNO TL,
EMP_SUPPERVSSN SU, EMP_DNO DN, EMP_SALARY S,
EMP_RANK R, EMP_LS LS
WHERE E.SSN = AD.SSN AND E.SSN = TL.SSN AND
E.SSN = SU.SSN AND E.SSN = DN.SSN AND E.SSN
= S.SSN AND E.SSN = R.SSN AND E.SSN = LS.SSN
AND AD.VET = 3000 AND TL.VET = 3000 AND
SU.VET = 3000 AND DN.VET = 3000 AND S.VET =
3000 AND R.VET = 3000 AND LS.LSET = 3000 AND
E.SSN = 2091;
```

Query 2: What are the descriptions of the latest valid data of life employees?

Fragment SQL Code 2

TTHR	TTSR
`SELECT * FROM EMP WHERE LSET = 3000;`	`SELECT * FROM EMP WHERE LSET = 3000;`

TTMR

```
SELECT E.SSN, E.NAME, E.BIRTH_DATE,
AD.ADDRESS, TL.TEL_NO, SU.SUPPERVSSN,
DN.DNO, S.SALARY, R.RANK, LS.LSST, LS.LSET
FROM EMP E, EMP_ADDRESS AD, EMP_TELNO TL,
EMP_SUPPERVSSN SU, EMP_DNO DN, EMP_SALARY S,
EMP_RANK R, EMP_LS LS
 WHERE E.SSN = AD.SSN AND E.SSN = TL.SSN AND
E.SSN = SU.SSN AND E.SSN = DN.SSN AND E.SSN
= S.SSN AND E.SSN = R.SSN AND E.SSN = LS.SSN
   AND AD.VET = 3000 AND TL.VET = 3000 AND
 SU.VET = 3000 AND DN.VET = 3000 AND S.VET =
   3000 AND R.VET = 3000 AND LS.LSET=30000;
```

Query 3: What are the descriptions of the latest valid data of not Life employees?

Fragment SQL Code 7.3

TTHR	TTSR
`SELECT * FROM EMP WHERE LSET <> 3000;`	`SELECT * FROM EMP WHERE E.LSET = (select max(E1.LSET) from EMP E1 where E1.SSN = E.SSN) AND LSET <> 3000;`

TTMR

```
SELECT E.SSN, E.NAME, E.BIRTH_DATE,
AD.ADDRESS, TL.TEL_NO, SU.SUPPERVSSN,
DN.DNO, S.SALARY, R.RANK, LS.LSST, LS.LSET
FROM EMP E, EMP_ADDRESS AD, EMP_TELNO TL,
EMP_SUPPERVSSN SU, EMP_DNO DN, EMP_SALARY S,
EMP_RANK R , EMP_LS LS
WHERE E.SSN = AD.SSN AND E.SSN = TL.SSN AND
E.SSN = SU.SSN AND E.SSN = DN.SSN AND E.SSN
```


https://sites.google.com/site/ijcsis/
ISSN 1947-5500

```
= S.SSN AND E.SSN = R.SSN AND E.SSN = LS.SSN
AND AD.VET = (select max(AD1.VET) from
EMP_ADDRESS AD1 where E.SSN = AD1.SSN)AND
TL.VET = (select max(TL1.VET) from EMP_TELNO
TL1 where E.SSN = TL1.SSN) AND
SU.VET = (select max(SU1.VET) from
EMP_SUPPERVSSN SU1 where E.SSN = SU1.SSN)
AND
DN.VET = (select max(DN1.VET) from EMP_DNO
DN1 where E.SSN = DN1.SSN) AND
S.VET = (select max(S1.VET) from EMP_SALARY
S1 where E.SSN = S1.SSN) AND
R.VET = (select max(R1.VET) from EMP_RANK R1
where E.SSN = R1.SSN) AND LS.LSET = (select
max(LS1.LSET) from EMP_LS LS1 where E.SSN =
LS1.SSN) AND LS.LSET  <> 3000;
```

Query 4: What is the latest current valid salary of employee *Ali* with SSN = *2091*.

Fragment SQL Code 7.4

TTHR	TTSR
`SELECT SSN, NAME, SALARY FROM EMP WHERE SSN = 2091;`	`SELECT E.SSN, E.NAME, E.SALARY FROM EMP E WHERE E.SSN = 2091 AND E.VET = (SELECT MAX(E1.VET) FROM EMP E1 WHERE E1.SSN = 2091);`

TTMR

```
SELECT E.SSN, E.NAME, S.SALARY
FROM EMP E, EMP_SALARY S
WHERE E.SSN = S.SSN AND E.SSN = 2091 AND
S.VET = (select max(S1.VET) from EMP_SALARY
S1 where E.SSN = S1.SSN);
```

Query 5: What is the latest current valid *name* of the department that employee Ali with *SSN = 2091* works on?

Fragment SQL Code 7.5

TTHR	TTSR
`SELECT E.SSN, E.Name, E.DNO, D.D_Name FROM EMP E, DEPT D where E.DNO = D.D_Number AND E.SSN = 2091;`	`SELECT E.SSN, E.Name, E.DNO, D.D_Name FROM EMP E, DEPT D where E.DNO = D.D_Number AND E.VET = (select max(E1.VET) from EMP E1 where E1.SSN = 2091) AND D.VET = (select max(D1.VET) from DEPT D1 where D1.D_Number = E.DNO) AND E.SSN = 2091;`

TTMR

```
SELECT E.SSN, E.NAME, DN.DNO, DNM.D_name
FROM EMP E, EMP_DNO DN, DEPT_D_Name DNM
WHERE E.SSN = DN.SSN AND DN.DNO =
DNM.D_Number AND
DN.VET = (select max(DN1.VET) from EMP_DNO
DN1 where DN1.SSN = E.SSN) AND
DNM.VET = (select max(DNM1.VET) from
DEPT_D_Name DNM1 where DNM1.D_Number =
DN.DNO) AND E.SSN = 2091;
```

Queries 1 to 5 are current queries. Querying the current valid data in TTHR and TTSR are the same. However, TTHR and TTSR need additional predicates for life/non-life data objects in where clause, TTMR costs a lot as the current valid data need to be collected from seven tables.

Query 6: What were the Salaries of All employees at any time?

For this query, TTHR's query and TTMR's query results are snapshot equivalent results, while TTSR's are not. Therefore, a new unary operator (*coalesce* function), is needed for querying the historical changes of time-varying attributes in TTSR. The coalescing operator (C) is a unary operator that merges value-equivalent tuples (tuples with mutually identical explicit attribute values) if the union of their timestamps is an interval [13, 14, 25, 26]. Temporal database in TTSR model is homogenous. Therefore, the query results of different time-varying attributes are not snapshot equivalent.

In TTHR representation, the current salaries are located in *EMP* relation. The historical changes of salary are located in *EMP_VT*. Then a set operation (Union) is used to combine the data from the main relation and the auxiliary relation. To simplify the query process, a view is created as shown in fragment code 6. For TTSR representation, since the model is homogeneous (value equivalence is allowed) the result might not be coalesced. Then, a view is created as shown in fragment code 6 in order to avoid non-coalesced data.

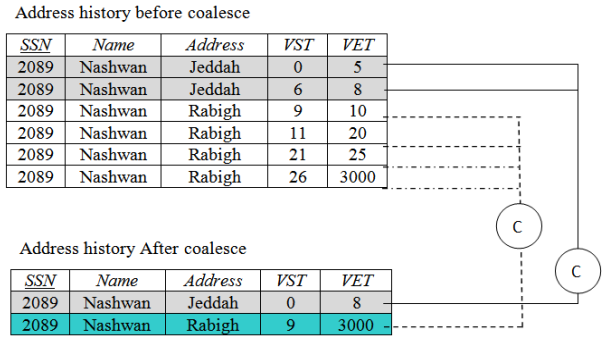

Figure 6. Address history of TTSR before and after Coalesce Function applied to query result

For example, retrieving the historical changes of *Nashwan's Address* in TTSR will result the data as shown in Figure 6, which violates the snapshot equivalent concept discussed by Bohlen [13]. The solution of that matter is to have a coalesce function. Coalesce function is denoted by C.

Note: A view for each time-varying attribute in both TTHR and TTSR will be created. These views will have the same names as the tables in TTMR.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Fragment SQL Code 6

TTHR

```
CREATE VIEW EMP_SALARY AS SELECT E.SSN,
E.SALARY,
MAX(CASE
  WHEN EV.VET IS NULL THEN
  E.LSST
  WHEN EV.VET IS NOT NULL
  AND E.LSST > EV.VET THEN
  E.LSST
  WHEN EV.VET IS NOT NULL
  AND
  E.LSST < EV.VET THEN
  (EV.VET +1) END) AS VST ,
  E.LSET AS  VET
FROM EMP E LEFT OUTER JOIN (SELECT EV1.SSN,
TO_NUMBER(EV1.UPDATED_V), EV1.VST, EV1.VET
FROM EMP_VT EV1 WHERE  EV1.ATT_INDEX = 7) EV
ON  E.SSN = EV.SSN
GROUP BY E.SSN, E.SALARY, E.LSET
UNION
SELECT  SSN,  TO_NUMBER(UPDATED_V),  VST,  VET
FROM EMP_VT WHERE  ATT_INDEX = 7;
```

TTSR

```
CREATE VIEW EMP_SALARY AS
SELECT F.SSN, F.SALARY, F.VST,L.VET FROM EMP
F, EMP L
WHERE F.VST < L.VET AND
F.SSN = L.SSN AND
F.SALARY = L.SALARY AND
 NOT EXISTS ( SELECT * FROM EMP
 M WHERE M.SSN=F.SSN AND
 M.SALARY = F.SALARY AND
 F.VST < M.VST AND
 (M.VST -1 ) <= L.VET AND
 NOT EXISTS ( SELECT * FROM EMP
 M1 WHERE M1.SSN=F.SSN AND
 M1.SALARY =  F.SALARY AND
 M1.VST < M.VST AND
(M.VST -1 ) <= M1.VET))AND
 NOT EXISTS ( SELECT * FROM EMP
 M2 WHERE M2.SSN = F.SSN AND
 M2.SALARY = F.SALARY AND
(( M2.VST < F.VST AND (F.VST -1) <= M2.VET )
OR
((M2.VST - 1 ) <= L.VET AND L.VET<M2.VET)));
```

Fragment SQL Code 7

TTHR	TTSR
SELECT SSN, SALARY, VST, VET FROM EMP_SALARY;	SELECT SSN, SALARY, VST, VET FROM EMP_SALARY;

TTMR

```
SELECT SSN, SALARY,VST, VET
FROM EMP_SALARYY;
```

Query 7: What are the historical change(s) of the Address of All employees?

This query is identical to query (6), but for time varying attribute Address, the same view as *EMP_salary* will be created for Address in TTHR and in TTSR with the same reason mentioned in query six.

Fragment SQL Code 8

TTHR	TTSR
SELECT SSN, ADDRESS, VST, VET FROM EMP_ADDRESS;	SELECT SSN, ADDRESS, VST, VET FROM EMP_ADDRESS;

TTMR

```
SELECT SSN, Address, VST, VET
FROM EMP_Address;
```

TTHR's query and TTMR's query results are a snapshot equivalent, while TTSR's are not. Therefore, a new unary operator which is called coalesce function is needed for querying the historical changes of time-varying attributes in TTSR. The coalescing operator (C) is a unary operator that merges value-equivalent tuples (tuples with mutually identical explicit attribute values) if the union of their timestamps is an interval [13]. Temporal database in TTSR model are homogeneous. Therefore, the query results of different time-varying attributes are not snapshot equivalent.

Query 8: Retrieve the historical change(s) of the Salaries and the Addresses of All Employees?

Fragment SQL Code 9

TTHR	TTSR
SELECT S.SSN, AD.ADDRESS, S.SALARY, MAX_P(S.VST, AD.VST) AS VST, Min_P(S.VET, AD.VET) AS VET FROM EMP_ADDRESS AD, EMP_SALARY S WHERE AD.SSN = S.SSN AND overlap(AD.VST, AD.VET, S.VST, S.VET) = 1 ;	SELECT S.SSN, AD.ADDRESS, S.SALARY, MAX_P(S.VST, AD.VST) AS VST, Min_P(S.VET, AD.VET) AS VET FROM EMP_ADDRESS AD, EMP_SALARY S WHERE AD.SSN = S.SSN AND overlap(AD.VST, AD.VET, S.VST, S.VET) = 1 ;

TTMR

```
SELECT S.SSN, AD.ADDRESS, S.SALARY,
MAX_P(S.VST, AD.VST) AS VST, Min_P(S.VET,
AD.VET) AS VET
FROM EMP_ADDRESS AD, EMP_SALARY S
WHERE AD.SSN = S.SSN AND
IV_overlap(AD.VST, AD.VET,  S.VST, S.VET) =
1 order by ssn,vst;
```


https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Query 9: What was the salary of Ali (SSN = 2091) when Jon (SSN = 2092) was a manager?

Fragment SQL Code 10

TTHR

```
SELECT S.SSN,
S.SALARY,
MAX_P(S.VST, DM.VST)
AS VST, Min_P(S.VET,
DM.VET) AS VET
FROM EMP_SALARY S,
     Dept_Mngssn DM
WHERE S.SSN = 2091
AND
      S.SSN = 2092
AND   overlap(S.VST,
S.VET,  DM.VST,
DM.VET) = 1 ;
```

TTSR

```
SELECT S.SSN,
S.SALARY,
MAX_P(S.VST, DM.VST)
AS VST, Min_P(S.VET,
DM.VET) AS VET
FROM EMP_SALARY S,
     Dept_Mngssn DM
WHERE S.SSN = 2091
AND
      S.SSN = 2092
AND   overlap(S.VST,
S.VET,  DM.VST,
DM.VET) = 1 ;
```

TTMR

```
SELECT S.SSN, S.SALARY, MAX_P(S.VST, DM.VST)
AS VST, Min_P(S.VET, DM.VET) AS VET
FROM EMP_SALARY S,Dept_Mng_ssn DM
WHERE S.SSN = 2091 AND
      DM.MNG_SSN = 2092 AND
      IV_overlap(S.VST, S.VET,  DM.VST,
DM.VET) = 1 ;
```

In this query, join is inevitable between *Emp_SALARY* and *Dept_Mngssn*. Query 10 requests the valid data of one employee's *salary* during the time of another employee when he was managing a department. This query uses join as well as interval comparison operator overlap.

Query 10: What were the Salaries of All employees during the time interval [10, 23]?

Fragment SQL Code 11

TTHR

```
SELECT S.SSN,
S.SALARY, S.VST,
S.VET
FROM EMP_SALARY S,
WHERE overlap(S.VST,
s.VET,  10,23) = 1 ;
```

TTSR

```
SELECT S.SSN,
S.SALARY, S.VST,
S.VET
FROM EMP_SALARY S,
WHERE overlap(S.VST,
s.VET,  10,23) = 1 ;
```

TTMR

```
SELECT S.SSN, S.SALARY, S.VST, S.VET
FROM EMP_SALARY S,
WHERE IV_overlap(S.VST, s.VET,  10,23) = 1 ;
```

In this query, TTHR and TTMR are identical because they request information for a valid time interval.

C. *Query Results analysis*

An experiment has been carried out on a temporal database with TTHR representation and the equivalent database for TTSR and TTMR representations as shown in Figure 5. The data set of this experiment has been randomly generated in the three models to simulate real-world scenarios (the same approach has been taken by Anselma [23]). The schema structure of each model has been depicted early in this section.

The experiments have been provided using SQL developer suite and Oracle11g running in virtual machine with Microsoft Windows XP, i7.3740QM CPU @2.70GB, 3.8 GB of RAM. The SQL Trace facility and TKPROF (Transient Kernel Profiler) are two basic performance diagnostic tools that have been used for queries analysis in the three approaches. TKPROF program outputs the parameters of each query as follows:

CPU = CPU time in seconds executing.
Elapsed = elapsed time in seconds executing.
Disk = number of physical reads of buffers from disk.
Query = number of buffers gotten for consistent read.

The data set consists of 108,004 instances of employees in *Emp* table. Queries from 1 to 10 have been run in sequence for each approach. Table I shows the experimental results of executing these queries for each Model. From Table I, Figures 7 and 8 have been plotted to compare the performance of each model in graphical view. It can be shown that TTSR satisfies good query performance in current query (Q1-Q5); the same performance is achieved by TTHR. However, TTMR costs a lot for current queries, but it costs less for both sequenced (Q6, Q7 and Q8) and non-sequenced (Q9, and Q10) queries and the same performance is achieved by TTHR. TTSR costs a lot for both sequenced and non-sequenced queries due to coalesce function that needs to be applied to the query results to make sure the query result is in snapshot equivalence.

SQL developer suite with TKPROF have been used for these experiments. Measuring the performance of the query by only running the query a few times is a pretty bad idea - equivalent to just accepting that the cost of the explanation plan tells you the best query. Therefore, it is really a need to take into account what resources query is taking up and therefore how it could affect the production system.

TABLE I. AN OUTPUT OF QUERY PROCESSING EXPERIMENTAL RESULTS

		TTHR				TTSR				TTMR			
Temporal	Query	CPU	Elapsed	Disk	Query	CPU	Elapsed	Disk	Query	CPU	Elapsed	Disk	Query
Current	Q1	0.00	0.01	3	3	0.00	0.01	1	4	0.00	0.04	15	25
	Q2	0.32	0.63	1193	8325	0.40	0.64	1251	8315	1.10	1.63	598	11467
	Q3	0.01	0.00	0	1199	0.01	0.01	0	1260	0.03	0.03	0	356
	Q4	0.00	0.00	0	3	0.00	0.00	0	7	0.00	0.00	0	6
	Q5	0.00	0.01	2	5	0.00	0.02	2	11	0.00	0.02	2	11
Non-sequenced	Q6	0.15	0.13	6	8332	2.43	1.63	0	351872	0.17	0.10	0	7552
	Q7	0.17	0.14	0	1206	2.17	1.60	0	351896	0.10	0.12	0	7672
	Q8	1.31	1.13	0	9538	5.84	4.03	0	696645	1.70	1.19	0	8054
Sequenced	Q9	0.01	0.03	12	18	0.01	0.02	5	30	0.00	0.01	6	12
	Q10	0.29	0.27	0	2869	1.03	0.73	0	95272	0.28	0.27	0	2038

The measurement of the performance is represented by the processing time to get the required data as well as the size of the whole stored temporal data. Figure 8 shows the number of buffer read by each query of each model. Although large disc capacity is currently available, there is still a need to effectively store and process data because temporal data are really extensive and contain changes of the object states over time.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

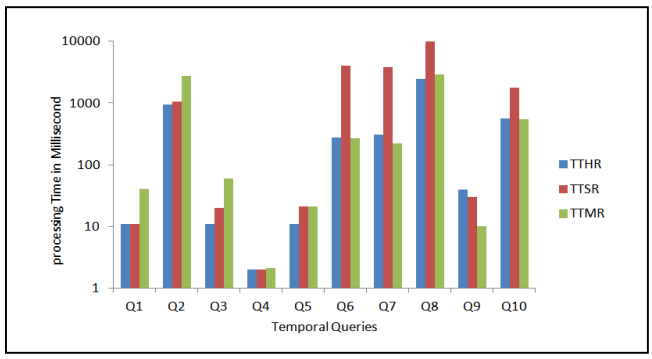

Figure 7. Query processing time for the 10 queries in the three models

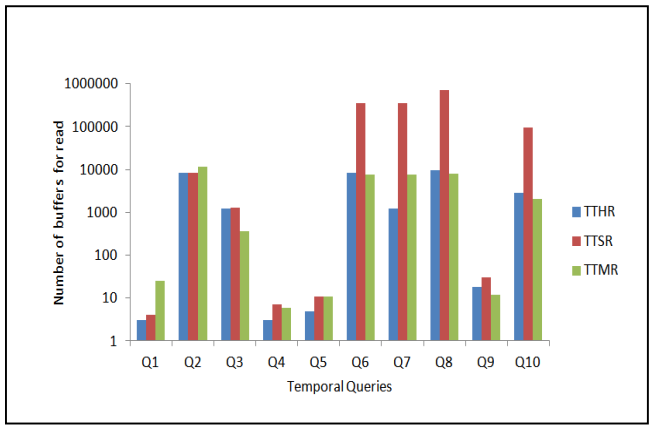

Figure 8. Number of Buffers read in the three models for the 10 Queries

IV. MEMORY STORAGE EFFICIENCY

The performance evaluation of the proposed model is considered in terms of memory storage efficiency. The Employees relation that is represented by the three models and the size in byte for the attributes in Employees relation is given as in Table II. The cost improvement of the memory storage is considered during one lifespan time and with a frequency of time-varying attributes update equal to 5. The results of memory storage efficiency for the three models are shown in Table II. As many parameters affect the cost improvements of TTHR over other models, Figure 9 shows the cost improvements where all the parameters have been fixed with varying the values of the frequency of time-varying attributes update from 5 to 440 times in a period of time. The memory efficiency has direct effect to the performance of the systems especially for image data as applications in [28, 29].

As shown in Figure 9, TTHR has achieved significant saving in storage memory space that ranges between 68%-81% over TTSR approach, and 10%-32% over TTMR that is based on the average change of the time varying attributes. TTHR has achieved some significant saving in storage memory space that is roughly equal or greater than TTMR. The proposed temporal data model is suggested for its simplicity as fewer database objects will be needed to capture the temporal aspects of time-varying data compared to TTMR. Moreover, applying TTHR to an existing database application does not require many changes compared to TTMR. Moreover, the only need is to create the auxiliary relation to capture the historical changes of time-varying attributes but without touching the system itself. This is contrary to TTMR, where the relations need to be decomposed and the integrity constraints need to be redefined.

TABLE II. COST MODEL OF *EMPLOYEES* RELATION REPRESENTED BY TTSR, TTMR AND TTHR.

Attribute name	Size/ Byte	Cost of data representation where the frequency of update = 5								
		TTSR			TTHR			TTMR		
		Snapshot	History	Total	Snapshot	History	Total	Snapshot	History	Total
SSN	9	9	27	36	9	27	36	63	45	108
Name	100	100	300	400	100	0	100	100	0	100
DOB	10	10	30	40	10	0	10	10	0	10
Address	20	20	60	80	20	0	20	9	9	18
Dept_no	3	3	9	12	3	0	3	3	6	9
Salary	6	6	18	24	6	0	6	6	12	18
Rank	1	1	3	4	1	0	1	1	0	1
VST	10	10	30	40	0	30	30	70	50	120
VET	10	10	30	40	0	30	30	70	50	120
LSST	10	10	30	40	10	0	10	10	0	10
LSET	10	10	30	40	10	0	10	10	0	10
Att_index	1	0	0	0	0	3	3	0	0	0
Updated_V	20	0	0	0	0	60	60	0	0	0
Total Cost				756			319			524

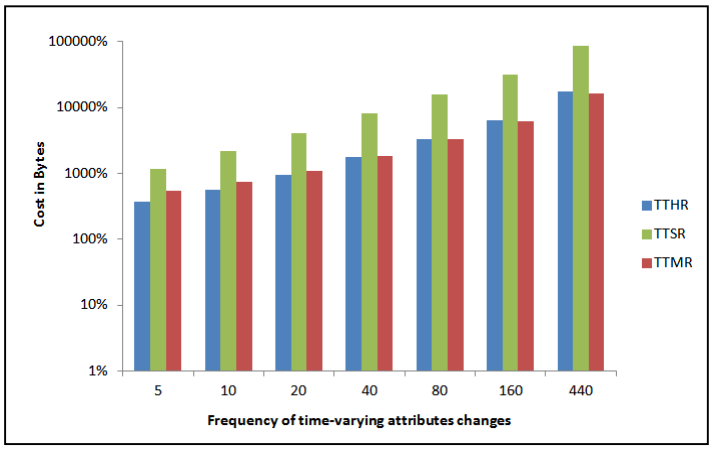

Figure 9. Memory storage cost efficiency of Employees relation represented by TTSR, TTMR and TTHR

V. CONCLUSION

A comparison study of TTHR with the main models in the literature with respect to the structure and performance has been demonstrated. The simplicity as well as the expressiveness of the model are considered as one measure. The measurement of the performance is represented by the processing time to get the required data as well as the size of the whole stored temporal data. TTHR satisfies good query performance in current query. The same performance is achieved by TTSR. However, TTMR costs a lot for current queries, but it costs less for both sequenced and non-sequenced queries and the same performance is achieved by TTHR. TTSR costs a lot for both sequenced and non-sequenced queries due to coalesce function that needs to be applied to the query results to make sure that the query result is in snapshot equivalence. This paper has also examined the representation of TTHR in the main models in literature namely, TTSR and

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

TTHR (expression power). It has been proved that TTHR has achieved significant saving in storage memory space that ranges between 68%-81% over TTSR approach, and 10%-32% over TTMR that is based on the average change of the time varying attributes. Finally, TTHR mimics TTMR in data representation by removing the needless redundancy of data and achieve better query processing for sequence and non-sequenced queries. Moreover, TTHR mimics TTSR in representing the current valid data in one relation, to benefit from querying the current snapshot data which costs a lot in TTMR.

REFERENCES

[1] Findler, N. V., & Chen, D. (1973). "On the problems of time retrieval of temporal relations causality, and coexistence". International Journal of Computer & Information Sciences, 2, 3, 161-185.

[2] Date, C. D., Darwen, H., & Lorentzos, N. A. (2003). "Temporal data and the relational data model". San Francisco: Morgan Kaufmann.

[3] Novikov, B. A., & Gorshkova, E. A. (2008). "Temporal databases: From theory to applications". Programming and Computer Software, 34, 1, 1-6. Pleiades Publishing, Ltd., 2008. Original Russian Text

[4] Tansel, A. U. (2004). "On handling time-varying data in the relational data model". Information and Software Technology, 46, 2, 119-126.

[5] Elmasri, R., and Navathe (2000). "Fundamentals of Database Systems". 3rd edition. Addison Wesley.

[6] Jensen, C. S., Clifford, J., Gadia, S. K., Segev, A., & Snodgrass, R. T. (1992). "A glossary of temporal database concepts". ACM Sigmod Record, 21, 3, 35-43.

[7] Snodgrass, R. T., (2000). "Developing Time-Oriented Database Applications in SQL", 1st edition, Morgan Kaufmann Publishers, Inc., San Francisco.

[8] Jensen, C. S., Snodgrass, R. T., & Soo, M. D. (1995). "The tsql2 data model", (pp. 157-240). Springer US.

[9] Patel, J. (2003). "Temporal Database System Individual Project". Department of Computing, Imperial College, University of London, Individual Project, 18-June-2003.

[10] Zimányi, E. (2006). "Temporal aggregates and temporal universal quantification in standard SQL". ACM SIGMOD Record, 35, 2, 16-21.

[11] Wang, F., Zhou, X., & Zaniolo, C. (2006, April). "Using XML to build efficient transaction-time temporal database systems on relational databases". In Proceedings of the 22nd International Conference on Data Engineering, 2006. ICDE'06 (pp. 131-131). IEEE.

[12] A-Qustaishat, M. (2001). "A visual temporal object-oriented model embodied as an expert C++ Library". ADVANCES IN MODELLING AND ANALYSIS-D-,6, 3/4, 3-43.

[13] Bohlen, M. H., Busatto, R., & Jensen, C. S. (1998, February). "Point-versus interval-based temporal data models". In Proceedings of 14th International Conference on Data Engineering, (pp. 192-200). IEEE.

[14] Dyreson, C., Grandi, F., Käfer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., ... & Wiederhold, G. (1994)."A consensus glossary of temporal database concepts".ACM Sigmod Record, 23, 1, 52-64.

[15] Tansel, A. U. (2006). "Modeling and Querying Temporal Data". Idea Group Inc.

[16] Tansel, A. U. (2004). "Temporal data modeling and integrity constraints in relational databases". In Computer and Information Sciences-ISCIS 2004 (pp. 459-469). Springer Berlin Heidelberg.

[17] Halawani, S. M., & Romema, N. A. (2010). "Memory storage issues of temporal database applications on relational database management systems". Journal of Computer Science, 6, 3, 296.

[18] Atay, C. (2016). "An attribute or tuple timestamping in bitemporal relational databases". Turkish Journal of Electrical Engineering & Computer Sciences. (2016) 24: (pp. 4305 – 4321). doi:10.3906/elk-1403-39.

[19] Noh, S.Y., Gadia, S.K. and Jang, H., (2013). "Comparisons of three data storage models in parametric temporal databases". Journal of Central South University, 20(7), pp.1919-1927.

[20] Kvet, M., Matiako, K. and Kvet, M., (2014). "Transaction management in fully temporal system". In Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on (pp. 148-153). IEEE.

[21] Snodgrass R, Ahn I. "Performance evaluation of a temporal database management system". Commun ACM 1986; 15:96-107.

[22] Arora, S. (2015). "A comparative study on temporal database models: A survey". In Advanced Computing and Communication (ISACC), 2015 International Symposium on (pp. 161-167). IEEE.

[23] Anselma, L., Stantic, B., Terenziani, P., and Sattar, A. (2013). "Querying now-relative data". Journal of Intelligent Information Systems, 41(2), 285-311.

[24] McKenzie Jr., and Snodgrass, R. T. (1991a). "Evaluation of relational algebras incorporating the time dimension in databases". ACM Computing Surveys (CSUR), 23(4), 501-543.

[25] Noh, S. Y., & Gadia, S. K. (2008). "Benchmarking temporal database models with interval-based and temporal element-based timestamping". Journal of Systems and Software, 81(11), 1931-1943.

[26] Petkovic, D., (2016). "Temporal Data in Relational Database Systems: A Comparison". In New Advances in Information Systems and Technologies (pp. 13-23). Springer International Publishing.

[27] Ab Rahman Ahmad, Nashwan AlRomema, Mohd Shafry Mohd Rahim, and Ibrahim Albidewi. "Temporal Database: An Approach for Modeling and Implementation in Relational Data Model." Life Science Journal 12.3 (2015).

[28] Rad, A. E., Rahim, M. S. M., Rehman, A., & Saba, T. (2016). Digital Dental X-ray Database for Caries Screening. 3D Research, 7(2), 1-5

[29] SaberiKamarposhti, M., Mohammad, D., Rahim, M. S. M., & Yaghobi, M. (2014). Using 3-cell chaotic map for image encryption based on biological operations. Nonlinear Dynamics, 75(3), 407-416.