A Mathematical Model for Comparing Memory Storage of Three Interval-Based Parametric Temporal Database Models

Nashwan Alromema[1]
[1]Department of Computer Science, Faculty of Computing and Information Technology-Rabigh, Rabigh, Saudi Arabia
[1]Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia

Mohd Shafry Mohd Rahim
Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia

Ibrahim Albidewi
Department of Computer Science, Faculty of Computing and Information Technology, Jeddah, Saudi Arabia

***Abstract*—Interval-Based Parametric Temporal Database Model (IBPTDM) captures the historical changes of database object in single tuple. Such data model violates 1NF and it is difficult to be implemented on top of conventional Database Management Systems (DBMS). The reason behind that, IBPTDM cannot directly use relational storage structure or query evaluation technique that depends on atomic attribute values as well as it is unfixed attribute size. 1NF model with its features can be used to solve such challenge. Modeling time-varying data in 1NF model raise a question about memory storage efficiency and ease of use. A novel approach for representing temporal data in 1NF model and compare it with other main approaches in literature is the main goal of this research. To this end, a mathematical model for comparing a three different storage models is demonstrated to illustrate that the proposed model is more efficient than other approaches under certain conditions. The simulation results showed that the proposed model overcomes the needless redundancy of data, achieves saving in memory storage, and it is easy to be implemented in relational data model or to be adapted with a production systems that need to track temporal aspects of functioning database Systems.**

***Keywords*—Valid-time data model; N1NF; tnterval-based timestamping; temporal data model; 1NF**

I. INTRODUCTION

Modeling temporal database is considered a vital and highly demanding problem. That is why varieties of techniques have been proposed to address this problem from different viewpoints [1]-[3]. Modeling temporal database in relational framework differs in many dimensions [4]-[11]. The most frequently stated approaches are tuple timestamping with First Normal Form (1NF), and attribute timestamping with Non-First Normal Form (N1NF). Based on the timestamp of the data, the first approach (1NF) has two distinctions namely, Tuple Timestamping Single Relation (TTSR), and Tuple Timestamping Multiple Relations (TTMR).

Models under TTSR approach are discussed by [1], [4], [5], [12]-[14]. An example of some of these temporal data models are LEGOL 2.0 by Jones [15], Temporally Oriented Data Model by Ariav [16], HSQL by Sarda [17], and TQuel by Snodgrass [18]. TTSR approach introduces redundancy, where attribute values that change at different time are repeated in multiple tuples. Furthermore, Steiner in [19] stated that, the main disadvantage of this approach is that the fact about a real world entity is spread over several tuples, where each tuple represents a state during a certain time period in the real world.

Models under TTMR approach have solved the problem of data redundancy in TTSR by decomposing the temporal relation as follows: time-varying attributes are distributed over multiple relations, and time-invariant attributes are gathered into separates relation. Many temporal data model discussed in literature are categorized under this approach [5]. An example of some of these temporal data models are Temporal Relational Model by Navathe and Ahmed [20], Snodgrass [18], Tansel [10], and Kvet [14]. The data models under this approach need a variation of join -known as temporal intersection join- that is used for combining the information for an object. Temporal intersection join is generally expensive to be implemented.

The second approach (N1NF) violates the atomicity of single data representations and based on the timestamp, the data can be timestamped in the level of tuple or in the level of attributes [5], [8]-[10]. An example of this approach is the parametric temporal data model that is based on attributes-timestamping and that uses a temporal element as a timestamp [21]. The bitemporal conceptual data model (BCDM) is another example of such approach that forms the basis for the temporal structured query language (TSQL) proposed by Jensen [5]. BCDM is based on tuple-timestamping and it uses interval-based timestamping [2].

Due to the needless redundancy of data in TTSR approach, expensive implementation in TTMR approach, and the implementation difficulty in parametric temporal data model in

top of relational data model, a new approach to model, implements, and query TDB in relational framework is proposed. The proposed approach is referenced as Tuple Timestamp Historical Relation (TTHR) [22]. This temporal data model (TTHR) is based on a tuple timestamping for the lifespan time of database objects, and it is also based on attributes timestamping for the historical valid time changes of time varying attributes. TTHR is in 1NF and it is an extension and reducible of Snodgrass (Tquel) temporal data model [14, 19]. TTHR mimics the features of TTSR and TTMR as well as the most common temporal database models discussed in literature.

Storage efficiency of temporal database systems has a direct impact to the system performance; therefore, in this study we will compare the three approaches (TTSR, TTMR, and TTHR) in terms of memory storage point of view. To measure the storage costs, we will establish mathematical model (formulas) for the three approaches. It can give us a reasonable judgment to determine whether TTHR is suitable for the implementation of the parametric temporal data model in top of conventional DBMS. Throughout our investigations into storage efficiency, we will show that the TTHR approach is comparable to the other approaches and it is even better under certain conditions. A similar study for calculating the efficiency of memory storage has been done by Atya in [2], this study compared the Snodgrass model (which is under TTMR-based approach in our study) with Tansel N1NF relational nested model. A study by Noh in [23] has introduced a new platform for modeling temporal database under XML-based platform. He compared the relational model as he named it (which is under TTSR-based approach in our study) with XML-based, and object-oriented based approach.

II. Interval-Based Parametric Temporal Database Model

Interval-based parametric temporal database model uses N1NF and attribute-timestamping data model with interval-based timestamps. The time interval $[t_1,t_2]$ consists of finite time points between t_1 and t_2 such that $t_1^2=[t_1,t_2]=\{t \in T \mid t_1 \le t \le t_2\}$, where T is defined as a set of time points in the domain D. Interval start point t_1 and interval end point and t_2 are the minimum and maximum boundary of the interval, both belong to the interval. Intervals can be defined as open, half-closed, or closed. In this study, Left-Right bounded (closed) representation for periods of validity is considered. Intervals can be compared to show their relative positions using Allen's interval logic [24], [25]. Fig. 1 shows *Emp* relation which represents the historical changes of employees' time-varying data with *Address*, *Tel_no*, *Supervssn*, *Dno* (department number), *Salary*, and *Rank*. *SSN*, *name*, and *Birth_date* is considered as time invariant attributes. It can be shown in Fig. 1 that the information about database object is modeled in one tuple and each time-varying attribute is timestamped by one or more time interval. An example of time intervals is [0 4] and [5 10] which timestamp the valid change time of address of employee *Nashwan*. The single time instance can be represented in interval-based, as an example time instance 10 has [10 10] representation.

Emp

SSN	Name	Birth_date	Address	Tel_no	Supervssn	Dno	Salary	Rank	T_{ls}
2091	Ali	1/10/1990	[0, ∞] Rabigh	[0, ∞] 098778776	[0, ∞] Null	[0, ∞] 10	[0, ∞] 13000	[0, ∞] C	[0, ∞]
2089	Nashwan	1/5/1980	[0, 4] Jeddah [5, 10] Rabigh	[0, 10] 059987767	[0, 6] Null [7, 10] 2091	[0, 8] 20 [9, 10] 30	[0, 2] 2000 [3, 9] 2500 [10, 10] 3000	[0, 10] A	[0, 10]

Fig. 1. Emp relation in interval-based parametric temporal data model.

The three data model (TTSR, TTMR, and TTHR) are based on schema extension approach of conventional relational data model and it can be implemented in conventional RDBMS.

III. Temporal Data Representations in Three Approaches

In this section, the *Emp* relation shown in Fig. 1 is going to be mapped to TTSR approach, TTMR, and TTHR approach (proposed model), respectively.

A. TTSR data representation approach

In TTSR, temporal database relations are in 1NF model with snapshot relations. To model *Emp* relation using TTSR representation, let R_{TTSR} represents the relation in TTSR representation that has the schema structure $R_{TTSR}=(A_K, A_U, A_C, T_{ls}, T_{le}, T_{vs}, T_{ve})$.

Emp

SSN	Name	Birth_date	Address	Tel_no	Supervssn	Dno	Salary	Rank	T_{vs}	T_{ve}	T_{ls}	T_{le}
2091	Ali	1/10/1990	Rabigh	098778776	2090	10	13000	C	0	∞	0	∞
2089	Nashwan	1/5/1980	Jeddah	059987767	Null	20	2000	A	0	2	0	10
2089	Nashwan	1/5/1980	Jeddah	059987767	Null	20	2500	A	3	4	0	10
2089	Nashwan	1/5/1980	Rabigh	059987767	Null	20	2500	A	5	6	0	10
2089	Nashwan	1/5/1980	Rabigh	059987767	2091	20	2500	A	7	8	0	10
2089	Nashwan	1/5/1980	Rabigh	059987767	2091	30	2500	A	9	9	0	10
2089	Nashwan	1/5/1980	Rabigh	059987767	2091	30	3000	A	10	10	0	10

Fig. 2. Emp relation in TTSR approach.

Fig. 2 demonstrates the *Emp* relation after transformation form parametric interval-based representation shown in Fig. 1. The evolution of data in *Emp* relation represented in TTSR approach is shown in Fig. 2. The semantic of update operation follows the temporal update operation introduced in literature [2], [3], [26]. The consequence of updating any time-varying attribute results in inserting a new tuple with the new updated values and new time points as shown in Fig. 2. Deleting any tuple is accomplished by updating T_{ls} to instant time point. The highlighted tuple with red color in Fig. 2 is an example of logical delete.

B. TTMR data representation approach

In TTMR, the relations are represented by: snapshot relation $R_{TTMR} = (A_K, A_U, T_{ls}, T_{le})$, for each time varying attribute there are separate relations $R_{A_{C_1}} = (A_K, A_{C_1}, T_{vs}, T_{ve}) \cdots R_{A_{C_i}} = (A_K, A_{C_i}, T_{vs}, T_{ve})$, and $R_{LS} = (A_K, T_{ls}, T_{le})$ for the lifespan time that are all in 1NF relations [3], [18], [21], [27], [28]. The relations in Fig. 3 show the representation of *Emp* relation in Fig. 1 using TTMR representation.

Emp

SSN	Name	Birth_date
2091	Ali	1/10/1990
2089	Nashwan	1/5/1980

Emp_LS

SSN	T_{ls}	T_{le}
2091	0	∞
2089	0	10

Emp_Address

SSN	Address	T_{vs}	T_{ve}
2089	Jeddah	0	4
2089	Rabigh	5	10
2091	Rabigh	0	∞

Emp_Telno

SSN	Tel_no	T_{vs}	T_{ve}
2089	098778776	0	10
2091	059987767	0	∞

Emp_Superssn

SSN	Superssn	T_{vs}	T_{ve}
2089	Null	0	6
2089	2091	7	10
2091	Null	0	∞

Emp_Dno

SSN	Dno	T_{vs}	T_{ve}
2089	20	0	8
2089	30	9	10
2091	10	0	∞

Emp_salary

SSN	Salary	T_{vs}	T_{ve}
2089	2000	0	2
2089	2500	3	9
2089	3000	10	10
2091	13000	0	∞

Emp_Rank

SSN	Rank	T_{vs}	T_{ve}
2089	A	0	10
2091	C	0	∞

Fig. 3. Emp relation in TTMR approach.

The temporal relation schema (*Emp*) in Fig. 3 that is corresponding to R_{TTMR} in TTSR, is decomposed into $i+2$ relations, where i (number of time-varying attributes) is equal to 6 and the 2 other relations are the lifespan relation and the relation that holds the time-invariant attributes. The 6 relations corresponding to each time-varying attribute that will be used to record the valid-time of the time-varying attributes in *Emp*. The lifespan relation will be used to track the changes of the lifespan of the objects in *Emp*. Finally, the non-temporal database relation is used to record the data of non-time-varying attributes.

C. TTHR data representation approach

In TTHR the general representation of R_T (temporal relational schema) is accomplished as two relations namely, R_T and R_T_VT, Where $R_T = (A_K, A_U, A_C, A_T)$, and R_T_VT is a new auxiliary relation schema that is created as $R_T\_VT = (A_K, Att\_index, \alpha, T_{vs}, T_{ve})$. Semantically the attributes of R_T_VT have the following meaning, Att_index: is a variable to identify the time-varying attribute A_{C_m} which begins updated such that $1 \le m \le j$. α is a new attribute that corresponds to attribute Updated_V as shown in Fig. 4. This attribute stores the updated value of any attribute in A_C set. T_{vs} : represents the Valid Start Time (VST). T_{ve} : represents the Valid End Time (VET).

The purpose of this representation is to keep the latest (current snapshot data) updated data in one relation R_T, and the historical changes of the validity of the time-varying data in the auxiliary relation R_T_VT [22, 29]. A relation instance is denoted by r_t, and r_t_vt, where $r_t(R_T)$ means r_t is an instance of R_T, and $vt\_r_t(R_T\_VT)$ means r_t_vt is an instance of R_T_VT. For tuples the symbols x, y and z can be used, thus a tuple, T_{ls} and T_{le} of that particular object (tuple), whilst the tuple(s) $r_t\_vt[x] = < a_K, Att\_index, \alpha, a_T >$ in the relation instance $r_t\_vt(R_T\_VT)$ is/are referencing to tuple x in r_t.

The tuple(s) in r_t_vt consist of the primary key of x, the identity(Att_index) of the time-varying attribute in x, the updated time-varying attributes value α in x, and the time validity of the updated attribute T_{vs} and T_{ve}. A subset of the domain of lifespan time is associated with each tuple in R_T shows that the existence of the object recorded by the tuple is true in the modeled reality during each lifespan chronon in that subset. A subset of domain of valid times is associated with each tuple in R_T_VT, represents the fact that the tuple $r_t\_vt[x]$ records the change of the validity of a_{C_m} in x. This fact is considered true in the modeled reality, such that the time of validity strictly contained in the time of the lifespan of x. Thus, the associated time with a tuple in TTHR is interval-based temporal timestamp. The tuples in r_t are timestamped by the lifespan time of the object denoted by t_{ls}, whereas the tuples in vt_r_t are timestamped by the valid-time denoted by t_v, both consisting of a temporal chronon in the time dimension spanned by lifespan and valid time.

Emp 1 2 3 4 5 6 7 8

SSN	Name	Birth_date	Address	Tel_no	Supervssn	Dno	Salary	Rank	T_{ls}	T_{le}
2091	Ali	1/10/1990	Rabigh	029876554	2090	10	17000	C	0	∞
2089	Nashwan	1/5/1980	Rabigh	059987767	2091	30	3000	A	0	10

Emp_VT

SSN	Att_index	Updated_V	T_{vs}	T_{ve}
2089	7	2000	0	2
2089	3	Jeddah	0	4
2089	5	Null	0	6
2089	6	20	0	8
2089	7	2500	3	9

Fig. 4. Emp relation in TTHR approach.

The example in the Fig. 4 uses two relations: *Emp*, describing employees information, such that, this relation is corresponding to R_T in TTHR, and the auxiliary relation *Emp_VT* that is used to record the changes of the validity of the time-varying attributes in *Emp* as well as the changes of the lifespan of the objects in *Emp*. The different types of attributes

of *Emp* and *Emp_VT* are: *Emp* relation: $A_K = \{SSN\}$, *i = 1*; $A_U = \{Name, Birth\_date\}$, *n = 2*; $A_C = \{Address, Tel\_no, Supervssn, D\_no, Salary, Rank\}$,*j=6*; $A_T = \{T_{ls}, T_{le}\}$ and *Emp_VT* relation: $Employees\_VT = \{SSN, Att\_index, Updated\_V, T_{vs}, T_{ve}\}$

As shown in Fig. 4, α is equivalent to *Updated_V* that stores the old value of the updated time-varying attributes. Att_index attribute stores the position of time-varying attributes location in the main relation *Employees*, such that the domain (Att_index) = {0,3,4,5,6,7,8} , where 0 is used to index the object's lifespan time and 3, 4, 5, 6, 7,and 8 are used to index the time-varying attributes as shown in Fig. 4. Granularity of chronon is assumed one month for both lifespan time and valid time. Integers are used as timestamp components that can be thought as dates, for example the integer 7 represents the date of *'April 2012'*.

IV. Discussion of Memory Storage Costs

In this section, we will formalize the storage costs of the three different approaches for representing the interval-based temporal data models in relational framework. The notations uses in this study are given in Table 1. Let R_T be a temporal relational schema with an arbitrary set of attributes $\{A_1, A_2, \cdots, A_n, T\}$, where these attributes can be classified into 4 groups: ***k***ey attributes, Time-invariant attributes (***U***nchangeable), Time-varying attributes (***C***hangeable), and ***T***imestamps (temporal) attributes. These groups can be represented by ***K***, ***U, C***, and ***T*** respectively.

Thus the schema of temporal relation can be redefined as $\{A_K, A_U, A_C, A_T\}$, where

$$A_K = \{A_{K_1}, A_{K_2}, \cdots, A_{K_j}\}$$
$$A_U = \{A_{U_1}, A_{U_2}, \cdots, A_{U_n}\}$$
$$A_C = \{A_{C_1}, A_{C_2}, \cdots, A_{C_i}\}$$
$$A_T = \{A_{T_1}, A_{T_2}\}$$

Definition 1: The cost of different attribute types are defined as:

$$\text{cost}(A_k) = \sum_{i=1}^{j} \text{cost}(A_{k_i}) = K \ byte \quad (1)$$
$$\text{cost}(A_U) = \sum_{i=1}^{n} \text{cost}(A_{U_i}) = U \ byte \quad (2)$$
$$\text{cost}(A_c) = \sum_{m=1}^{i} \text{cost}(A_{C_m}) = C \ byte \quad (3)$$
$$\text{cost}(A_T) = \sum_{i=1}^{2} \text{cost}(A_{T_i}) = T \ byte \quad (4)$$

Definition 2: The update frequency of time-varying attributes $A_{C_m} \in \{A_{C_1}, A_{C_2}, \cdots, A_{C_i}\}$ in a period of time is calculated as:

$$f(A_C) = \sum_{m=1}^{i} f(A_{C_m}) = \delta \ times \quad (5)$$

TABLE I. Notations

Symbol	Meaning
R_T	A temporal relational schema with an arbitrary set of attributes $\{A_K, A_U, A_C, A_T\}$
A_K	Set of ***k***ey attributes
A_U	Set of Time-invariant attributes(***U***nchangeable)
A_C	Set of Time-varying attributes (***C***hangeable)
A_T	The interval-based timestamp attribute.
J	total number of key attributes (A_K)
N	total number of Time-invariant attributes
I	total number of Time-varying attributes (A_C)
$f(A_{C_m})$	Update frequency of *m*-th time-varying attribute. Such that $A_{C_m} \in \{A_{C_1}, A_{C_2}, \ldots\ldots A_{C_i}\}$ and $1 \le m \le i$
$S(A_{\chi\gamma})$	A function to be defined on all the attributes in R_T , where $S(A_{\chi\gamma})$ returns the size of attribute $A_{\chi\gamma}$ in bytes. $\chi \in \{K, U, C, T\}$ and $\gamma \in \{1,2,\ldots, j\}$ (key attributes), $\gamma \in \{1,2,\ldots, n\}$ (time-invariant attributes), $\gamma \in \{1,2,\ldots, i\}$ (time-varying attributes) or $\gamma \in \{1,2\}$ (timestamping attributes)
$Cost(A_\chi)$	A function to be defined on the subset attributes χ , where $\chi \in \{K, U, C, T\}$ and return the size of the attributes group in byte.
$Cost(z)$	The cost of a tuple(row) z in relation instance r_t is the summation of the cost of all subsets attributes equals to $\text{cost}(A_K) + \text{cost}(A_U) + \text{cost}(A_C) + \text{cost}(A_T)$.
K	Cost of ***k***ey attributes
U	Cost of Time-invariant attributes(***U***nchangeable)
C	Cost of Time-varying attributes (***C***hangeable)
T	Cost of ***T***imestamps (temporal) attributes
χ	A tuple in a temporal relation
z	A tuple in a temporal relation

A. TTSR data representation approach

The temporal relation in TTSR can be represented as:

$$R_{TTSR}(A_{k_1}, \cdots, A_{k_j}, A_{U_1}, \cdots, A_{U_n}, A_{C_1}, \cdots, A_{C_m}, A_T),$$

To calculate the memory storage efficiency of interval-based temporal database relation represented by TTHR approach, a general formula is constructed for calculating the size of a single tuple in a temporal relation. The cost of representing one tuple x in relation instance $r(R_{TTSR})$ is calculated as:

$$\begin{aligned} \text{Cost}(x) &= \text{cost}(A_K) + \text{cost}(A_U) + \text{cost}(A_C) + \text{cost}(A_T) \\ &= K + U + C + 2T \ byte \end{aligned} \quad (6)$$

as stated in (1), (2), (3) and (4) $\text{cost}(A_T) = 2T$, because the tuple in TTSR will be timestamped by valid time and lifespan time. The cost of storing the history of the changes of A_C with $f(A_C) = \delta$ times in a period (lifespan interval) of time λ is calculated as:

$$= \delta(K + U + C + 2T) \quad (7)$$

An update in any A_C requires the insertion of a new row with all attributes. Using (6) and (7), the memory storage cost of one object represented by TTSR approach can be defined as:

$$\text{Cost}(TTSR) = (K + U + C + 2T) + \delta(K + U + C + 2T) \quad (8)$$

B. TTMR data representation approach

The temporal relation in TTMR is represented as:

R_{TTMR} ($A_{k1}, \cdots, A_{kj}$, $A_{u1}, \cdots, A_{un}$)

$R_{A_{c1}}$ ($A_{k1}, \cdots, A_{kj}$, A_{c1} , A_T)

$R_{A_{c2}}$ ($A_{k1}, \cdots, A_{kj}$, A_{c2} , A_T)

$R_{A_{c3}}$ ($A_{k1}, \cdots, A_{kj}$, A_{c3} , A_T)

$R_{A_{c4}}$ ($A_{k1}, \cdots, A_{kj}$, A_{c4} , A_T)

……

$R_{A_{ci}}$ ($A_{k1}, \cdots, A_{kj}$, A_{ci} , A_T).

To calculate the memory storage efficiency of interval-based temporal database relation represented by TTHR approach, a general formula is constructed for calculating the size of a single tuple in a temporal relation. The cost of storing one tuple x in relation instance $r(R_{TTMR})$ is calculated as stated in (1), (2), (3) and (4), as follows:

$$\begin{aligned}\text{Cost}(x) &= \text{Cost}(A_k) + \text{Cost}(A_u) + \text{Cost}(A_T) + \\ &\quad \sum_{m=1}^{i}\left[S(A_{c_m}) + \text{Cost}(A_k) + \text{Cost}(A_T)\right] \\ &= K + U + T + \sum_{m=1}^{i} S(A_{c_m}) + \sum_{m=1}^{i} K + \sum_{m=1}^{i} T \\ &= (K + U + T) + i(K + T) + \sum_{m=1}^{i} S(A_{c_m})\end{aligned}$$

Since

$$\begin{aligned}\text{Cost}(A_C) &= i(K + T) + \sum_{m=1}^{i} S(A_{c_m}) \\ &= i(K + T) + C\end{aligned}$$

Then the $\text{Cost}(x)$ can be represented as:

$$\text{Cost}(x) = K + U + T + C + i(K + T)$$

$$= K(i+1) + U + C + T(i+1) \quad byte \quad (9)$$

The variable *i* represents the total number of time varying attributes A_C .The cost of storing the history of the changes of each A_{c_m} with $f(A_{c_m}) = \delta_m$ times in a period/interval of time λ can be calculated as:

$$\begin{aligned}&= \sum_{m=1}^{i} \delta_m \left(S(A_{c_m}) + K + T\right) \\ &= \sum_{m=1}^{i} \delta_m S(A_{c_m}) + (K + T)\sum_{m=1}^{i} \delta_m\end{aligned}$$

Since $\sum_{m=1}^{i} \delta_m = \delta$, as in Eqns. 5, then the equation becomes:

$$= \sum_{m=1}^{i} \delta_m S(A_{cm}) + \delta(K + T) \quad (10)$$

Using (9) and (10), the memory storage cost of one object represented by TTMR approach can be defined as:

$$\begin{aligned}&\text{Cost}(\text{TTMR}) = \text{K}(\text{i}+1) + \text{U} + \text{C} + \text{T}(\text{i}+1) + \\ &\sum_{\text{m}=1}^{\text{i}} \delta_{\text{m}} \text{S}(\text{A}_{\text{cm}}) + \delta(\text{K} + \text{T})\end{aligned} \quad (11)$$

C. TTHR data representation approach

In TTHR model, the temporal relation schema is represented by R_{TTHR} and R_{VT} as shown below:

R_{TTHR} ($A_{K_1}, \cdots, A_{K_j}$, $A_{U_1}, \cdots, A_{U_n}$, $A_{C_1}, \cdots, A_{C_m}$, A_T),

R_{VT} ($A_{K_1}, \cdots, A_{K_j}$, Att_index , α , A_T). To calculate the memory storage efficiency of interval-based temporal database relation represented by TTHR approach, a general formula is constructed for calculating the size of a single tuple in a temporal relation. The cost of representing one tuple x in relation instance $r(R_{TTHR})$ is calculated as:

$$\begin{aligned}\text{Cost}(\text{x}) &= \text{cost}(\text{A}_{\text{k}}) + \text{cost}(\text{A}_{\text{u}}) + \text{cost}(\text{A}_{\text{c}}) + \text{cost}(\text{A}_{\text{T}}) \\ &= \text{K} + \text{U} + \text{C} + \text{T byte}\end{aligned} \quad (12)$$

As stated in (1), (2), (3) and (4), the cost of storing the history of changes of A_C with $f(A_C) = \delta$ times in a period/interval (lifespan interval) of time λ can be calculated as:

$= \delta(K + S(Att\_index) + S(\alpha) + T)$, since the size of *Att_index* = 1, and

$S(\alpha) = Max\left(S(A_{C_1}), S(A_{C_2}), \cdots, S(A_{C_i})\right)$, let $\beta = S(\alpha)$, then

$$= \delta(K + 1 + \beta + T) \quad (13)$$

Such that, Att_index: is an attribute to index the time-varying attributes with one byte size. α : is a new added attribute of variant data type to hold data from different types. Its size is assumed to be the same size of the largest field size in A_C . The size of α in byte is $S(\alpha) = Max\left(S(A_{C_1}), S(A_{C_2}), \cdots, S(A_{C_i})\right)$, using (12) and (13), the memory storage cost of one object represented by TTHR approach can be defined as:

$$\text{Cost}(TTHR) = (K + U + C + T) + \delta(K + 1 + \beta + T) \quad (14).$$

V. Comparisons of Memory Storage Cost and Result Analysis of the Three Approaches

In this section, we will mimic the storage cost of the three models based on various settings of the parameters that have direct impact to the temporal data storage. The Default values are initiated with consideration of general cases as follows: *K*=

9, U= 110, C=37, T=20, β *=9*. Fig. 5 shows the memory storage cost for the initial values for the different parameters that construct the temporal relation. For these values, TTSR-based approach shows worse storage costs comparing to TTMR-based and TTHR-based approaches. However, the graph shows a positive indication that TTHR can be used as an efficient storage that is better than TTMR-based approach until the value of δ = 40. After this point it seems that both TTHR and TTMR have the same storage efficiency.

Fig. 6 shows the storage costs of the temporal relational approach after freezing all the parameters and varying the sizes of the time-varying attributes (C). For these values, TTSR-based approach shows worse storage costs comparing to TTMR-based and TTHR-based approaches. However, the graph shows a positive indication that TTHR can be used as an efficient storage that is better than TTMR-based approach until the value of C = 150 byte. After this point it seems that both TTHR and TTMR have the same storage efficiency.

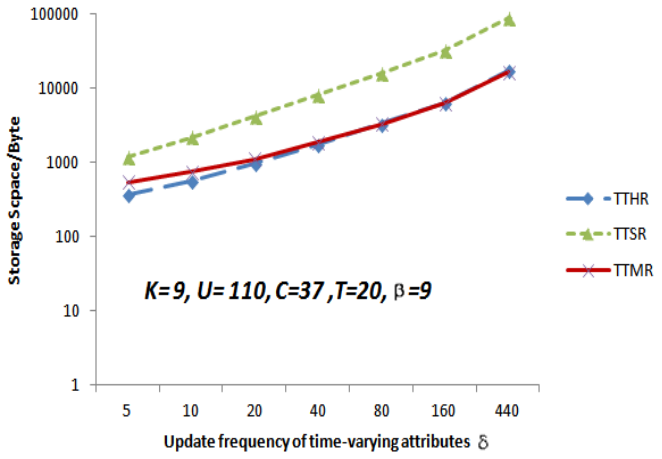

Fig. 5. Storage cost for the update frequency (δ) variations.

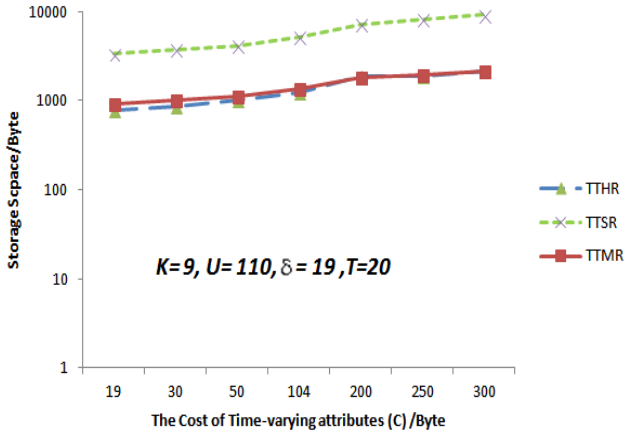

Fig. 6. Storage Cost for the Time-varying attributes' size (C) variations.

Fig. 7 shows the storage efficiency after freezing all the parameters and varying the sizes of key attributes (K) value variations.

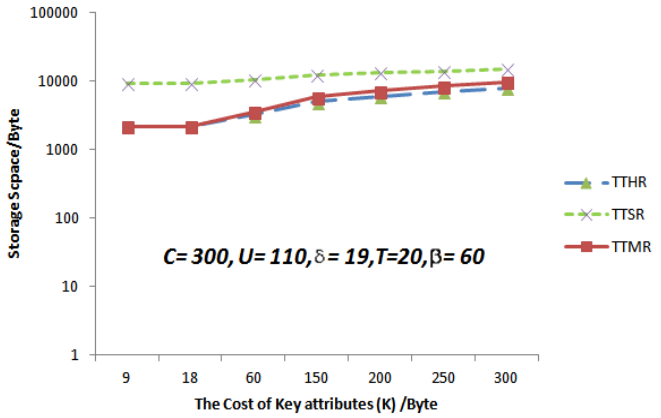

Fig. 7. Storage Cost for the Key attributes' size (K) variations.

We increase K value from 9 to 300 bytes. As we can see, the TTHR-based approach shows the best storage efficiency than the others. However, it is shown that the difference of storage efficiency is marginal between the TTHR-based approach and the TTMR-based approach.

A similar study for calculating the efficiency of memory storage has been done by Atya in [2], this study compared the Snodgrass model (which is under TTMR-based approach in our study) with Tansel N1NF relational nested model. A study by Noh in [23] has introduced a new platform for modeling temporal database under XML-based platform. He compared the relational model as he named it (which is under TTSR-based approach in our study) with XML-based, and object-oriented based approach.

VI. Conclusion

A new approach for representing temporal database in relational data model has been demonstrated in this research work. A comparison study of the proposed model (TTHR) with the main models in literature (TTSR and TTMR) with respect to the memory storage efficiency has been mathematically illustrated. To measure the storage costs, we have established a mathematical model (formulas) for the three approaches. The measurement of the performance is represented by the size of the whole stored temporal data as stated in [22], [29], [31]. It has been proved that TTHR has achieved significant saving in memory storage that ranges between 68%-81% over TTSR approach, and 10%-32% over TTMR. The memory storage save is based on the average change of the time varying attributes [29], [30], [31]. A validation and verification study of the correctness and the expressiveness of TTHR model has been depicted in [32]. Finally, TTHR mimics TTMR in data representation by removing the needless redundancy of data. 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 as stated in [22].

Acknowledgment

This paper was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University. The authors, therefore, acknowledge with thanks to DSR's technical support.

REFERENCES

[1] Anselma, L., Bottrighi, A., Montani, S., & Terenziani, P. Extending BCDM to cope with proposals and evaluations of updates. IEEE T KNOWL DATA EN, IEEE Transactions on 2013; 25(3), 556-570.

[2] Atay, C. An attribute or tuple timestamping in bitemporal relational databases. TURK J ELECTR ENG CO 2016; 24: (pp. 4305 – 4321). doi:10.3906/elk-1403-39

[3] Elmasri, R., and Navathe. Database Systems 6th edition. Pearson, 2011..

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

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

[6] Tansel, A. Adding time dimension to relational model and extending relational algebra. INFORM SYST 1986; 11(4), 343-355.

[7] Tansel, A., and Snodgrass, R. *Temporal databases: Theory, design and implementation.* Redwood City, CA: Benjamin Cummings, 1993.

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

[9] Tansel, A. On handling time-varying data in the relational data model. INFORM SOFTWARE TECH 2004; 46(2), 119-126.

[10] Tansel, A. Modeling and Querying Temporal Data, Idea Group Inc 2006.

[11] Terenziani, P. Coping with events in temporal relational databases. . IEEE T KNOWL DATA EN, IEEE Transactions on 2013; 25(5), 1181-1185.

[12] Garani, G. (2003). *A temporal database model using nested relations* (Doctoral dissertation, BIRKBECK COLLEGE).

[13] Garani, G. A generalised relational data model. International Journal of Computer Systems Science and Engineering 2007; 11, 43-59.

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

[15] Jones S., Mason P., and Stamper, R. A Relational Specification Language for Complex Rules. Information Systems 1979; 4(4):293{305}.

[16] Ariav, G. A temporally oriented data model. ACM T DATABASE SYST 1986; 11(4), 499-527.

[17] Sarda, N. L. Algebra and query language for a historical data model. The Computer Journal 1990; 33(1), 11-18.

[18] Snodgrass, R. The Temporal Query Language TQUEL. ACM T DATABASE SYST 1987; 12(2):247{298, June 1987.

[19] Steiner, A. A Generalization Approach to Temporal Data Models and their Implementations. A dissertation submitted to the SWISS FEDERAL INSTITUTE OF TECHNOLOGY, ZURICH, 1998.

[20] Navathe, S. B., and Ahmed, R. A temporal relational model and a query language. INFORM SCIENCES 1989; 49(1), 147-175.

[21] Noh, S.Y., and Gadia, S. K. *Benchmarking temporal database models with interval-based and temporal element-based timestamping*. J SYST SOFTWARE 2008; 81(11):1931–1943.

[22] Alromema, N., Rahim, M.S.M. and Albidewi, I., 2016. Performance Evaluation of Attribute and Tuple Timestamping In Temporal Relational Database. International Journal of Computer Science and Information Security, 14(9), p.374.

[23] Noh, S.Y., Gadia, S.K. and Jang, H. Comparisons of three data storage models in parametric temporal databases. J CENT SOUTH UNIV 2013;20(7), pp.1919-1927.

[24] Allen, J.F. Maintaining knowledge about temporal intervals. *Communications of the ACM* 1983; *26*(11), 832-843.

[25] Yang, C., Wang, X., Zhang, M., Zheng, R., Wei, W. and Lou, Y. Standardization on bitemporal information representation in BCDM. In Information and Automation, 2015 IEEE International Conference on (pp. 2052-2057). IEEE.

[26] Snodgrass, R. *Developing Time-Oriented Database Applications in SQL*, 1st edition, Morgan Kaufmann Publishers, Inc., San Francisco, 2000.

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

[28] Snodgrass, R. Temporal Databases. IEEE Computer 1986; 19(9):35{42}.

[29] Halawani, S. M., Al-romema, N. A. *Memory Storage Issues of Temporal Database Applications on Relational Database Management Systems***,** Journal of Computer Science 2010; 6, (3): 296-304

[30] Ab Rahman Ahmad, N.A., Rahim, M.S.M. and Albidewi, I., 2015. Temporal Database: An Approach for Modeling and Implementation in Relational Data Model. Life Science Journal, 12(3).

[31] Halawani, S.M., AlBidewi, I., Ahmad, A.R. and Al-Romema, N.A., 2012. Retrieval optimization technique for tuple timestamp historical relation temporal data model. Journal of Computer Science, 8(2), p.243.

[32] Alromema, N.A., Rahim, M.S.M. and Albidewi, I., 2016. Temporal Database Models Validation and Verification using Mapping Methodology. VFAST Transactions on Software Engineering, 11(2), pp.15-26.