1. **Introduction.** Modeling temporal database is considered a vital and highly demanding problem. That is why a variety of techniques have been proposed to address this problem from different viewpoints [2], [5], [6]. Modeling temporal database in relational framework differs in many dimensions [4], [10], [23]-[28]. 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 time stamp 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 [2]-[4], [7], [8], [10]-[12], [20]. Many temporal data model discussed in literature are categorized under this approach as in [10], [12], [14], [20], [27]. The second approach (N1NF) violates the atomicity of single data representations and based on the time stamp, the data can be timestamped in the level of tuple or in the level of attributes [10], [25]-[27]. An example of this approach is the parametric temporal data model [15]. 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, 10].

A new approach to model, implement, and query TDB in relational framework is proposed in our previous work in [9] [29]. The proposed approach is referenced as Tuple Timestamp Historical Relation (TTHR). 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 mimics the features of TTSR and TTMR as well as the most common temporal database models discussed in literature.

To validate the proposed data model (TTHR), a mapping methodology is used which is considered as an approach for database models validation and verifications [8]. The mapping concept has been used to proof that the proposed model (TTHR) is an extension of CRM [8], [13]. The structure, database, and operations of the two database models are mapped and compared as part of the mapping of the two-model comparisons methodology. This paper introduces, firstly, the comparison method of different database models will be
presented. Next, the properties of CRM will be reviewed. Following that, the properties of TTHR will be presented. Then, TTHR will be proofed to be superset of CRM. Subsequently all the properties of TTTSR and TTMR will be given comprehensively. Finally, TTHR will be proved to have the expression power to be represented by TTTSR and TTMR.

2. Comparison Criteria of Database Models. Comparing a new defined model with the conventional data model in order to prove that the new model is an extension of CRM is one of the most frequent approaches used and discussed by [8], [16]. A few models (including Garani model [8]) have provided a formal proof for this proposition to ensure the expressiveness of these models. Another approach for comparing data models is the mapping between two models [13]. There are three features of the database models, which have to be compared as part of the mapping of the two-model comparisons methodology. These features are structure, database, and operations. Other features like data type (domain), constraints, relational comparison operators ($>$, $\geq$, $<$, $\leq$, $! =$, $=$), and arithmetic function are omitted from the comparing lists of the models characteristics because all of these characteristics apply to all models. Constructive mapping is used in this paper to prove the mapping between TTHR and CRM. Regarding the operations, a snapshot reducibility property has been used to prove that the proposed temporal data model is a consistent extension of the snapshot algebra.

3. The Conventional Relational Model (CRM). CRM is the most common relational data model that was first introduced by Codd in IBM Research in 1979 [1]. The components of this model are described below.

3.1 Structure. CRM or Classical Relation Schema is a relational database that contains a set of attributes, and does not have temporal supports [6]. The structure of the relation is defined as $R = \{A_1, A_2, A_3, \ldots, A_n\}$ where $A_1, A_2, A_3, \ldots, A_n$ are the atomic attributes of $R$ and $n > 0$.

3.2 Database. The data in the schema of CRM is represented as 1NF relation, an example database relation in CRM is shown in Figure-1.

![Figure-1: A database relation in CRM (Employees).](image)

3.3 Operations. The relational algebra operations are the well known and well-defined (union, intersection, difference, projection, selection, rename, Cartesian product, and natural join). The definitions of these operators are not included here because they are standard definitions.

4. TTHR Data Model. The Temporal relational data model proposed in [9], [29] has been described in details in this section, below are some of its components.

4.1 Structure. The schema of relation in TTHR is defined as two relations namely, $R_1$ and $R_2$ such that $R_1 = (A_k, A_u, A_c, \ldots, A_j)$ and $R_2 = (A_i, A_u, A_c, A_j)$. $R_2$ is an auxiliary relation schema. The set of attributes that construct $R_1$ can be classified into key attributes $A_k = \{A_{k1}, A_{k2}, \ldots, A_{ki}\}$ where $i = \text{number of key attributes}$, time-invariant attributes $A_u = \{A_{u1}, A_{u2}, \ldots, A_{un}\}$ where $n = \text{number of time invariants attributes}$, time-varying attributes $A_c = \{A_{c1}, A_{c2}, \ldots, A_{cj}\}$ where $j = \text{number of time varying attributes}$, and temporal attributes $A_t = \{T_{h}, T_{w}\}$. An example, is the schema of Employees relation in Figure-2 with schema structure :-

$Employees = (SSN, Name, Birth_date, Address, Tel_no, Supervssn, D_no, Salary, Rank, T_h, T_w)$

$Employees_VT = (SSN, Att_index, Updated_V, T_h, T_w)$.
4.2 Database. The database in TTHR is represented as shown in Figure-2. The database in TTHR is equivalent to the database in CRM (Figure-1). Moreover, TTHR has extra information in Employees \((T_{ls}, T_{le})\) and the history of updated time-varying attributes in Employees_VT.

5. Mapping TTHR To CRM. The components of TTHR and CRM that have been described in previous sections are going to be mapped, in order to prove that TTHR is a proper consistent extension and a superset of the CRM.

5.1 Structure. Proposition 5.1: the set of schema structure in CRM is a subset of the set of the schema structure in TTHR, or in other words, TTHR is a consistent extension of CRM.

Proof: the set of schema structure in CRM as defined in section 3 is \( R = \{A_1, A_2, A_3, \ldots A_n\} \), where the attributes in CRM can also be categorized into Key attributes, time-invariants, and time-varying attributes. Therefore, \( R = \{A_1, A_2, A_3, \ldots A_n\} \) in CRM could be re-written as \( R = \{A_K, A_U, A_V \} \). The set of schema structure in TTHR as defined in section 4 is \( R = \{A_K, A_U, A_V \} \), and \( R_{VT} = \{A_K, Att\_index, A_T\} \). Removing \( A_T \) form \( R_{VT} \) and discarding \( R_{VT} \) will reduce TTHR to CRM. Based on that TTHR is a superset/consistent extension of CRM.

5.2 Database. Proposition 5.2: the set of database in CRM is a subset of the set of the database in TTHR.

Proof: the database in CRM that represent the current snapshot of the employees data in Figure-1 is a subset of the database in TTHR that are represented by the current valid data as well as the historical changes of the validity of time-varying data. CRM holds one snapshot (the latest state of the database) of the database, whereas TTHR holds all the states of the database in different time points. Therefore, the database in CRM holds one snapshot of different snapshots in TTHR, and then it is a proper subset of database in TTHR.

6. TTSR Data Model. TTSR is one of the approaches of modeling temporal database in relational data model, the components of TTSR are described below.

6.1 Structure. The schema structure of relation \( R \) in TTSR is defined as \( R_{TTSR} = (A_K, A_U, A_V, T_{ls}, T_{le}, T_{ls}, T_{le}) \), the example in Figure-3 shows a temporal database relation Employees represented by TTSR, Employees = (SSN, Name, Birth\_date, Address, Tel\_no, Supervssn, D\_no, Salary, Rank, T_{ls}, T_{le}, T_{ls}, T_{le})

6.2 Database. The database of Employees relation in TTSR as shown in Figure-3 is equivalent to the database in TTHR (Figure-2). The difference is that TTSR holds the current and the historical change of data in one relation, whereas in TTHR, the data are located in two relations as introduced in section 4.
7. Mapping Between TTHR and TTSR. The components of TTHR and TTSR are going to be mapped, in order to prove that TTHR has the expression power in the main temporal data models in literature.

7.1 Structure. Proposition 7.1: The set of schema structure in TTHR can be mapped into the set of the schema structure in TTSR, or in other words, TTHR can be represented in TTSR.

Proof: The set of schema structure in TTSR as defined in section 6 is $R_{TTSR} = (A_x, A_u, A_c, T_u, T_v, T_w)$. The set of schema structure in TTHR as defined in section 4 is $R_x = (A_x, A_u, A_c, A_v)$, where $A_v = (T_u, T_w)$, and $R_{VT} = (A_x, Att\_index, \alpha, A_v)$, where $A_v = (T_u, T_w)$. Adding $T_u, T_w$ to the schema structure $R_x = (A_x, A_u, A_c, T_u, T_v)$ of TTHR, and discarding $R_{VT}$ will yield to the structure of TTSR. Based on that schema structure in TTHR is a fully expressive in TTSR.

Proposition 7.2: The set of schema structure in TTSR can be mapped into the set of the schema structure in TTHR, or in other words, TTSR can be extended to TTHR.

Proof: The set of schema structure $R_{TTSR} = (A_x, A_u, A_c, T_u, T_v, T_w)$ in TTSR can be extended to TTHR by firstly removing the temporal attributes $T_u, T_v$ from $R_{TTSR}$, and secondly creating a new auxiliary structure $R_{VT} = (A_x, Att\_index, \alpha, T_u, T_w)$. Based on that, schema structure in TTSR can be extended to be represented in TTHR.

7.2 Database. Proposition 7.3: The set of database in TTHR representation can be mapped into TTSR representation with zero percent loss of data.

Proof: The following Algorithm 7.1 transfers TTHR relation instance into a corresponding instance in TTSR representation scheme. The convert function in the routine TTHR to TTSR is a converting function that takes as an argument the sets of interval time of validity of all the time-varying attributes of each object, and returns a set of time intervals after removing the repeated and overlapping intervals.

![Figure-3: Temporal database relation in TTSR (Employees)](image)

```
Algorithm 7.1 Map TTHR to TTSR

Input: Temporal relation in TTHR $r_T$, $\forall t \in r_T$

Output: Temporal relation in TTSR, $r_{TTSR}$

Begin

$r_{TTSR} \leftarrow \emptyset$;

$Tem \leftarrow \emptyset$;

For each $m \in r_T$

\{ $m \leftarrow m$;

For each $z \in \forall t \in r_T$ and $x[a_z] = z[a_z]$

$Tem \leftarrow Tem \cup \{z[a_z], \{index\}, z[T_u], z[T_v]\}$;

$Tem \leftarrow Tem \cup \{x[a_z], 100, x[T_u], x[T_v]\}$; // 100 used only for union compatibility

For each $t \in \text{convert}(Tem)$

\}

End
```
The routine starts by scanning each object \( x \) in relation instance \( R_r \) and find from \( R_r \_VT \) the related objects (tuples) \( Z \) to store all the time intervals of time-varying attributes and lifespan time of \( x \) in a temporary relation \( Tem \). Applying this routine on the Employees relation and its auxiliary relation in Figure-2, the \( Tem \) will be as in Figure-4(a). As shown in Figure-4 \( Tem \) relation has repeated and overlapping intervals. Algorithm 7.2 for convert function below will remove the repeated and overlapping intervals from \( Tem \), and put the result in temporary output relation \( Temo \) (Figure-4(b)). At this stage the intervals in \( Temo \) relation covers all the time intervals in \( Tem \), and represents all the time intervals that object \( x \) has changed in its lifespan time or the validity of time-varying attributes.

![Figure-4](image-url)

**Figure-4:** Convert function (a) input relation \( Tem \), (b) output relation \( Temo \)

The next step TTHR to TTSR routine will take each interval \( i_{temo} \) from \( Temo \) and first, copy the attributes form \( x \) object to \( y \) object, then scan each related tuple(s) to \( x \) in \( R_r \_VT \), if there is an overlapping between \( i_{temo} \) and the corresponding interval \( i_{R_r \_VT} \) in \( R_r \_VT \) then \( z \) tuple in \( R_r \_VT \) will be checked by attribute \( Att\_index \), if its value equal to 0 (zero value or sometimes ‘LS’) then the lifespan time of \( y \) will be overwritten with the values in \( z \) tuple, otherwise \( z \) tuple records would change in
time-varying attributes then the corresponding time-varying attributes in \( y \) will be overwritten with the values in \( z \). Finally the values in \( y \) will be appended into \( R_{TTSR} \).

Algorithm 7.2 \textit{Convert} \\
\textbf{Input:} Temi \\
\textbf{Output:} Temo \\
\textbf{Begin} \\
\begin{align*}
Temo \leftarrow \emptyset ; & \quad B' \leftarrow \text{Temi} ; \\
\text{For each } z \in B' & \quad \text{//Remove lifespan intervals and repeated intervals} \\
\{ & \quad \text{If } z[\text{index}] = 0 \text{ then} \\
& \quad \text{//This flag indicates the lifespan intervals} \\
& \quad B' \leftarrow B' - z ; \\
& \quad \text{For each } w \in B' \\
& \quad \text{If } z[T_w] = w[T_w] \text{ and } z[T_w] = w[T_w] \text{ then} \\
& \quad B' \leftarrow B' - w ; \\
\} \\
\text{For each } z \in B' & \quad B \leftarrow \{z[T_w], z[T_w]\} ; \\
& \quad l \leftarrow \text{min}(B[T_w]) ; \quad u \leftarrow \text{min}(B[T_w]) ; \\
& \quad \text{Temo} \leftarrow \text{Temo} \cup \{l, u\} ; \\
& \quad B \leftarrow B - \{l, u\} ; \\
\text{While } \exists i \in B \\
& \quad \text{//Case1 overlap} \\
& \quad l \leftarrow \{T_w\} ; \quad u \leftarrow \{T_w\} ; \quad z \leftarrow \text{last row (Temo)} ; \quad B \leftarrow B - \{l, u\} ; \\
& \quad \text{If } u > z[T_w] \text{ and } l < z[T_w] \text{ then} \\
& \quad \text{equals Null;} \\
& \quad \text{Else If } u = z[T_w] \text{ and } l = z[T_w] \text{ then} \\
& \quad \text{equals Null;} \\
& \quad \text{Else If } l < z[T_w] \text{ and } z[T_w] \leq u \text{ then} \\
& \quad \text{//Case3 overlap} \\
& \quad \text{Temo} \leftarrow \text{Temo} \cup \{z[T_w], z[T_w]\} ; \quad \text{Temo} \leftarrow \text{Temo} \cup \{l[T_w], l[T_w]\} ; \\
& \quad \text{Temo} \leftarrow \text{Temo} \cup \{l + 1, u\} ; \\
& \quad \text{Else If } z[T_w] > l \text{ and } z[T_w] \leq u \text{ then} \\
& \quad \text{//Case4 overlap} \\
& \quad \text{Null;} \\
& \quad \text{Else If } z[T_w] + 1 = l \text{ then} \\
& \quad \text{//Case5 interval comes after the second} \\
& \quad \text{Temo} \leftarrow \text{Temo} \cup \{l, u\} ; \\
& \quad \text{Else If } z[T_w] < l \text{ then} \\
& \quad \text{//Case6 no overlap} \\
& \quad \text{Temo} \leftarrow \text{Temo} \cup \{l, u\} ; \\
\end{align*}
\text{Return } \text{Temo} ; \quad \text{End Convert;} \\
\textbf{End Convert.}

\textbf{Figure-3} depicts the representation of Employees temporal database relation using TTSR that is corresponding to Employees temporal database relation represented in TTHR model. A temporal relation schema Employees (\textbf{Figure-3}) is used to record employees' data together with the validity of the time-varying attributes in Employees as well as the changes of the lifespan of the objects in Employees.

\textbf{Proposition 7.3:} The set of database in TTSR representation can be mapped into TTHR with zero percent loss of data.

\textbf{Proof:} The reverse transformation Algorithm 7.3 TTSR TO TTHR converts between TTSR relation instance and a corresponding instance in TTHR representation. The routine starts by scanning each object \( x \)
in relation instance $R_{TTSR}$, find the latest version of $x$ in $R_{TTSR}$ and insert it into $R_T$.

Algorithm 7.3 Map TTSR to TTHR

**Input:** Temporal relation in TTSR $r_{TTSR}$

**Output:** Temporal relation in TTHR $r_T, _vT_r$

**Begin**

$r_T \leftarrow \emptyset$

$_vT_r \leftarrow \emptyset$

$Tem \leftarrow \emptyset$

$r'_{TTSR} \leftarrow r_{TTSR}$

For each $x \in r_{TTSR}$ // insert the latest current valid data into $r_T$

$r_T \leftarrow r_T \cup \{x[a_1], x[a_2], \max(x[T_u]), x[T_u]\}$ ; // find the latest version of $x$ and insert it into $r_T$

$r'_{TTSR} \leftarrow r'_{TTSR} \setminus \{x[a_1], x[a_2], \max(x[T_u]), x[T_u]\}$ ; // remove the latest version of $x$ from $r'_{TTSR}$

For each $a_{cm} \subseteq r'_{TTSR}$ // insert the historical changes of time-varying attributes into $_vT_r$

For each $x \in r'_{TTSR}$

$Tem \leftarrow Tem \cup \{x[a_1], m, x[a_{cm}], x[T_u], x[T_u]\}$

$_vT_r \leftarrow _vT_r \cup coalesce (Tem)$ ; // coalesce function

$Tem \leftarrow \emptyset$

} For each $x \in r'_{TTSR}$ // insert the historical changes of lifespan into $_vT_T$

$Tem \leftarrow Tem \cup \{x[a_1], 0, LS, x[T_u], x[T_u]\}$

$_vT_r \leftarrow _vT_r \cup coalesce (Tem)$ ; // coalesce function

End Map TTSR to TTHR;

Return $R_T, _vT_T$

Then, for each time-varying attribute in $R_{TTSR}$ a selection of key attributes with the time-varying attribute and the interval of validity will be done and stored in Tem relation, which produces a set of value-equivalent representational tuples. An example is shown in Figure-5 where the selection was on Address time varying attributes. The Algorithm 7.4 for coalesced function below is used to coalesce Tem and store the data in $R_T, _vT_T$ relation.

Figure-5: Coalesce function
Algorithm 7.4 Coalesce Algorithm

**Input**: non-coalesce relation $R$ (Query result)

**Output**: coalesced relation $R_c$. 

**Begin**

```plaintext
R ← $R$ orderby $(a_1, a_2, ..., a_n, T_{vs}, T_{ve})$
$R_c$ ← $\emptyset$
Size ← $\text{Count}(R)$;
$j = 1$;
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 $(T'_{vs} \leq T_{ve} + 1)$
$T_{ve} = \text{max}(T_{ve}, T'_{ve})$;
$j = j + 1$;
END WHILE;
$R_c$ ← $R_c \cup \{a_1, a_2, ..., a_n, T_{vs}, T_{ve}\}$;
$j = j + 1$;
WHILE $j < \text{Size}$;
RETURN $R_c$;
**End Coalesce;**
```

8. TTMR Data Model. TTMR is one of the approaches of modeling temporal database in relational data model, the component of TTMR is described in the next supsection.

8.1 Structure. The schema structure of a temporal database relation $R$ in TTMR is represented by, first, snapshot relation $R_{TTMR} = (A_k, A_v)$, second, for each time-varying attributes there are a separate relations $R_{A_i} = (A_k, A_v, T_{vs}, T_{ve})$ . . . . . . . . . . $R_{A_n} = (A_k, A_v, T_{vs}, T_{ve})$, and finally, $R_{LS} = (A_k, T_{vs}, T_{ve})$ for the lifespan time which are all in 1NF relations. The example in Figure-6 shows a temporal database relation Employees represented by TTMR, the schema structure is as below:

**Employees** = $(SSN, Name, Birth _date)$,
**Emp_Is** = $(SSN, T_{vs}, T_{ve})$,
**Emp_Address** = $(SSN, Address, T_{vs}, T_{ve})$,
**Emp_Tel_no** = $(SSN, Tel_no, T_{vs}, T_{ve})$,
**Emp_supervssn** = $(SSN, Supervssn, T_{vs}, T_{ve})$,
**Emp_Dno** = $(SSN, D_no, T_{vs}, T_{ve})$,
**Emp_salary** = $(SSN, Salary, T_{vs}, T_{ve})$,
**Emp_Rank** = $(SSN, Rank, T_{vs}, T_{ve})$.

8.2 Database. In TTMR, the database are separated over multiple temporal database relation as shown in Figure-6. The database in TTMR is equivalent to the database in TTHR (Figure-2). The difference is that TTHR holds the current and the historical changes of data in two relations, whereas in TTMR the data is located in 8 relations.
9. Mapping Between TTHR and TTMR. The components of TTHR and TTMR are going to be mapped in order to prove that TTHR has the expression power in temporal data models in literature.

9.1 Structure. Proposition 9.1: The set of schema structure in TTHR can be mapped into the set of the schema structure in TTMR, or in other words, TTHR can be represented in TTMR.

Proof: Referring to the set of schema structure in TTMR as defined in section 7, and the set of schema structure in TTHR as defined in section 4. Removing \( A_c, A_t \) from \( T_R \), creating for each time-varying attributes \( C A \) separate relations \( R_{C i} = (A_c, A_t, T_m) \) for the lifespan time, and discarding \( R_{VT} \) will yield to the temporal representation structure of TTMR. Based on that schema structure in TTHR is a fully expressive in TTMR.

Proposition 9.2: the set of schema structure in TTMR can be mapped into the set of the schema structure in TTHR, or in other words, TTMR can be extended to TTHR.

Proof: the set of schema structure in TTMR can be extended to TTHR by firstly removing all time-varying attributes relations and add them as an attributes to main relation together with the temporal attributes \( ls, le \), and secondly creating a new auxiliary structure \( R_{VT} = (A_c, att_index, a, T_m) \). Based on that, schema structure in TTMR can be extended to be represented in TTHR.

9.2 Database. Proposition 9.3: The set of database in TTHR representation can be mapped into TTMR with zero percent loss of data.

Proof: The following Algorithm 6.5 TTHR _ TO _ TTMR transfers between TTHR relation instance and a corresponding instance in TTMR representation. The routine starts by scanning each object \( x \) in relation instance \( R_x \) and takes only the key attributes, non-time-varying attributes, and lifespan time attributes to be inserted into \( R_{TTHR} \), then for each time-varying attributes \( \{C_1, ..., C_c \} \) another scan is going on \( R_{VT} \) to find the corresponding updated time-varying attributes (\( m \)) and insert the tuples into the corresponding relation \( R_{VT} \). Finally, the historical changes of the lifespan of the database objects are recorded in \( R_s \) relation. TTMR approach is almost the same as our proposed model (TTHR) except in our model, we merged the historical changes of time-varying attributes in one relation \( R_{VT} \) and distinguish
them using $Att\_index$ attribute. One more difference is keeping the current valid data in one relation getting rid of temporal intersection join.

Algorithm 9.1 Map TTHR to TTMR.

Input: Temporal relation in TTHR $r_T\_VT\_r_T$

Output: Temporal relation TTMR, $r_{TTMR}\_r_1\_\ldots\_r_{a_i}\_r_{ls}$

Begin

\[
\begin{align*}
gr_{TTMR} & \leftarrow \emptyset ; \\
\_r_1 \ldots \_r_{a_i} & \leftarrow \emptyset ; \\
\_r_{ls} & \leftarrow \emptyset ; \\
\text{For each } x \in r_T
\end{align*}
\]

\[
\begin{align*}
\{ & \{r_{TTMR} \leftarrow \{x[a_1], x[a_2], x[T_1], x[T_n]\}; \\
\text{For each } a_{cm} \subset r_T \quad \text{// from m=1 to i, total number of time-varying attributes} \\
\{ & \{y \in \_r_T \quad \text{and } y[index] = m \\
\text{For each } a_{cm} \quad \text{// last row } y \in \_r_T
\end{align*}
\]

\[
\begin{align*}
\text{If } x[a_1] = y[a_1] \quad \text{and } y[T_{ve}] > x[T_{ls}] & \quad \text{then} \\
r_{a_{cm}} & \leftarrow r_{a_{cm}} \cup \{x[a_1], x[a_{cm}], y[T_{ve}]+1, x[T_{ls}]\}; \\
\text{Else} & \\
r_{a_{cm}} & \leftarrow r_{a_{cm}} \cup \{x[a_1], x[a_{cm}], x[T_{ls}], x[T_n]\}; \\
\text{End if;}
\}
\]

\[
\begin{align*}
\text{For each } y \in \_r_T \quad \text{and } y[index] = 0 \quad \text{// lifespan} \\
\_r_0 & \leftarrow _r_r \cup \{y[a_1], y[T_{ns}], y[T_{ve}]\}; \\
\}
\}
\]

End Map TTHR to TTMR:

Proposition 9.2: The set of database in TTMR representation can be mapped into TTHR with zero percent loss of data.

Proof: The reverse transformation Algorithm 9.2 converts between TTMR relation instance and a corresponding instance in TTHR representation scheme.

Algorithm 9.2 Map TTMR to TTHR.

Input: Temporal relation in TTMR $r_{TTMR}\_r_1\_\ldots\_r_{a_i}\_r_{ls}$

Output: Temporal relation TTHR $r_T\_VT\_r_T$

Begin

\[
\begin{align*}
gr_T & \leftarrow \emptyset ; \\
\_r_T & \leftarrow \emptyset ; \\
\text{For each } x \in r_{TTMR}
\end{align*}
\]

\[
\begin{align*}
\{ & \{m = 1 \text{ to } i \\
\text{For each } y_m \in r_{a_{cm}} \quad \text{and } y_m[a_k] = x[a_k] \\
\text{If } y_m[T_{ve}] \text{ is the max value then} \\
r_T & \leftarrow r_T \cup \{x[a_1], x[a_2], y_m[a_{cm}] \ldots y_m[a_m], x[T_{ls}], x[T_n]\}; \\
\text{Else}
\}
\}
\]

24
Conclusion. In this paper we have demonstrated that TTHR is a consistent extension and a super set of CRM. A mapping methodology has also shown that TTHR has the expressive power to be represented in the main temporal data models in literature. Mapping of two data models is the method that has been adopted for this proof. This method is presented in [7, 16]. The structures, the databases, and the operators, are the features of the two models that have been compared. The comparisons have formally proved that TTHR is a consistent extension of CRM and has the expressive power to be represented in the reference temporal data model BCDM and the main temporal data models in literature (TTSR and TTMR).

REFERENCES