A Novel Study of Attribute time stamping models in Temporal Database

Shailender Kumar
Department of Computer Science & Engineering
Ambedkar Institute of Advanced Communication
Technologies and Research
New Delhi, India

Aman Jolly
Department of Computer Science & Engineering
Ambedkar Institute of Advanced Communication
Technologies and Research
New Delhi, India

Abstract -- Time based data is efficiently managed by temporal database models whereas the conventional database models are only capable of storing the present value, temporal database models not only stores historic values but also provide a roadmap to process future valid data. This paper briefly introduces some vital temporal database concepts. After that, various attribute time stamping models are examined in this domain. Finally, a comparative analysis of the various attribute timestamp models is prepared to discuss their strengths and weaknesses of their Implementations and provide useful and optimised results for solving the problems in the field of the temporal database model.

Keywords-- Temporal database, temporal data models, Attribute time stamping models, transaction time, valid time

I. INTRODUCTION

A temporal database is the extension of a relational database that is capable of storing data relating to time instances. It has temporal data types that stores data of the past, present and future time, for example, maintaining the historical record of fashion trends that gradually changed from 1980’s to 2017 or keeping the track of student's preference of stream in choosing a career path that got changed over the period of time. Thus, the collection of time-related data can be stored, managed and analysed by the temporal database. For example, maintaining the historical record of fashion trends that gradually changed from 1980’s to 2017 or keeping the track of student’s preference of stream in choosing a career path that got change over the period of time.[3] Thus, the temporal database stores a collection of time-related data. This type of time-based database uses a certain timeframe (such as microseconds or milliseconds or seconds) and then stores only the variations in the measured data. [12] The timestamp is a discretely stored value for each measurement in relational database management system which makes it very inefficient. This time travelling feature of a temporal database makes it possible to record information about fixing errors in the database. It is essential for a true audit photograph of the data which modifies when it records and allows queries related to how to modify the data over time.[13] There is a large application domain that works with temporary data in computer applications for example history of file backups, Financial Applications such as history of stock markets & share prices, Reservation Systemsto know when was a flight booked, Medical Systems for keeping a record of file backups, Financial Applications such as history of stock markets & share prices, Reservation Systems, sporting events, publications and journals, Weather analyzing Systems, for analyzing weather records for data mining. Thus, the collection of time-related data can be stored, managed and analysed by the temporal database. For example, maintaining the historical record of fashion trends that gradually changed from 1980’s to 2017 or keeping the track of student’s preference of stream in choosing a career path that got change over the period of time.[3] Thus, the temporal database stores a collection of time-related data. This type of time-based database uses a certain timeframe (such as microseconds or milliseconds or seconds) and then stores only the variations in the measured data. [12] The timestamp is a discretely stored value for each measurement in relational database management system which makes it very inefficient. This time travelling feature of a temporal database makes it possible to record information about fixing errors in the database. It is essential for a true audit photograph of the data which modifies when it records and allows queries related to how to modify the data over time.[13] There is a large application domain that works with temporary data in computer applications for example history of file backups, Financial Applications such as history of stock markets & share prices, Reservation Systems, sporting events, publications and journals, Weather analyzing Systems, for analyzing weather records for data mining.

a) Categorization of Temporal Databases

Based on Time dimension associated with the temporal database, temporal relations were implemented by attaching each tuple with a time interval, a start and end time value indicating when the data is valid. Information on how to form temporal data and relations is given in the background section.

Temporal Aspects includes:
- Valid Time
- Transaction Time

A table that implements one of these is temporal, two is bi-temporal.

1) Valid Time

[1] The time during which a fact is true with respect to the real world, it is modelled as a range between two timestamps closed at the lower bound can be open at upper bound.

Valid Time Example [12]

<table>
<thead>
<tr>
<th>Owner</th>
<th>Property</th>
<th>StartVT</th>
<th>EndVT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aman</td>
<td>‘Some place’</td>
<td>2012-09-28</td>
<td>2016-02-20</td>
</tr>
<tr>
<td>Akshay</td>
<td>‘Some place’</td>
<td>2014-02-21</td>
<td>∞</td>
</tr>
</tbody>
</table>

2) Transaction Time

[1] The time period during which a fact stored in the database is considered to be true modelled as a range between two timestamps closed at lower bound can be open at upper bound.

<table>
<thead>
<tr>
<th>Owner</th>
<th>Property</th>
<th>StartVT</th>
<th>EndVT</th>
<th>Start</th>
<th>EndTT</th>
<th>EndT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aman</td>
<td>‘Some place’</td>
<td>2012-09-28</td>
<td>2016-02-20</td>
<td>2016-09-28</td>
<td>2016-09-09</td>
<td></td>
</tr>
<tr>
<td>Akshay</td>
<td>‘Some place’</td>
<td>2014-02-21</td>
<td>∞</td>
<td>2016-09-09</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>

However, a Bitemporal Conceptual data model contains both valid and transaction timestamps.

Based on Time stamping technique supported by temporal databases
1) Attribute time stamping

In the case of attribute time stamping, N1NF relations are used, and time stamps are attached to attribute values. Entire data about an object being modelled can be stored in one tuple of the N1NF relation-the entire history of an object is stored in a tuple. Thus, the data in a tuple which are not affected by any updates are not to be repeated, all time-specific attributes can be drawn in a relation. Time is linked to the attribute values of a relation and the history of an attribute is included in a set of triplet valued, as shown in table below. The triplet of the form <[l, u), v> means “l” represents lower time bound, “v” represents upper time bound and “v” represent the value of the attribute, this approach violates 1NF since it does not contain asingle or indivisible value.

VV: Valid Time, TT: Transaction time

Here we are considering a temporal database of teachers in a college for example consider an example of EMP#E2 in table I. Jagriti Verma in between the time <[T12,T27]TT,[T15,T27]VT, JagritiVerma(value)> lives at Pashchimvihar<([T12,T27]TT,[T15,T27]VT, Pashchimvihar(value)>:when Jagriti got married, her surname and residential address got changed <([T28,T40]TT,[T28,T40]VT, Jagriti jolly(value)> lives at Adarshnagar<([T28,T40]TT,[T28,T40]VT, Adarshnagar(value)>).Later she had a divorce so her surname and residential address in valid time [T41,T45] got changed again and restored with the values <[T41,T45],[T41,T45],Jagritiverma> and residential address <[T41,T45],[T41,T45],Pashchimvihar> in valid time [T53,Now] Jagriti get re married, changing her surname <[T53,Now],[T55,Now],Jagritiyadav> and address <[T53,Now],[T55,Now],Noida>, this value get stored in the bitemporal table with a transaction time [T55,Now].All those values are stored in single tuple and all time varying attribute like address and name is modelled in one relation.

2) Tuple time stamping

There are two models in the Tuple time-stamping approach and use all the benefits of traditional relational databases. [2] The Tuple Timestamp Single Relations (TTSR), which keeps its all-time-related features with non-temporal features with different properties, is therefore presented as two additional time attributes called "from" and "to" fields, when the approach is not efficient since if a relation has many attributes, a whole new tuple version is created whenever any one of the attributes is updated.

If the attributes are updated asynchronously, then each new version can have only one attribute difference, thus repeating the value of the other attributes which are same requires a huge space to accommodate and another is called Tuple Timestamp Multiple the relations (TTSMR) where time is as decomposed into time varying attribute and distributed over several relations and non-temporal attributes are gathered into separate relation. Also, a separate relation is required for each time dependent attribute as well. A relation schema is augmented with two (four) time attributes to designate the time reference of its tuples. However, each time varying attribute should be in a separate relation, which results in many small relations. Here we are using Tuple Timestamp Multiple Relation (TTMR) because [1] TTMR is cost effective and surpass Tuple Timestamp Single Relation (TTSR) in terms of themagnitude of performance with regards to execution time and used space memory.
The example which was mentioned above for Table 1 (Attribute time-stamping) same have been used in tuple time stamping here the table has been bifurcated into many relations. Because the 1NF form is used in tuple time stamping. Separate tuples are needed to store any change in the object which increases the tuple size. Here in Table 3 E2 of EMP attribute is replicated four times just to show the change in name. In Table 4 also E3 of EMP attributes replicated four times just to show the change in address. Same thing happened when E3 address got changed, the tuple of table 4 with EMP attribute E3 multiplied 2 times just to show the change in address whereas non-time varying attributes like birth date and department are grouped into separate Table 2. This methodology increases the relations between the tables. In attribute time stamping a single relationship is used whereas in tuple time stamping relationship withintable increases as the time varying attributes increases.

A brief comparison shows that less space is used in attribute time-stamping as compared to tuple time-stamping due to data redundancy whereas attribute time stamping requires a bit more time than tuple time-stamping because nested Bitemporal Relational database creates a separate table. Attribute timestamping supports both homogeneous and as well as temporally heterogeneous data whereas tuple timestamping supports only homogeneous data. In attribute time stamping 1NF rule is violated as non-temporal attributes are allowed in a single relation with temporal attributes which allows nesting and avoids redundancy unlike tuple time-stamping.

II. Literature Review

In this section, we have presented a literature review of Temporal Databases in various domains.

CananErerAtay[2] have presented how attribute and tuple time stamped bitemporal databases can be implemented in an object-relational database and tested to see which one performs better in terms of time and space. The Author has done a comparative analysis using Oracle 9i’s AUTOTRACE utility and found that an attribute time-stamping approach might require relatively a bit more time because the nested bitemporal relational database creates a separate table for each time-related attributes, however, attribute time stamping requires less space and supports temporally heterogeneous as well as homogeneous data.

Michal Kvet, Karol Matiaško [4] have shown that how brain tumour can be monitored and processed by data obtained from magnetic resonance imaging management techniques and storing the MRI results in temporal database models. He chose two models one is column level temporal system and other is epsilon column level temporal system by commercial tools. They concluded that there is still very marginal support to changes in the schema by commercial tools. They compared two commercial tools, one is SQL Compare and the other is Oracle Change Management Pack. SQL Compare compares and synchronises SQL Server database schema, and can be used when changes made to the schema of a local database need to be pushed to a central database on a remote server. Oracle Change Management Pack is aimed to report and track the evolving state of metadata, thus allowing to compare database schema, and to generate and execute scripts to carry out the changes. In both cases, formulating a single query spanning multiple databases with different schema is not possible so there is a need to deliver effective solutions by both researchers and vendors.

Ricardo Campos, Gae² L Dias, Al’Ipio M. Jorge, Adam Jatow[6] overviewed the important advances in Temporal information retrieval as they explained crucial concepts related to the notion of time, calendar systems, handling temporal expressions in texts, and the different types of sources of temporal information on the web. They also surveyed existing research that deals with the temporal aspects of both search queries and documents and the diverse ways of generating temporally enhanced search results.

Michal Kvet, Karol Matiaško, and Marek Kvet[7] proposed a Column level temporal system. It is an efficient approach to process data with different granularity and keeps the duplicity to minimal. Minimising the duplicities is one of the important factors which improves processing speed. To get a current snapshot and all data during the lifetime of the object in the proposed system.

Michal Kvet, Karol Matiaško and Marek Kvet[8] extended the concept of Column Level Temporal System (CLTS) and proposed two models- Extended Column Level Temporal System and Epsilon Column Level Temporal System. Extended Column Level Temporal System keeps the principal of CLTS intact but changes the history management and management of temporal table. It uses two additional attributes in the temporal table - Data type, Operation and also introduces Restore operation which provides validity restoration after transaction failure or data distribution failure. Data Manipulation Operations such as Insert, Update are manipulated using triggers and procedures.

III. Attribute time stamping models

The temporal model aims at managing static information as well as time varying information in a singular approach. There are lots of existing conventional database models like relational, entity relationship models. However, there is a robust need for temporal data models as a result of using the conventional databases, managing the time-varying data is left at the developer end or other ad hoc methods that must be reimplemented or reworked upon whenever a new application or software is developed. Therefore usage of conventional database models results in an inefficient approach to manage the temporal data and so there is the
requirement of a standard framework for all applications provided by temporal information models. A huge quantity of work has been done to store and access the temporal data. All the work that has been done revolves around two approaches. Initial approach has been to change the underlying conventional database model by introducing special attribute, temporal types etc. to map the temporal feature of data. Many information models are projected using this approach. Out of those conventional models, using the relational model to increase and propose temporal model is that the most popular choice because it continues to be widely employed in commercial applications. As mentioned, to represent the temporal data using this approach needs representation of temporal aspect within the kind of attributes in relations, in addition to the static data attributes. Next would be the brief introduction relating to completely different Attribute data models and comparative analysis among them.

A. Column level temporal system

[7]The column level temporal system is coined by Michal Kvet, Karol Matiasko and Marek Kvet in 2004. Because of the duplicate values in uni-temporal and bitemporal systems, the difference between the processing in currently valid data and historical data is important. Thus, it's an economical approach to process information with totally different granularity and keeps the duplicity to lowest. Minimising the duplicities is one of the necessary factors that improve processing speed to get current snapshot and all information throughout the life cycle of the object within the projected system. They have planned to manage the attributes and not the complete object for the modification. Every temporal attribute is maintained in a separate table with history being managed by the temporal table. The static attributes are kept in one table with no duplicity. The values are tracked by the ID of the table, the ID of the column being updated and table ID, row ID, and column ID wherever the modified value is inserted. Performance experiments are carried out on Oracle 11g database with a total number of hundred thousand records in the main structure. Though providing spectacular results on performance, the system isn't able to get future valid values before executing the job, an automatically scheduled functionality to be executed after the scheduled time.

[7]In the column attribute level, the entire state is formed by the grouping the properties and states of the attributes. The most advantage is that the chance of data processing in time with different granularity - sensory data.

The temporal table consists of the next mentioned attributes [8] see also figure 1:

- **ID change**: it’s a primary key of temporal table
- **ID previous change**: projects the last change of an object identified by ID. This attribute can also have a NULL value that means, so the data has been inserted for the first time in past and are still actual and the data have not been updated yet.
- **ID tab**: projects the table whose record has been managed by DML statement.
- **ID_orig**: carries the information about the identifier of the row that has been changed.
- **ID_column, ID_row**: hold the referential information to the old value of an attribute (if the DML statement was UPDATE). Only update statement of temporal column sets not null value.
- **BD**: the beginning date of the new state validity of an object.

![Figure 1. Column Level Temporal System Flowchart](image-url)
B. Extended Column level temporal system

The extended column level temporal system relies on object attribute (one sensor) [8] because the main part of the granularity and can be considered because of the improvement of the column level characteristics within the term of performance and model management simplicity. The differences are based on the explicit definition of the type of the operation and opportunities to manage also transaction access. The principles of column level approach are delineated in. Moreover, new improved system definition permits existing applications to be connected to the current system without any Amendment of the program code. Only database characteristics are enlarged.

The core of the system is that the table, that manages any modification on temporal attributes. The temporal attribute represents the state (characteristics) that ought to be monitored over the time – sensing element. If the value is modified, data regarding the update is kept within the developed temporal table and historical value is inserted into to the table containing historical values. Conversely, conventional attribute represents data, that don't modify their values over the time or the system needs solely actual state of those elements – e.g. descriptive knowledge. The application is directly connected to the main tables with current valid values. It implies that presently used applications are used without any modification. Historical values are kept within the special section, every temporal data type has one table outlined by the identifier (primary key) got using the sequence and trigger and also the values themselves. Thus, the principle and system are comparable to the column level temporal system, however historical values management and the temporal table is different.

C. Epsilon Column level temporal system

Epsilon Column Level Temporal System is an additional improvement of Extended CL TSI[4]. It records only important knowledge that is defined as data wherever the difference between the new and previous value is larger than or equal to epsilon value (constant). This additional improves the performance of assorted operations like Insert and select due to reduction in the variety of records. These models offer transaction access by using four transaction parameters. These four parameters outline the rules and conditions within which transaction is applied. These models are enforced for performance experiments using Oracle I1g database with ten thousand magnetic resonance imaging records where every record contains ten magnetic resonance imaging results.

Widening of the temporal system definition provides epsilon (_) principle that is predicated on dependability and important change. Every sensing element has defined quality – Measurement precision. Therefore, it's not necessary to store all modified values, only those, that indeed represent important modification. The relevant data modification is characterised by the minus of the actual and new value got from the sensing element, that ought to be on top of outlined parameter (_) - threshold.

Figure 3. Process In Epsilon Column Level Temporal System

IV. Comparative Analysis of various data Models

<table>
<thead>
<tr>
<th>Name of Model</th>
<th>Column Level Temporal System</th>
<th>Extended Column Level Temporal System</th>
<th>Epsilon Column Level Temporal System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Management Techniques</td>
<td>1. Temporal table for handling historic data</td>
<td>1. same as Column Level, same conventional main tables-no change in current application</td>
<td>1. Extended temporal table</td>
</tr>
<tr>
<td></td>
<td>2. Each attribute of main base table has a distinct table</td>
<td>2. Added two extra attributes in temporal table Data type Operation Also Introduced Restore operation which provides validity restoration after transaction</td>
<td>2. Captures only significant changes-change greater than epsilon value</td>
</tr>
<tr>
<td></td>
<td>3. Transaction handling</td>
<td></td>
<td>3. Conventional main tables-no change in current application</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Transaction management by Transaction Access Rules</td>
</tr>
</tbody>
</table>

Figure 2. Extended Column Level Temporal System Architecture
### V. CONCLUSION AND FUTURE SCOPE

The goal of this survey is to define the concept of the temporal database in a nutshell and compare some vital Attribute time-stamping temporal models, we have assessed few attribute time-stamping models in Temporal Database in this paper. There is no doubt that the Column Level Temporal System/Extended Column Level Temporal System of attribute time stamping model is useful whereonly the attribute is changed but not the entire object, unlike Object level temporal system where data redundancy is the key issue. Also Epsilon Column Level Temporal System would be considered as an excellent choice where only the significant data in the historic tables are recorded, Epsilon Column Level System have a capability to save the cost of data and processing time as it selectively choose data to be inserted in the tables instead of recording each and every change which significantly reduces the size of the structure.

We believe that our effort will be helpful for the researchers in designing new data models and rectifying the shortcomings of the present system.

### VI. REFERENCES


