**International Journal of Engineering Sciences Paradigms and Researches (IJESPR)**
**(Vol. 43, Issue 01) and (Publishing Month: June 2017)**
**An Indexed and Referred Journal with Impact Factor: 2.80**
**ISSN (Online): 2319-6564**
**www.ijesonline.com**

# Management of Temporal Data

**Shailender Kumar**

**Computer Science & Engineering**
**Ambedkar Institute of Advanced Communication Technologies and Research**
**Geeta Colony Delhi, India**
***shailenderkumar@aiactr.ac.in***

**Publishing Date: June 24, 2017**

**Abstract**

In this paper, tuple time-stamping temporal model is proposed for the retrieval of meteorological data. This paper accentuate upon the retrieval of temporal database by extending the functionality of Postgres database system. The tuple time-stamping approach is used to query the time-varying data. Conventional databases are not capable of managing temporal data as it is a series of snapshot relations. We proposed and implemented a temporal model for the retrieval of temporal data using the meteorological dataset of a particular location in the National Capital Region of Delhi for the year 2014. The results may be used for the analysis of climate trends. Various temporal models have been proposed and implemented by the researchers but fewer efforts have been put in the real world problems.

***Keywords:*** *Indexing, Time-stamping, Temporal Relational Algebra, Valid Time.*

## 1. Introduction

Time is an important attribute of the real world. Consequently, a lot of efforts have been put by the researchers to develop dedicated approaches in the field of temporal relational databases. Most of the research in this area [1, 5] is focused on the implementation of the time varying features in the database and less work has been done in applying these temporal concepts in the real world problems [7]. This study targeted to apply temporal concepts to the meteorological data for the efficient retrieval [9] of various weather parameters.

Temporal databases support different aspects of time except the user-defined time. Time-varying data is represented by appending timestamp to the values [2]. The various timestamps used may be time points, intervals or set of time intervals. The three different kind time-stamps in temporal databases are transaction time, valid time and bitemporal. The Valid time is the time when the data is valid in the real world. The Transaction time is the time when the data is recorded in the database. Whereas, when the union of valid and transaction time is used in the temporal database, it is termed as called bitemporal database. The temporal databases use two different approaches to attach the timestamp to the data. The first one is the tuple time-stamping that uses 1NF relations and the other is attributing time-stamping where the domains of the attributes are non-atomic in nature.

This paper proposes a temporal model for the analysis of climate trends using the meteorological dataset of a particular location in the National Capital Region of Delhi for the year 2014. The implementation of the proposed model has been done using Java NetBeans for the development of the application and Postgres as the underlying database. The model assisted in the efficient retrieval of temporal climate data.

### 1.1. Literature Survey

There has been an extensive escalation of the research interest in the field of temporal databases that is visible in the recent years. As a result of which, numerous temporal data models [2, 6, 7, 10] and their query languages have been proposed and implemented. Few earlier works presented the survey of the temporal data models. The design of the temporal data models mainly focuses on the semantics, storage, query processing and finally the implementation.

The time dimension is the most significant aspect in the designing of temporal databases. The three different variants of time used in temporal databases are valid time, transaction time and bitemporal. Valid time has been the most frequently

used dimension in the majority of the research works. The temporal element is attached to the database using the time-stamping approach. The two different techniques of time-stamping are tuple time-stamping and attribute time-stamping. In this paper, we have used tuple time-stamping technique with valid time as the dimension.

The rest of this paper is arranged as follows. Section 2 of the paper discusses the key concepts involved in the logical design of the temporal database model. In Section 3, we provide a temporal data model for the retrieval of the meteorological data. Section 4 briefly introduces the PostgreSQL Section 5. Finally in Section 6, we draw some conclusions and point to directions for future study.

## 2. Methodology

In this section, the stage has been set up for presenting the background of our designing approach.

### 2.1. Temporal Data Model

A temporal data model is defined as the collection of conceptual tools for describing schema and its constraints. In a temporal data model, we have time points that are valid times associated with each tuple and are monotonically increasing with which a temporal relation schema $R_t$ (Starting Time, Ending Time, Attributes) is associated. Furthermore, some constraints are also needed to be applied on the attributes such as Starting Time must be less than Ending Time and attributes must not be NULL. Mathematically, temporal relational schema can be represented as:

$$R_t = (A_1, A_2, A_3 \dots \dots . A_n | T_v)$$

Where $R_t$ is a temporal relation schema having finite set of attributes names $\{A_1, A_2, A_3 \dots \dots . A_n\}$ and $T_v$ represents the set of valid times. $D_i$ is the domain of the attributes where $1 \leq i \geq n$. A temporal relation r on schema $R_t$ expressed as $r(R_t)$ is a set of mappings $m_1, m_2, m_3 \dots \dots . m_k$ from $R_t$ to $D$.

### 2.2. Temporal Relational Algebra

A temporal relational operator is an operator that generates temporal relation whenever applied on the temporal relations[11]. The definitions of the important temporal operators are as under:

**Definition 1 (Temporal Selection)**: Temporal Selection operator is used to retrieve the facts associated with time attributes which are defined in the selection predicate P.

$$\sigma_P^t(r) = \{t \mid t \epsilon r \ \Lambda \ P(t[A])\}$$

**Definition 2 (Temporal Union)**: Temporal union is defined as

$$r_1 \cup_t r_2 = \{t \mid (\exists x \in r_1 \ \exists y \in r_2 \ (x[A] = y[A] \ \Lambda \ t[A] = x[A] \ \Lambda \ t[T] = x[T] \cup y[T])) \ \mathrm{V} \ (\exists x \in r_1 (t[A] = x[A] \Lambda \ (\sim \exists y \in r_2 (y[A] = x[A])) \ \Lambda \ t[T] = x[T] )) \ \mathrm{V} \ (\exists y \in r_2 (t[A] = y[A] \Lambda \ (\sim \exists x \in r_1 (x[A] = y[A])) \ \Lambda \ t[T] = y[T] ))\}$$

The above expression contains three different clauses to generate the output. The first clause will generate all the common tuples from both relations $r_1$ *and* $r_2$. The second clause will generate tuples present in relation $r_1$ only. Similarly, the third clause will generate tuples that are present only in relation $r_2$. Tuples of $r_1$ *and* $r_2$ agreeing on a key attribute are collapsed into one single tuple. For $r_1 \cup_t r_2$ to be valid, the domains of the attribute should be compatible and both $r_1$, $r_2$ have the same arity.

**Definition 3 (Temporal Join)**: Temporal join is defined as

$$r_1 \bowtie_t r_2 = \{t \mid \exists x \in r_1 \ \exists y \in r_2 \ (x[A] = y[A] \ \Lambda \ x[T] \cap y[T] \neq 0 \ \Lambda \ t[A] = x[A] \Lambda \ t[B] = x[B] \ \Lambda \ t[C] = y[C] \ \Lambda \ t[T] = x[T] \cap y[T])\}$$

In the above expression, it is clear that the timestamp of a particular tuple in the result is the intersection of the timestamps of the two tuple which produced it.

**Definition 4 (Temporal Difference)**: Temporal difference is defined as

$$r_1 -_t r_2 = \{t \mid \exists x \in r_1 \ ((t[A] = x[A]) \Lambda \ ((\exists y \in r_2 \ (t[A] = y[A] \ \Lambda \ t[T] = x[T] - y[T])) \ V \ (\sim \exists y \in r_2 \ (t[A] = y[A] \ \Lambda \ t[T] = x[T])))\}$$

Above expression contains three different clauses to generate the output. The first clause will generate all the tuples from relation $r_1$ whereas the second and third clause will check that none of the tuples from relation $r_2$ appear in the final result. The domains of both the temporal relations $r_1$ *and* $r_2$ are compatible and have same arity.

**Definition 5(Temporal Intersection)**: Temporal intersection is defined using temporal relation difference as

$r_1 \cap_t r_2 = r_1 -_t (r_1 -_t r_2)$

For $r_1 \cap_t r_2$ to be valid, the domains of the attributes should be compatible and both $r_1$, $r_2$ have the same arity.

### 2.3. PostgreSQL

PostgreSQL/Postgres is a leading open source object-relational database system [12] which is developed by teams of volunteers spread over worldwide. It has gone through 15 years of active development that has made it a strong and reliable database. Postgres runs on all major operating systems such as Windows, Mac, Linux, Solaris etc. The long and still growing feature list includes capabilities like extensive backend support, high speed performance, robust security, extensive networking support and user friendly GUI based interaction which contributes to its ever growing popularity. Moreover, this database also has many progressive functionalities or expanded functionalities for academic research and also for basic reliability and stability. A very active community support, extensive research and analytics capabilities that it provides and its capabilities extensively tested by industry veterans makes it an obvious choice for the future.

In comparison to various leading proprietary vendors postgres stands with its advantages and capabilities far apart. It offers flexibility in concept research and allows trial deployment and with a very simple and clear cut licensing terms removes most of the legal headaches. It can be used with a very low cost associated as compared to most proprietary DBs and it provides much more support in efficient manner. The easiness to tailor and extend the code according to personal requirements without any associated cost and the availability across various platforms proves its robustness, stability and extensibility. The Multi version concurrency control and easiness in handling high volumes of data added with functionalities like Nested transactions, Patch access, Temporal support via time dependent data types, powerful set of operations and extensive data types gives it the extra edge over other popular DB vendors.

Moving on to the postgres nuances it carefully addresses the minutes details by giving functionalities like importing foreign schemas, additional row level security which prevents un-authorized access to rows. Block Range Indexing (BRIN) capability, which makes the processing of very large tables fairly easy without need to explicitly declare partitioning, has only been implemented by postgres alone.

Major database vendors like Oracle, PostgreSQL, Teradata and IBM provide temporal support in one way or other but Postgres being open source and a powerful ORDBMS supports rich temporal data types compared to others. It handles time zones intelligently with a support for ISO 8601 standard. Apart from time zone flexibility, Postgres has an interval data and time type. This interval date and time types like tsrange, tstzrange and daterange has a capacity of storing an interval of up to 178 million years with 14 digits precision and measures time at different precisions.

These PostgreSQL data types are considered to be the best temporal data types which can store information related to past, present and future time. In an object-oriented data model, the objects are time stamped to incorporate the temporal functionality. This is due to all these functionalities of the Postgres that enable its use as the underlying database in this paper.

## 3. Proposed Model

Our proposed model is based on tuple time-stamping approach with valid time as the time dimension. In this approach, a separate temporal relation is required for each time dependent attribute if the attributes are not changing concurrently. Otherwise, if the attributes are varying simultaneously, the original temporal relation will not be decomposed. Temporal data model can be defined as the collection of conceptual tools for describing schema and constraints. Integrity constraints are the conditions imposed on the temporal data to ensure its accuracy and efficiency.

### 3.1. Architecture of the Model

Figure 1 illustrates the architecture of the proposed temporal model. The temporal users are able to retrieve the temporal data with the help of the application program. The details of the operations on the temporal attributes are not visible to the user.

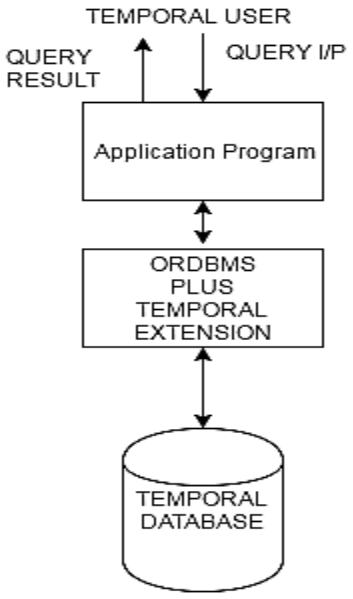



**Figure 1: Architecture of the Model**

The temporal extension of the Postgres composes of two main components. The first one is the temporal data type used as the domain of the temporal attributes and the other is the set of operators used to implement the temporal functionality. The PostgreSql version 9.5 built-in range data type "tsrange" is being used to record the time-stamp. It is used to record a pair of disjoint times such that the tuple is valid during the period.

## 4. Experiments and Results

### 4.1. Software and Hardware used

All the experiments are carried out on a system with 8GB of RAM and Intel(R)Core(TM) i5-3210M CPU@2.5GHz running on Windows operating system. PostgreSQL version 9.5 is being used the underlying database and Java NetBeans IDE version 8.0.2 is used to design the application program.

### 4.2. Data Set

The meteorological dataset of a particular location in the National Capital Region of Delhi for the year 2014 is used for the experimental results of the proposed model.

The original dataset received from the MET department has been preprocessed. The original temporal relation been decomposed to enable the eradication of redundancy that occurred due to variable time intervals in the original temporal relation. There are two 1NF temporal relations comprising of 8760 and 2920 tuples respectively. The time range data type "tsrange" is used to time-stamp the tuples of the temporal relation.

### 4.3. Experiment Results

This section summarizes the result of the five different point and range temporal queries [6] applied on the temporal database. The interface of the application is designed using Java NetBeans.

**Example 1: Show all the weather records till 12 noon of 1st January:**

**Query:**

```
SELECT we.time_range, we.temp, we.humidity, we.visiblity, ew.cloud_amount, ew.wind_speed,
ew.wind_dir, ew.rainfall FROM (SELECT * FROM wcperhour WHERE wcperhour.time_range && TSRANGE('2014-01-01 00:00:00', '2014-01-01 12:00:00')) we
INNER JOIN
(SELECT * FROM wcper3hour WHERE wcper3hour.time_range && TSRANGE('2014-01-01 00:00:00', '2014-01-01 12:00:00'))ew ON we.time_range && ew.time_range ORDER BY we.time_range;
```

**Output:** The result of the query is shown in Figure 2.

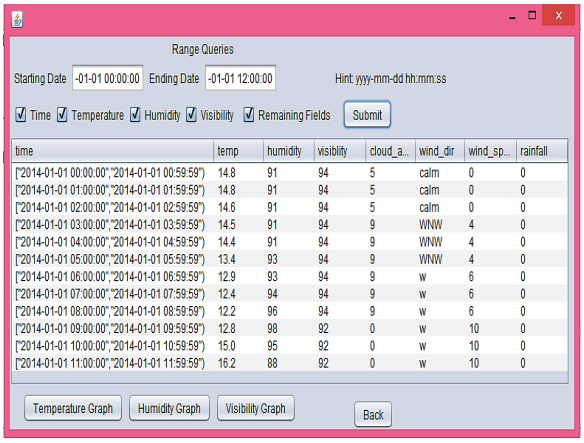


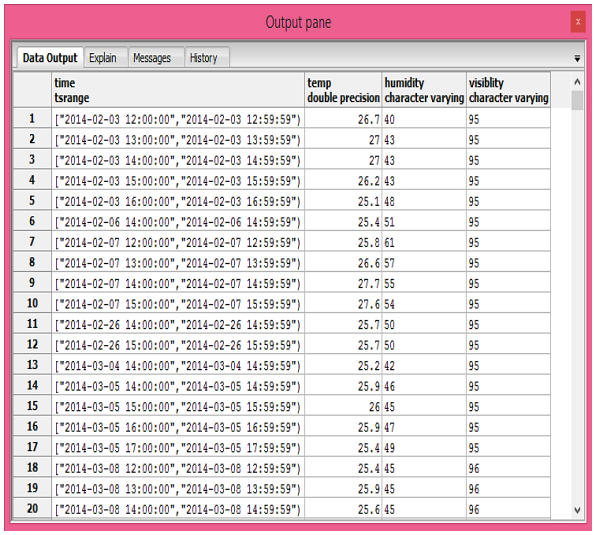

**Figure 2: Output of Query 1**

**Example 2: Find the weather details when the temperature lies between 25 and 30 degree Celsius using intersection.**

**Query:**

```
SELECT we.time_range, we.temp,we.humidity,we.visiblity FROM (SELECT * FROM wcperhour WHERE wcperhour.temp >25) we INTERSECT
SELECT ew.time_range, ew.temp,ew.humidity,ew.visiblity FROM(SELECT * FROM wcperhour WHERE wcperhour.temp < 30)ew ORDER BY time_range;
```

**Output:** The result of above query is shown in Figure 3.


Output pane

Data Output | Explain | Messages | History

| | time tsrange | temp double precision | humidity character varying | visiblity character varying |
|---|---|---|---|---|
| 1 | ["2014-02-03 12:00:00","2014-02-03 12:59:59") | 26.7 | 40 | 95 |
| 2 | ["2014-02-03 13:00:00","2014-02-03 13:59:59") | 27 | 43 | 95 |
| 3 | ["2014-02-03 14:00:00","2014-02-03 14:59:59") | 27 | 43 | 95 |
| 4 | ["2014-02-03 15:00:00","2014-02-03 15:59:59") | 26.2 | 43 | 95 |
| 5 | ["2014-02-03 16:00:00","2014-02-03 16:59:59") | 25.1 | 48 | 95 |
| 6 | ["2014-02-06 14:00:00","2014-02-06 14:59:59") | 25.4 | 51 | 95 |
| 7 | ["2014-02-07 12:00:00","2014-02-07 12:59:59") | 25.8 | 61 | 95 |
| 8 | ["2014-02-07 13:00:00","2014-02-07 13:59:59") | 26.6 | 57 | 95 |
| 9 | ["2014-02-07 14:00:00","2014-02-07 14:59:59") | 27.7 | 55 | 95 |
| 10 | ["2014-02-07 15:00:00","2014-02-07 15:59:59") | 27.6 | 54 | 95 |
| 11 | ["2014-02-26 14:00:00","2014-02-26 14:59:59") | 25.7 | 50 | 95 |
| 12 | ["2014-02-26 15:00:00","2014-02-26 15:59:59") | 25.7 | 50 | 95 |
| 13 | ["2014-03-04 14:00:00","2014-03-04 14:59:59") | 25.2 | 42 | 95 |
| 14 | ["2014-03-05 14:00:00","2014-03-05 14:59:59") | 25.9 | 46 | 95 |
| 15 | ["2014-03-05 15:00:00","2014-03-05 15:59:59") | 26 | 45 | 95 |
| 16 | ["2014-03-05 16:00:00","2014-03-05 16:59:59") | 25.9 | 47 | 95 |
| 17 | ["2014-03-05 17:00:00","2014-03-05 17:59:59") | 25.4 | 49 | 95 |
| 18 | ["2014-03-08 12:00:00","2014-03-08 12:59:59") | 25.4 | 45 | 96 |
| 19 | ["2014-03-08 13:00:00","2014-03-08 13:59:59") | 25.9 | 45 | 96 |
| 20 | ["2014-03-08 14:00:00","2014-03-08 14:59:59") | 25.6 | 45 | 96 |

**Figure 3: Output of Query 2**

## 5. Conclusion and Future Scope

An effective and powerful temporal information system should incorporate good temporal data model and powerful query language. The effective retrieval of the time variant data is very important for the decision making and analysis in the area of weather forecasting. In this paper, we have proposed a temporal model for the retrieval of temporal climate data using the meteorological dataset of a particular location in the National Capital Region of Delhi for the year 2014.The results may be used for the analysis of climate trends.

We have discussed the semantics of the proposed model. The temporal relational algebra is also discussed. Our proposed model is based on tuple time-stamping approach with valid time as the time dimension. The PostgreSql version 9.4 built-in range data type “tsrange” is being used to record the timestamp. The model also takes account of the preprocessing of the database to eradicate the redundancy of the data values. In the future, we wish to extend our model to take care of other refinements apart from the ones that are covered in this paper.

## References

[1] Kumar, Shailender. "A Relative Analysis of Modern Temporal Data Models.”IEEE International Conference on Computing for Sustainable Global Development, 2016.

[2] Atay, C. "A Comparison of Attribute and Tuple Time Stamped Bitemporal Relational Data Models." Int. Conf. on Applied Computer Science. 2010.

[3] Terenziani, Paolo. "Irregular Indeterminate Repeated Facts in Temporal Relational Databases." IEEE Transactions on Knowledge and Data Engineering 28.4 (2016): 1075-1079.

[4] Anselma, Luca, Paolo Terenziani, and Richard T. Snodgrass. "Valid-time indeterminacy in temporal relational databases: Semantics and representations." IEEE Transactions on Knowledge and Data Engineering25.12 (2013): 2880-2894.

[5] Kumar, Shailender. "A Study of Temporal Database Research." International Journal of Artificial Intelligence and Knowledge Discovery 3.3 (2013): 1-8.

[6] Bohlen, Michael H., Renato Busatto, and Christian S. Jensen. "Point-versus interval-based temporal data models." Data Engineering, 1998. Proceedings. 14th International Conference on. IEEE, 1998.

[7] Ali, Noraida Haji, and Sumazly Sulaiman. "Managing News Archive Using Temporal Data Modeling." Journal of Applied Sciences 12.3 (2012): 284-288.

[8] AL-romema, Nashwan. "Memory storage issues of temporal database applications on relational database management systems." Journal of Computer Science 6.3 (2010).

[9] Halawani, Sami M., et al. "Retrieval Optimization Technique for Tuple Timestamp Historical Relation Temporal Data

Model." Journal of Computer Science 8.2 (2012): 243.

[10] Kvet, Michal, Karol Matiaško, and Monika Vajsová. "Sensor based transaction temporal database architecture." Factory Communication Systems (WFCS), 2015 IEEE World Conference on. IEEE, 2015.

[11] Jensen, Christian S., and Richard T. Snodgrass. "Temporal data management." Knowledge and Data Engineering, IEEE Transactions on 11.1 (1999): 36-44.

[12] https://www.postgresql.org/docs/9.5/static/docguide.html