Management of Bi-Temporal Properties of Sql/Nosql Based Architectures – A Review

Lyn Kurian, Jyotsna A
Department of Computer Science and Information Systems Engineering
Rajagiri School of Engineering and Technology, Cochin, Kerala, India

Abstract- Data engineering is the most important field in computer science engineering. Data is computed, stored and manipulated according to the user requirement. Data can be text, picture, audio, video or document which is in various formats. This paper deals with various database management systems that stores and manipulates data efficiently, including banking system, scientific or commercial systems. Traditionally, we use RDBMS like MS SQL Server, IBM DB2, Oracle, MySQL and Microsoft Access for transactional processing and analytical processing. On the advent of Big Data, the strict RDBMS is not a customized solution on considering the performance and scalability aspects of the Information Technology that needs today. The new era needs new technologies. Google introduced the concept of NoSQL Databases in 2005, which led to the revolution of NoSQL databases like Noe4j, HBase, Redis, Mango DB etc. But migration to NoSQL Databases is a challenging area for the database architects in various fields of business. Different tools and techniques for database migration, query translation and query optimization is being adopted and the research area is open. This paper comprises the categorization of the proposed and implemented bi-temporal databases along with their bi-temporal properties till date.

Keywords –SQL databases, NoSQL databases, Bi-temporal Databases

I. INTRODUCTION

RDBMS are dominating structures in database management systems because of the services they provide. RDBMS applies relational model. All of them use the SQL language and they differ in their enhancements. The RDBMS popularity faced a set of challenges due to emergence of the INTERNET, and the vast amount of data is to be handled.

The important properties of RDBMS are performance and scalability but that is not feasible for many of the newly distributed applications. The complexity of ACID (Atomicity, Consistency, Isolation, and Durability) feature in design of RDBMS to ensure the transactional reliability is not required in some applications and is the reason for other aspects such as the performance. For example, in social networks, high scalability with high degree of structure, flexibility and set of simple operations is awaited. Due to these, many systems emerged to support scalability. They depend on a set of simple operations and do not consider the strict relational databases design principles. These systems are called NoSQL (Not only SQL). The term NoSQL generally refers non-relational databases.

Big data played the role for NoSQL and for its growing popularity. The continuous availability for data required NoSQL databases. Google was the leader in adopting these systems by using BigTable in 2006, followed by Dynamo introduced by Amazon in 2007. They have the properties such as ability to scale rapidly, performance, continuous availability and partition tolerance which overcome even the different versions of relational database model.
But the majority systems do not support bi-temporal properties that are an assertions in many applications. The temporal data management is seen important in many well-known applications which includes insurance, airline ticket reservations, medical applications and more. The temporal data management is implemented in the RDBMS, and temporal databases are mostly the extended RDBMS itself. On the other hand, the temporal properties need to be implemented and used in NoSQL databases that produces large amount of timestamped data, such as sensor data, financial tickers and e-commerce.

[5] NoSQL databases divides into four main categories: Key-value stores, Document stores, Column Oriented Stores, and Graph Databases. NoSQL databases are heterogeneous and has different structures, and thus they have unique properties which categorizes them. Embedding bi-temporal properties into these databases are discussed here in the paper.

This paper continues as follows. Literature review is explained in section II. Concluding remarks are given in section III.

II. LITERATURE REVIEW

A. A Generalisation Approach to Temporal Data Models and their Implementations

A Temporal Relational DBMS : TimeDB –

In this dissertation [8], the author showcases that, a temporal data model has:

1. Valid time and/or transaction time, also, other time lines.
2. Temporal data structures that do not make any assumptions about a specific type system or the presence of specific time attributes.
3. Temporal data structures allows time stamping of all constructs supported by the model.
4. Temporal data structures overcome the vertical temporal anomaly.
5. A temporally complete query language or algebra with snapshot reducible semantics.
6. Temporal constraints with snapshot reducible semantics.

The bi-temporal relational DBMS, Time DB implements the language ATSQL2 which is based on the standard query language SQL. ATSQL2 includes not only a bi-temporal query language, but also a bi-temporal modification, data definition and constraint specification language.

It is the result of integrating three different approaches, namely
1. ATSQL2, a temporal query language based on SQL.
2. Chrono Log, introducing the concept of temporal completeness.
3. Bi-temporal ChronoSQL, featuring a bi-temporal query language

Methodology

The translation algorithm used in Time DB is the following: a temporal query is translated into a temporal algebra expression using the temporal set operations union ($\cup^{t)}$, intersect ($\cap^t$) and difference ($\backslash^t$). Each argument to these set operations is either a simple algebra expression using a temporal select (σ^t), project (π^t) and cross product ($\times^t$) operation or the result of another temporal set operation. Each simple algebra expression using only a combination of a select, project and cross product operation can then be evaluated separately using a standard SELECT-FROM-WHERE statement. These intermediate results are stored in temporary tables and then used to calculate other parts of the expression.

In Time DB, a valid-time interval $I = [vts_#$ - vte_#$), closed at the lower bound and open at the upper, is mapped internally to two attributes vts_#$ (valid-time start) and vte_#$ (valid time end). Thus, each valid-time relation will have two additional (hidden) attributes. Transaction-time tables are extended accordingly. Bi-temporal tables contain attributes for both valid time and transaction time.

Bi-temporal Algebra Operations

Time DB handles bi-temporal queries. For these queries, special operations for set union, set difference, set intersection and cross product need to be implemented. A bi-temporal query is translated to standard SQL the same way as a unitemporal query. It uses bi-temporal algebra operations instead.

I. Bi-temporal Set Union Operation

The bi-temporal union operation ($\cup^{vt}$) of two union-compatible bi-temporal relations R1 and R2, having attributes R1 = R2 = is the same as the non-temporal union,

$$R1\ (\cup^{vt})\ R2 \equiv R1 \cup R2$$

Consider the bi-temporal relation R1 contains the tuple

TRANSACTION	VALID	NAME	DEPT
[1993-1997]	[1986-1993]	John	Sale

And relation R2 contains the tuple

TRANSACTION	VALID	NAME	DEPT
[1990-1995]	[1984-1991]	John	Sale

The bi-temporal set union operation returns a bi-temporal relation containing two tuples. The query

```
 VALID AND TRANSACTION
        (SELECT * FROM R1
        UNION
        SELECT * FROM R2);
```

returns the bi-temporal relation:

TRANSACTION	VALID	NAME	DEPT
[1993-1997]	[1986-1993]	John	Sale
[1990-1995]	[1984-1991]	John	Sale

II. BI-TEMPORAL SET DIFFERENCE OPERATION

Depending on the sequence - VALID AND TRANSACTION or TRANSACTION AND VALID - used in the bi-temporal query, the resulting timestamp is set up differently. In Time DB, the bi-temporal difference R1 ($\backslash^{vt}$) R2 can be expressed in two different ways. The query

```
VALID AND TRANSACTION
        (SELECT * FROM R1
        EXCEPT
        SELECT * FROM R2);
```

returns the bi-temporal relation:

TRANSACTION	VALID	NAME	DEPT
[1993-1995]	[1991-1993]	John	Sale
[1995-1997]	[1986-1993]	John	Sale

And also the query

```
TRANSACTION AND VALID
```

```
    (SELECT * FROM R1
    EXCEPT
    SELECT * FROM R2);
```

returns the bi-temporal relation:

VALID	TRANSACTION	NAME	DEPT
[1986-1991]	[1995-1997]	John	Sale
[1991-1993]	[1993-1997]	John	Sale

III. BI-TEMPORAL SET INTERSECTION OPERATION

The bi-temporal set intersection of the two relations R1 and R2, R1 ($\cap^{vt}$) R2, can be written as

$$R1\ (\backslash^{vt})\ (R1\ (\backslash^{vt})\ R2)$$

The query in Time DB is written as

```
VALID AND TRANSACTION
    (SELECT * FROM R1
    INTERSECT
    SELECT * FROM R2);
```

and returns the bi-temporal relation:

TRANSACTION	VALID	NAME	DEPT
[1993-1995]	[1986-1991]	John	Sale

IV. BI-TEMPORAL CROSS PRODUCT OPERATION

The bi-temporal cross product of R1 and R2, R1 ($\times^{vt}$) R2, can be translated into the following SQL statement:

```
SELECT A1, A2, . . . , An, B1, B2, . . . , Bm,
    GREATEST (R1. v t s_#$, R2. v t s_#$) v t s _#$,
    LEAST (R1. v t e_#$, R2. v t e_ #$) v t e_#$,
    GREATEST(R1. t t s_#$, R2. t t s_ #$) t t s _#$,
    LEAST(R1. t t e_ #$, R2. t t e _#$) t t e_#$
FROM R1, R2
WHERE GREATEST(R1. v t s _#$, R2. v t s_ #$) < LEAST(R1. v t e_ #$, R2. v t e_ #$)
AND GREATEST(R1. t t s_ #$, R2. t t s_ #$) < LEAST(R1. t t e_ #$, R2. t t e_ #$)
```

The bi-temporal cross product query can be written as

```
VALID AND TRANSACTION
    (SELECT * FROM R1, R2);
```

and returns the bi-temporal relation:

TRANSACTION	VALID	NAME	DEPT	NAME	DEPT
[1993-1995]	[1986-1991]	John	Sale	John	Sales

V. BI-TEMPORAL SELECTION OPERATION

Time DB supports the predicates =, precedes, overlaps, meets and contains.

VI. BI-TEMPORAL PROJECTION OPERATION

The bi-temporal projection operation is same as its non-temporal counterpart. The only difference is whether or not the timestamp attributes are in the projection.

B. Implementing bi-temporal properties into various NoSQL database categories

B.1 KEY-VALUE

A Key-value stores, the data is stored as Key Value pairs. The authors of this paper [2] proposes the below mentioned methodologies.

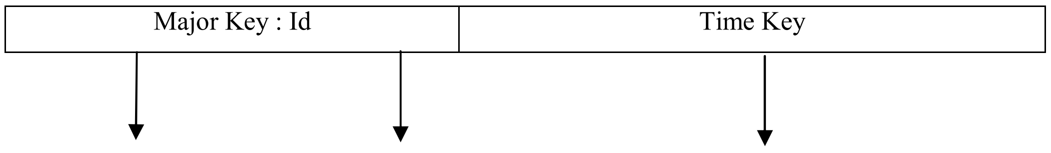
Methodology

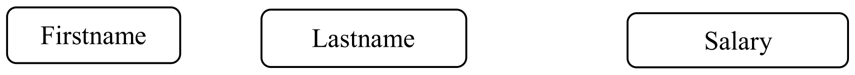
One of the solutions is to add two attributes in each Key-value pair, one for the valid time in the form of validTimeStart and validTimeEnd as attributes.

ROW		
ID=1	Firstname	ARAVIND
	Lastname	GOPAN
	Salary	100000
	Validtimestart	25-8-2020
	Vaildtimeend	15-9-2020
	Transactiontime	1-7-2020

Here, the key is 1 and it points to the set of values (Firstname, Lastname, Salary, and Validtime). The salary is a field of data that is affected by the time. The temporal property of the field salary, is managed by the valid time, in each inserted Key-value pair we keeps track of transaction time and the valid time. Transaction time is part of the key so it will be able to retrieve chain of time series information.

A clearer and more structured method leads us to the idea to divide the key into two parts to form a composite key. This composite key constitutes a part that identifies the row, the regular key and the interval of the valid time. In such a way, the key is suitable for extracting data accordingly that satisfies the validity of the information.





B.2 : COLUMN ORIENTED STORES

Column oriented NoSQL databases are motivated from big table. The main idea is to have structured data that can scale out to larger size. It has three main components the key, the column family and the super column. Row keys are the main object in data distribution and partitioning. The authors of this paper [2] proposes the below mentioned methodologies.

Bi-temporal Implementation

The row key property is exploited in the model for temporal data management. Values belonging to the same column are stored separately on the same disk. Columns are grouped into super columns.
One way is to use the timestamps to manipulate valid time or transaction time, timestamps are more suitable to illustrate transaction time. Furthermore, maintaining valid time can be done by adding additional columns to express the valid time interval. Adding two columns is trivial and straightforward solution, the first column will represent the valid time start and the second one is to represent valid time end.

In the example given below, we can track the salary of the employee using the validation time stored, i.e., the added two columns.

ID=1	Name	Salary	vtFrom	vtTo	Tt
	Thira	30000	1	2	1/1/2015
	Thira	40000	3	5	1/6/2017
	Thira	50000	5	∞	10/10/2019

III. CONCLUSION

The four ways in which bi-temporal databases can be implemented using temporal database applications are [8]

• Use a type date and implement all temporal semantics in the application program
• Specify an ADT for time that is the basis in temporal applications to timestamp and query the temporal data
• Extend a non-temporal data model to support time-varying data
• Generalize a non-temporal data model into a temporal model

The advantage of the first two approaches is that they do not need any changes to be made to the data model and system used. While these approaches in some way support the data structures and functionality that supports time-varying data, they cannot, however, exploit the advantage of the special semantics time has, for example, for optimization.

The last two approaches can only be achieved by modifying both the data model and corresponding systems. With respect to these approaches, if changes are done to both the data model and corresponding systems, they should be general and orthogonal, supporting temporal data structures, temporal operations and temporal constraints without any unnatural restrictions. In other words, the generalization approach considers all constructs and concepts of a data model.

With respect to the second approach, the ADT was implemented using the object oriented DBMS and also the implementation of the temporally complete language ATSQL2 which is based on extending relation schemas to store time-varying data. The resulting bi-temporal DBMS Time DB supports a temporal query language, a temporal data definition and modification and a temporal constraint specification language. Time DB is a front-end to the commercial DBMS Oracle.

[2] When coming to the NoSQL database, the need for bi-temporal implementation is obvious because of the increasing popularity of NoSQL databases and the vast number of applications that require temporal data management. The two main categories of NoSQL databases and proposed set of solutions to describe how to manage bi-temporal characteristics. The solution to store valid and transactions times in Key value databases, and solutions to handle the bi-temporal properties in column oriented stores are studied. In the future work, it is possible to embed the bi-temporal properties in document and graph databases.

IV. REFERENCES

[1] Jiao Dai SQL to NoSQL : What to do and How . IOP Conf. Series: Earth and Environmental Science 234 (2019) 012080

[2] Mohammed Eshtay, Assam Sleit, Monther Aldwairi Implementing Bi-Temporal Properties Into Various Nosql Database Categories. International Journal of Computing, 18(1) 2019, 45-52

[3] Gerhard F. Knolmayer, Thomas Myrach Concepts of Bi-temporal Database Theory and the Evolution of Web Documents. Proceedings of the 34th Hawaii International Conference on System Sciences - 2001

[4] Alexander Campos, Jorge Mozzino, and Alejandro Vaisman Towards Temporal Graph Databases. arXiv:1604.08568v2 [cs.DB] 2 May 2016.

[5] Morgan D. Monger, Ramon A. Mata-Toledo, Pranshu Gupta Temporal Dtata Management In NoSQL Databases.

[6] Safa Brahmia1, Zouhaier Brahmia, Fabio Grandi, and Rafik Bouaziz τJSchema: A Framework for Managing Temporal JSON- Based NoSQL Databases. c Springer International Publishing Switzerland 2016 S. Hartmann and H. Ma (Eds.): DEXA 2016, Part II, LNCS 9828, pp. 167–181, 2016. DOI: 10.1007/978-3-319- 44406-213

[7] Yong Hu and Stefan Dessloch Defining Temporal Operators for Column Oriented NoSQL Databases. Y. Manolopoulos et al. (Eds.): ADBIS 2014, LNCS 8716, pp. 39–55, 2014. Springer International Publishing Switzerland 2014

[8] ANDREAS STEINER A Generalisation Approach to Temporal Data Models and their Implementations. Diss. ETH No. 12434