CHECKING AND VERIFYING TEMPORAL DATA VALIDITY USING VALID TIME TEMPORAL DIMENSION AND QUERIES IN ORACLE 12C

Jaypalsinh A. Gohil[1] and Dr. Prashant M. Dolia[2]

[1]Assistant Professor & Research Scholar, C. U. Shah College of MCA,
C. U. Shah University, Wadhwan City, India
[2]Associate Professor & Research Guide, Department of Computer Science,
MK Bhavnagar University, Bhavnagar, India

ABSTRACT

Temporal Database is the most convenient form to represent time element associated with data. Temporal validity support a unique feature of temporal database lets you associate one or more valid time dimensions with a table and have data be visible depending on its time-based validity, as determined by the start and end dates or time stamps of the period for which a given record is considered to be a valid record. This study focuses on checking and verification of temporal data using valid time dimension of temporal database. This study covers the steps for adding a valid time dimension on a table, and various methods for querying the table and retrieving records based on a specified valid time value or range with help of Oracle 12c.

KEYWORDS

Temporal Database, Temporal Validity, Valid Time, Transaction Time, AS of PERIOD FOR, VERSIONS PERIOD FOR.... BETWEEN.

1. INTRODUCTION

A temporal database contains time-varying data. Time is an important aspect of all real-world phenomena. Events occur at specific points in time; objects and the relationships among objects exist over time.

Temporal data stored in a temporal database is different from the data stored in non-temporal database in that a time period attached to the data expresses when it was valid or stored in the database. As mentioned above, conventional databases consider the data stored in it to be valid at time instant now, they do not keep track of past or future database states. By attaching a time period to the data, it becomes possible to store different database states.

Majority of database applications associate the temporal features of data. The most common example of such applications involves insurance records, banking and accounting, record management, portfolio management, medical history of patients, schedule and ticket booking for airline or train, hotel reservation and even for company's project management. The above mention application depends on temporal data, which records time element for data [1].

Temporal database stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time, for example, the history of the stock market or the movement of employees within an organization. Thus, a temporal database stores a collection of time related data [2].

Online and real-time database manages modification in the database through events. We always relate events with temporal information. Every event that occurred at a specific time has some supplementary information. Any event reflects its time-stamp and time-interval information [3]. Computer based real-time information system highly dependent on database system for storing and retrieving all kind of information system also must respond to events that generates in the information system in stipulated time limit.

2. HISTORY IN TEMPORAL DATABASE

A temporal database is like any other database, but it also records and maintains passage of time. Traditional databases, during old days were only made to store strings of text and numbers, and they did not recognize the passage of time. It can leads to various problems, the most general one being that time-based events could not be tracked from beginning to end, but it only reflects event's present state. One more side effect my arise if a primary key, or row name, was associated with a date, then that primary key could be used over and over, which resulted in redundant data [4].

To minimize above reflected problems, the database community was invited to make a temporal variable that could be integrated into the database format. The temporal database was created in 1993 and implemented in 1994. Due to this revolution, temporal databases were able to track when an event began and when it ended [4].

Historical information can be stored systematically and in uniformed manner using temporal databases [5]. It provides a unique platform to store and manipulate time-varying information. Some of the unique property of temporal database includes identification of an appropriate data type for time, prevent fragmentation of an object description; provide query algebra to deal with temporal data, compatible with old database without temporal data [6]. Temporal databases encompass all database applications that require some aspect of time when organizing their information.

The history of an "object" of the real world or of a database is the temporal representation of that object. Each object, can have attribute histories, entity histories, relationship histories, schema histories, transaction histories, etc. "History" is a general concept, intended in the sense of "train of events connected with a person or thing."

In the glossary of temporal databases, the concept of history is added to include multiple time dimensions as well as multiple data models. So we can have, e.g., valid-time histories, transaction-time histories, and bitemporal histories [7].

The term "history," defined formally or informally, has been used in many temporal database paper to represent time aspect of any particular object.

3. TIME REPRESENTATION IN TEMPORAL DATABASE

Temporal databases includes mainly two time dimensions namely valid time and transaction time. Valid time and transaction time can be merged to create bitemporal data.

- **Valid time** is the time period during which a fact is true with respect to the real world.
- **Transaction time** is the time period during which a fact stored in the database is considered to be true.
- **Bitemporal** data combines both Valid and Transaction Time.

There is a possibility to have timelines other than valid time and transaction time, such as decision time, in the database. In such situations it is called a multitemporal database.

The foundation block for any database system are called facts. Majority of temporal aspects depends on facts. The most important aspect of temporal database is valid time dimension. Valid time dimension represents past, present and future information when facts are true in real world[8]. Valid time aspect records time-varying states of real world. Valid time dimension can attached with all the facts, but it is not necessary that it is always recorded in the database due to many reasons. Let’s taken an case where valid time may not be known. The facts can have more than one valid time records associated with it, because either past, present or future valid time information needs to be recorded along with data in the database.

One can also attach a transaction time with facts. The transaction time of fact reflects the time element when the fact is current in the database. The entities can also be assigned a transaction time and not just facts. For and illustration consider that the numeric value “42” is stored in the database, but only numeric value cannot convey anything. In this situation valid time is relevant; instead we associate transaction time with value “42”. Similarly we can associate transaction time with all database entities [9]. Transaction time highlight the duration, like from insertion to deletion, so more than one insertion and deletion can be possible for same entity [10]. Deletion process is logical and deleting an entity does not permanently physically remove the entity from database. Transaction time records the time-varying states of database and provides traceability.
Valid time allows temporal database to associate a valid time dimension with a relation or table and data said to be valid depending on its validity of time. This validity of time is determined by start and end dates or timestamps of the period for which a given tuple or record is considered to be valid [11].

The following are the various features of Temporal Database [12]:

- A time period data type, including the ability to represent time periods with no end (infinity or forever)
- The ability to define valid and transaction time period attributes and bitemporal relations
- System-maintained transaction time
- Temporal primary keys, including non-overlapping period constraints

- Temporal constraints, including non-overlapping uniqueness and referential integrity
- Update and deletion of temporal records with automatic splitting and coalescing of time periods
- Temporal queries at current time, time points in the past or future, or over durations.

4. DIFFERENT FORMS OF TEMPORAL DATABASES

The temporal database supports two different dimensions of time i.e. valid time and transaction time. These two aspect of temporal database provides distinction between various forms of temporal databases. A historical database stores data with respect to valid time, a rollback database stores data with respect to transaction time. A bitemporal database stores data with respect to both valid time and transaction time [13].

The traditional DBMS are designed to store only a single state of the real world, usually the most recent state. Such databases usually are called snapshot databases, which maintains only the last (current) state of database. The following figure represents a snapshot database in the context of valid time and transaction time:

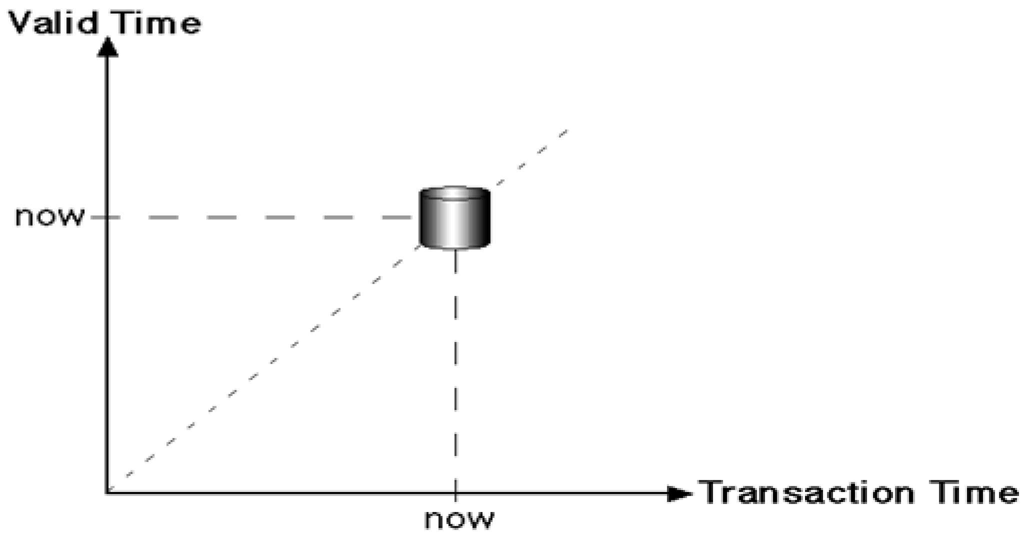

Figure 1. Snapshot of temporal database reflecting the most recent state of database.
Source: © TimeConsult - May 21, 2005

Valid time and temporal time aspects can be combined using bitemporal relation. So it provides both historical and rollback information. Let’s consider the sample historical information say, “When Amar did joined the company?” is provided by valid time and rollback information say, “When Amar’s working department was is changed?” is provided by transaction time.

The single fact do not have same valid time and transaction time. For example consider the following temporal relation employee.

Table 1. Some of the records of temporal relation employee

EMP_ID	EMP_NAME	DEPT	SALARY	VALID_TIME_START	VALID_TIME _END
510	Jimit	Research	55000	2005	2010
510	Jimit	Research	60000	2010	2013
510	Jimit	Sales	65000	2013	now
521	Pratik	Research	50000	2008	2014
533	Ganesh	Research	52500	2011	now
552	Reshma	Sales	65000	2008	now

The above given relation stores the history of the employees as a valid time. The attributes ValidTimeStart and ValidTimeEnd actually represent a time interval which is closed at its lower and open at its upper bound. So, we see that during the time period [2005 – 2010], employee Jimit was working in the research department, having a salary of 55000. Then in 2010 he transferred to the sales department, with increase in salary amounting to 60000. In 2013, he agin got a salary raise with total salary amounting to 65000. The upper bound "now" denotes that the tuple is valid until further notice. Note that it is now it is possible to store information about past states. As per data in the table Pratik was employed from 2008 until 2014. Since in the table we does not have current version of Pratik's record i.e. it does not have "now" in upper bound of any records related to Pratik, it means that Pratik is left the company and it does not have any current version of Pratik's record in the temporal table. In the corresponding non-temporal table, this information was (physically) deleted when Pratik left the company.

Now to crate and translate table which we have discussed above the Oracle 12c provides Valid time temporal support which is typically used with Oracle Flashback technology, to perform AS of PERIOD FOR and VERSION PERIOD FOR.....BETWEEN queries that specify the valid time period [13].

5. TEMPORAL FEATURES SUPPORT IN ORACLE 12C

Oracle introduced Oracle Database 12c on June 25, 2013, which is considered to be the important architectural transformation in the legacy of the world's leading database in its 25 years with respect to market presence and dominance [12]. The first outlook of Oracle Database 12c was unveiled during Oracle Open World in San Francisco in September 2012.

Oracle 12c supports temporal validity using Oracle flashback Technology using valid time period clauses like AS OF and VERSIONS BETWEEN [11].

DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME procedure can also be used to specify an option for the visibility of table data: all table data, data valid at a specified time, or currently valid data within the valid time period [13].

Temporal Validity feature of Oracle 12c allows the user to add (one or more) time dimensions to a table by using current columns or using columns automatically created by database. It also enables a simple SQL syntax to filter the columns to access only active data using Oracle flashback technology [13].

6. USING VALID TIME TEMPORAL DIMENSION AT THE TIME OF TABLE CREATION

6.1. Creating a table with Valid_Time temporal dimension

In the following discussion we illustrated how one can add temporal dimension while creating a relation (table).

```
SQL> CREATE TABLE CUSTOMER
     (
     CUST_ID     NUMBER(10) PRIMARY KEY,
     CUST_NAME   VARCHAR2(30) NOT NULL,
     CONTECT_NO  NUMBER(10) NOT NULL
     );
     Table created.
SQL> CREATE TABLE DTH_PACKAGE
     (
     PACKAGE_ID     NUMBER(10) PRIMARY KEY,
     PACKAGE_NAME   VARCHAR2(20) NOT NULL,
     PACKAGE_RENT   NUMBER(10,2) NOT NULL
     );
     Table created.

SQL> CREATE TABLE CUSTOMER_PACKAGE
     (
     ID          NUMBER(10) PRIMARY KEY,
     CUST_ID     NUMBER(10) REFERENCES CUSTOMER(CUST_ID),
     PACKAGE_ID  NUMBER(10) REFERENCES DTH_PACKAGE(PACKAGE_ID),
     START_DATE  DATE,
     END_DATE    DATE,
     PERIOD FOR  customer_package_period   (START_DATE,END_DATE)
     );
     Table created.
```

As visible from the above example that by using the PERIOD FOR clause while creating a table one can add the Valid_Time temporal dimension. See how Valid_Time temporal dimension is added in CREATE TABLE command of CUSTOMER_PACKAGE in above example [14].

6.2. Applying Valid_Time temporal dimension in existing table

By using the following approach one can add temporal dimension in existing table [14].

6.2.1. Period creation from existing columns:

If already exist table has START_DATE and END_DATE columns in the table then period can be added in the following way:

```
SQL> CREATE TABLE CUSTOMER_PACKAGE
     (
     ID          NUMBER(10) PRIMARY KEY,
     CUST_ID     NUMBER(10) REFERENCES CUSTOMER(CUST_ID),
     PACKAGE_ID  NUMBER(10) REFERENCES DTH_PACKAGE(PACKAGE_ID),
     START_DATE  DATE,
     END_DATE    DATE
     );
     Table created.
```

We can use the following ALTER TABLE command to add Valid_Time temporal dimension period to the existing table.

```
SQL> ALTER TABLE customer_package ADD PERIOD FOR
customer_package_period (START_DATE, END_DATE);
Table altered.
```

6.2.2. Period creation system generated hidden columns:

If we have existing table without START_DATE and END_DATE columns as shown in the below example:

```
SQL> CREATE TABLE CUSTOMER_PACKAGE
     (
     ID          NUMBER(10) PRIMARY KEY,
     CUST_ID     NUMBER(10) REFERENCES CUSTOMER(CUST_ID),
     PACKAGE_ID  NUMBER(10) REFERENCES DTH_PACKAGE(PACKAGE_ID)
     );
     Table created.
```

Then by using the ALTER TABLE command we can add the Valid_Time temporal dimension period to the already existing table. The following ALTER TABLE command adds two hidden columns to the table CUSTOMER_PACKAGE table: USER_VALID_TIME_START and USER_VALID_TIME_END.

```
SQL>   ALTER   TABLE   customer_package   ADD   PERIOD   FOR
customer_package_period;

Table altered.
```

7. TEMPORAL QUERIES ON A TABLE WITH VALID_TIME SUPPORT

As discussed earlier valid time periods can be added with AS of PERIOD FOR clause in a table using start and end DATE or TIMESTAMP columns. We can also use VERSIONS PERIOD FOR ... BETWEEN clause to search the records which are valid between two given time points (intervals). We can also use this valid time periods to query a table [13].

7.1. AS of PERIOD FOR temporal query clause

Insert the following data in a table before implementing AS of PERIOD FOR clause:

```
SQL> SELECT * FROM CUSTOMER;

   CUST_ID CUST_NAME                      CONTECT_NO
---------- ------------------------------ ----------
         1 AMAR                           5674889923

         2 BHAVESH                        4468975213

         3 CHAITALI                       3232158872

         4 DISHA                          5567821346

SQL> SELECT * FROM DTH_PACKAGE;

PACKAGE_ID PACKAGE_NAME         PACKAGE_RENT

---------- -------------------- ------------

         1 PLATINUM                      499

         2 GOLD                          460

         3 SUPER FAMILY                  335


SQL> SELECT * FROM CUSTOMER_PACKAGE;

        ID    CUST_ID PACKAGE_ID START_DAT END_DATE

---------- ---------- ---------- --------- ---------

         1          1          1 01-JAN-12 10-FEB-12

         2          1          2 01-FEB-12 15-MAR-12
         3          1          3 01-JAN-12 01-APR-12
         4          2          1 01-JAN-12 10-FEB-12
         5          2          2 01-FEB-12 15-MAR-12
         6          2          3 01-JAN-12 01-APR-12
         7          3          1 01-JAN-13 10-FEB-13
         8          3          2 01-FEB-13 15-MAR-13
         9          3          3 01-JAN-13 01-APR-13
        10          4          1 01-JAN-14 10-FEB-14
        11          4          2 01-FEB-14 15-MAR-14
        12          4          3 01-JAN-14
SQL>COMMIT;
```

Now to find out which customer has subscribed which package on a specific date? We could write the following query using AS of PERID FOR temporal clause [14]:

```
SQL> SELECT customer_package.start_date,
     customer_package.end_date,
     customer.cust_name,
     dth_package.package_name
     FROM   customer_package
     AS OF PERIOD FOR customer_package_period
     TO_DATE('12-FEB-2013','DD-MON-YYYY')
     JOIN customer ON customer_package.cust_id = customer.cust_id
     JOIN dth_package ON customer_package.package_id =
dth_package.package_id ORDER BY 1, 2, 3;
```

The query outputs the records of the customers along with its active dth packages as on a specific date i.e. 12-FEB-2013.

```
OUTPUT:
START_DAT END_DATE  CUST_NAME                       PACKAGE_NAME
--------- --------- ------------------------------ ------------
01-JAN-13 01-APR-13 CHAITALI                        SUPER FAMILY
01-FEB-13 15-MAR-13 CHAITALI                        GOLD
2 rows selected.
```

By using the SYSDATE one can display customers along with their active packages as on today which shown in below query.

```
SQL> SELECT customer_package.start_date,
     customer_package.end_date,
     customer.cust_name,
     dth_package.package_name
     FROM   customer_package AS OF PERIOD FOR
customer_package_period SYSDATE
     JOIN customer ON customer_package.cust_id = customer.cust_id
     JOIN dth_package ON customer_package.package_id =
dth_package.package_id ORDER BY 1,2,3;
OUTPUT:
START_DAT END_DATE  CUST_NAME                      PACKAGE_NAME
--------- --------- ------------------------------ ------------
01-JAN-14           DISHA                          SUPER FAMILY
1 row selected.
```

7.2. VERSIONS PERIOD FOR.....BETWEEN Queries

To find customers and their active dth packageb during a specified time period one can use the VERSIONS PERIOD FOR ... BETWEEN temporal clause [14].

```
SQL> SELECT customer_package.start_date,
     customer_package.end_date,
     customer.cust_name,
     dth_package.package_name
     FROM   customer_package
     VERSIONS PERIOD FOR customer_package_period BETWEEN
     TO_DATE('12-FEB-2013','DD-MON-YYYY') AND TO_DATE('06-JAN-
2014','DD-MON-YYYY')
     JOIN customer ON customer_package.cust_id = customer.cust_id
     JOIN dth_package ON customer_package.package_id =
dth_package.package_id ORDER BY 1,2,3;
```

The above generates the records of the customer along with their active dth package on a given specific time period i.e. between 12-FEB-2013 and 06-JAN-2014. The output is as below.

```
OUTPUT:
START_DAT END_DATE  CUST_NAME                      PACKAGE_NAME
```

```
--------- --------- ------------------------------ ------------
01-JAN-13 01-APR-13 CHAITALI                       SUPER FAMILY
01-FEB-13 15-MAR-13 CHAITALI                       GOLD
01-JAN-14           DISHA                          SUPER FAMILY
01-FEB-14 15-MAR-14 DISHA                          GOLD

4 rows selected.
```

8. SUMMERY

The beginning half of paper concentrated on formal introduction of temporal database and valid time temporal support in temporal database system. In the consecutive sections we can visualise how Oracle 12c supports temporal database features along with its predicates. The primary goal of this work is to show how one can create and query a table using valid-time temporal database dimension, predicates and clause like AS of PERIOD FOR and VERSIONS PERIOD FOR.... BETWEEN supported in Oracle 12c.

9. CONCLUSION

Using temporal database features one can add the time element with data. Temporal Validity support is a crucial thing for real-time online database applications. Temporal Validity feature allows temporal database to add a time dimension to each row in table consisting of two date-time columns which shows the validity of data on given date-time. It generally improves he overall performance when data is large, it optimizes this performance by processing only current active records instead querying entire table. Oracle 12c supports temporal validity by using its various temporal validity predicates and clauses like AS of PERIOD FOR and VERSIONS PERIOD FOR.... BETWEEN.

REFERENCES

[1] Jensen, C. S. (n.d.). Introduction to Temporal Database Research. Retrieved from http://infolab.usc.edu/csci599/Fall2001/paper/chapter1.pdf

[2] Patel, J. (2003). Temporal Database System. London: Department of Computing, Imperial College, University of London.

[3] O. Etzion, S. Jajodia, and S. Sripada (eds.). (1998) “Temporal Databases: Research and Practice”, LNCS 1399.

[4] Newth, A. (n.d.). What is temporal database? Retrieved August 5, 2013, from WiseGeek: http://www.wisegeek.com/what-is-a-temporal-database.htm

[5] Richard T. Snodgrass and Ilsoo Ahn, (1986) "Temporal Databases," IEEE Computer 19(9), pp. 35–42.

[6] Christian S. Jensen and Richard T. Snodgrass, (1999) "Temporal Data Management," IEEE Transactions on Knowledge and Data Engineering 11(1):36–44.

[7] Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts – February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399, Springer, 1998, pp. 367–405.

[8] C. E. Dyreson, R. T. Snodgrass, (1998) “Supporting Valid-Time Indeterminacy”, ACM Transaction on Database Systems.

[9] C. S. Jensen, R. T. Snodgrass, (1996) "Semantics of Time-Varying Information", Information Systems.
[10] C. S. Jensen, C. E. Dyreson (eds.), (1998) "A Consensus Glossary of Temporal Database Concepts", February 1998 Version.
[11] Christian S. Jensen, James Clifford, Ramez Elmasri,Shashi K. Gadia, Pat Hayes, Sushil Jajodia (eds.) (1994). A Consensus Glossary of Temporal Database Concepts— SIGMOD RECORD, Vol. 23, No. 1.
[12] Temporal Database. (n.d.). Retrieved August 9, 2013, from wikipedia: http://en.wikipedia.org/wiki/Temporal_database
[13] J. F. Allen. "Maintaining Knowledge about Temporal Intervals" (1983), Communications of the ACM, 26(11):832–843.
[14] J. A. Gohil, P.M.Dolia, "Testing Temporal Data Validity in Oracle 12c using Valid Time Temporal Dimension and Queries" Journal of Engineering Computers & Applied Sciences(JECAS), Volume 4, No.4, April 2015

AUTHORS

Jaypalsinh A. Gohil is working as a Assistant Professor at C.U.Shah College of M.C.A., C.U.Shah University, Wadhwan City. Gujarat-India. He is having more than nine year of teaching experience at U.G. & P.G. level. He has published FIVE research papers in international journals and ONE research paper in national journal. He has also presented TEN papers at various conferences. He has also contributed by publishing TWO books in subject of Computer Science & Applications. k

Dr. Prashant M. Dolia is Associate Professor at Department of Computer Science & Applications, Bhavnagar University, Bhavnagar, Gujarat-India. He is having seventeen years of teaching experience at Post Graduate level. He has published EIGHTEEN research papers in international journals and ONE research paper in national journal. He has also published NINE books in subject of Computer Science & Applications.
.