Testing Temporal Data Validity in Oracle 12c using Valid Time Temporal Dimension and Queries

Jaypalsinh A. Gohil, Assistant Professor & Research Scholar, C. U. Shah College of MCA, C. U. Shah University, Surendranagar, Gujarat, India
Dr. Prashant M. Dolia, Associate Professor & Research Guide, Department of Computer Science, MK Bhavnagar University, Bhavnagar, Gujarat, India

Abstract
The time aspect of data can be easily represented by Temporal Database. Temporal Database allows you to associate time element with data through temporal validity support. Temporal Validity allows you to 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 work aims at testing temporal data with valid time dimensions of temporal database. By using the capabilities of Oracle 12c this work highlights sequence for adding a valid time dimension to a table, and various methods for querying the table and retrieving temporal data based on a specified valid time value or range.

Keywords: Temporal Database, Temporal Validity, Valid Time, AS OF PERIOD FOR, VERSIONS PERIOD FOR... BETWEEN.

Introduction
Temporal databases provide a uniform and systematic way of dealing with historical data [1]. It provides mechanisms 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 [2]. Temporal databases encompass all database applications that require some aspect of time when organizing their information.

Majority of today’s applications of database technology are time oriented, so basically they are temporal in nature. Typical implementation areas includes applications such as clients portfolio management, financial accounting, and banking, record-keeping applications such as personnel management, medical history of patients records, and stock management; time critical real time applications such as airline, train, and hotel reservations and project management; and scientific applications such as weather forecasting [3]. The above listed class of applications depends on temporal databases, which records data with reference of time.

Uniqueness of information is represented by time element. Majority of large scale corporations and enterprises encounters the problem when data becomes out of date. However, in longer run such data is very important, so one should provides the means for storage of such historical data. It may be possible that some of the information may be inherited from past records in the especially medical, insurance and legal information. So, Temporal Database is best suitable medium to store and manage historical data [4].

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 [5]. 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.

Temporability and real-time processing are in some way interrelated, for example due dates, insurance premium dates, medical check up dates are computed based on the temporal data in the database and events are triggered accordingly. With present database technology there are some loop holes in the integration of temporal and real-time information. This problem domain should be addressed so that we can join the capabilities of temporal and real-time information.

The facts which are collected by an entity is the building block for any database system. Many temporal semantics and aspects can be associated with facts. Out of which valid time is most crucial. It reflects the past, present and future information when facts are true in real world [6]. Ultimately valid time records the time-varying states of real world. Valid time can be associated with all facts, however the valid time information may not always be recorded in the database, for several reasons. Consider the case where valid time may not be known, or it is not relevant for some information to record it along with data in the database. Since, either past, present or future valid time information can be associated with facts, facts
may have more than one valid time records associated with it.

Similar to valid time, a transaction time can also be associated with facts. The transaction time of fact represents the time when the fact is current in the database. Transaction time can be associated with entities also and not just facts. For example suppose the numeric value “75” is stored in the database, but merely it cannot convey anything. In this case associating the valid time is not meaningful; instead we associate transaction time with value “75”. Based on above case we can associate transaction time with all database entities [7]. Transaction time aspect reflects duration, like from insertion to deletion, so more than one insertion and deletion can be possible for same entity. 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.

See that the transaction time of a database fact “FACT1”, is the valid time of related fact for “FACT1” is current in the database. Generally, the pair of valid and transaction time are related to content of all databases, and recording the same is essential for majority of real-time online applications [8].

Valid time temporal support in temporal database enables you to associate a valid time dimension with a table and data considered to be valid depending on its time-based validity, as determined by the start and end dates or timestamps of the period for which a given record is considered valid [9,11].

Valid time temporal support 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 [10].

**Temporal Validity 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 OpenWorld 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_VAL ID_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].

**Table Creation Using Valid Time Temporal Dimension**

A. Applying Valid_Time temporal dimension at the time of creation of a table

The following example shows how one can add temporal dimension while creating a table (relation).

```sql
CREATE TABLE STUDENTS
(
    STUDENT_ID     NUMBER(10) PRIMARY KEY,
    STUDENT_NAME VARCHAR2(30) NOT NULL
);

CREATE TABLE COURSE
(
    COURSE_ID      NUMBER(10) PRIMARY KEY,
    COURSE_NAME    VARCHAR2(20) NOT NULL
);

CREATE TABLE STUDENT_COURSE
(
    ID NUMBER(10) PRIMARY KEY,
    STUDENT_ID  NUMBER(10) REFERENCES STUDENTS(STUDENT_ID),
    COURSE_ID NUMBER(10) REFERENCES COURSE(COURSE_ID),
    START_DATE DATE,
    END_DATE   DATE,
    PERIOD FOR student_course_period

    START_DATE, END_DATE

)
```
The Valid_Time temporal dimension can be added by using the PERIOD FOR clause while creating a table as shown in CREATE TABLE command of STUDENT_COURSE table in above example [14].

B. Applying Valid_Time Temporal Dimension when Table is already created

In the following way one can add temporal dimension when tables are already created [14].

-- Create a period using existing columns:
If we have already created the table with START_DATE and END_DATE columns in the following manner:

```sql
CREATE TABLE STUDENT_COURSE
(
    ID NUMBER(10) PRIMARY KEY,
    STUDENT_ID  NUMBER(10) REFERENCES STUDENTS(STUDENT_ID),
    COURSE_ID NUMBER(10) REFERENCES COURSE(COURSE_ID),
    START_DATE DATE,
    END_DATE   DATE
);
```

Then we can add the Valid_Time temporal dimension to the existing table by using the following ALTER TABLE command.

```sql
ALTER TABLE student_course ADD PERIOD FOR student_course_period (START_DATE, END_DATE);
```

-- Create a period with system generated hidden columns:
If we have already created the table without START_DATE and END_DATE columns as shown in the below example:

```sql
CREATE TABLE STUDENT_COURSE
(
    ID NUMBER(10) PRIMARY KEY,
    STUDENT_ID  NUMBER(10) REFERENCES STUDENTS(STUDENT_ID),
    COURSE_ID NUMBER(10) REFERENCES COURSE(COURSE_ID)
);
```

Then we can add the Valid_Time temporal dimension to the existing table by using the following ALTER TABLE command.

```sql
ALTER TABLE student_course ADD PERIOD FOR student_course_period;
```

The following statement adds two hidden columns to the table STUDENT_COURSE: USER_VALID_TIME_START and USER_VALID_TIME_END.

```sql
ALTER TABLE student_course ADD PERIOD FOR student_course_period;
```

**Querying A Temporal Table With Valid_Time Support**

Temporal validity allows you to use the AS of PERIOD FOR clause to define valid time periods on a table using start and end DATE or TIMESTAMP columns. In the similar fashion VERSIONS PERIOD FOR BETWEEN clause can be used to find out the records which are valid between two given time intervals. These valid time periods can be used in queries against the table [13].

**A. Query based on AS of PERIOD FOR temporal query clause**

Before querying the table insert the following data in the existing tables:

```sql
INSERT INTO students VALUES (1, 'RAM');
INSERT INTO students VALUES (2, 'SUMIT');
INSERT INTO students VALUES (3, 'YESHA');
INSERT INTO students VALUES (4, 'GAURAV');
COMMIT;
INSERT INTO course VALUES (1, 'DATABASE');
INSERT INTO course VALUES (2, 'OPEN SOURCE');
INSERT INTO course VALUES (3, 'PROGRAMMING');
COMMIT;
INSERT INTO student_course VALUES (1, 1, 1,'01-JAN-12','10-FEB-12');
INSERT INTO student_course VALUES (2, 1, 2,'01-FEB-12','15-MAR-12');
```
INSERT INTO student_course VALUES
(3, 1, ’01-JAN-12’, ’01-APR-12’);
INSERT INTO student_course VALUES
(4, 2, ’01-JAN-12’, ’10-FEB-12’);
INSERT INTO student_course VALUES
(5, 2, ’01-FEB-12’, ’15-MAR-12’);
INSERT INTO student_course VALUES
(6, 2, ’01-JAN-12’, ’01-APR-12’);
INSERT INTO student_course VALUES
(7, 3, ’01-JAN-13’, ’10-FEB-13’);
INSERT INTO student_course VALUES
(8, 3, ’01-FEB-13’, ’15-MAR-13’);
INSERT INTO student_course VALUES
(9, 3, ’01-JAN-13’, ’01-APR-13’);
INSERT INTO student_course VALUES
(10, 4, ’01-JAN-14’, ’10-FEB-14’);
INSERT INTO student_course VALUES
(11, 4, ’01-FEB-14’, ’15-MAR-14’);
INSERT INTO student_course VALUES
(12, 4, ’01-JAN-14’, NULL);
COMMIT;

To find out which students were active in a course on a specific date. We could write the following query using AS of PERID FOR temporal clause [14]:

```
SELECT student_course.start_date,
student_course.end_date,
students.student_name,
course.course_name
FROM   student_course
JOIN students ON student_course.student_id = students.student_id
JOIN course ON student_course.course_id = course.course_id
ORDER BY 1, 2, 3;
```

As a result the above query outputs the records of the students whose course is active on a specific date i.e. 12-FEB-2013.

**OUTPUT:**

<table>
<thead>
<tr>
<th>START_DATE</th>
<th>END_DATE</th>
<th>STUDENT_NAME</th>
<th>COURSE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-JAN-13</td>
<td>01-APR-13</td>
<td>YESHA</td>
<td>PROGRAMMING</td>
</tr>
<tr>
<td>01-FEB-13</td>
<td>15-MAR-13</td>
<td>YESHA</td>
<td>OPEN SOURCE</td>
</tr>
</tbody>
</table>

2 rows selected.

In the same way, students who are currently on active course are displayed by using SYSDATE which shown in below example.

```
SELECT student_course.start_date,
student_course.end_date,
students.student_name,
course.course_name
FROM   student_course
AS OF PERIOD FOR student_course_period SYSDATE
JOIN students ON student_course.student_id = students.student_id
JOIN course ON student_course.course_id = course.course_id
ORDER BY 1, 2, 3;
```

```
01-JAN-13 01-APR-13 YESHA PROGRAMMING
01-FEB-13 15-MAR-13 YESHA OPEN SOURCE
```

2 rows selected.
B. Query using VERSIONS PERIOD FOR.....BETWEEN

To find students that were on active courses during a specified time period one can use the VERSIONS PERIOD FOR ... BETWEEN temporal clause[14].

`SELECT student_course.start_date, student_course.end_date, students.student_name, course.course_name
FROM student_course
VERSIONS PERIOD FOR student_course_period BETWEEN
TO_DATE('12-FEB-2013','DD-MON-YYYY') AND TO_DATE('06-JAN-2014','DD-MON-YYYY')
JOIN students ON student_course.student_id = students.student_id
JOIN course ON student_course.course_id = course.course_id
ORDER BY 1, 2, 3;

The above query outputs the records of the students whose course is active on a given specific time period i.e. between 12-FEB-2013 and 06-JAN-2014. The outcome of the above query is highlighted below.

**OUTPUT:**

<table>
<thead>
<tr>
<th>START_DATE</th>
<th>END_DATE</th>
<th>STUDENT_NAME</th>
<th>COURSE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-JAN-14</td>
<td></td>
<td>GAURAV</td>
<td>PROGRAMMING</td>
</tr>
<tr>
<td>01-FEB-13</td>
<td>15-MAR-13</td>
<td>YESHA</td>
<td>OPEN SOURCE</td>
</tr>
<tr>
<td>01-JAN-14</td>
<td>10-FEB-14</td>
<td>GAURAV</td>
<td>DATABASE</td>
</tr>
<tr>
<td>01-JAN-14</td>
<td></td>
<td>GAURAV</td>
<td>PROGRAMMING</td>
</tr>
</tbody>
</table>

4 rows selected.

**Resetting The Environment**

Perform the following steps to reset your environment.

Drop the three tables namely STUDENTS, COURSE and STUDENT_COURSE table by issuing the following commands.

DROP TABLE STUDENT_COURSE purge;
DROP TABLE STUDENTS purge;
DROP TABLE COURSE purge;

**Summary**

In the beginning the work focuses on formal introduction of temporal database and valid time temporal support in temporal database system. The work also highlights various temporal features included in the Oracle 12c. The above work primarily highlights the procedures and mechanism for creating a table with valid-time dimension and querying it using AS OF PERIOD FOR and VERSIONS PERIOD FOR.... BETWEEN valid time temporal dimensions.

**Conclusion**

Temporal Database allows us to embed time element with the data. Temporal Validity support is essential for any real-time database management system application. Temporal Validity feature of Temporal Database adds a time dimension to each row in table consisting of two date-time columns which shows the validity of data on given date-time. When data is bulky, by processing only active records instead of querying entire table can improve the overall performance. This temporal validity can be controlled by the user or application by using AS OF PERIOD FOR and VERSIONS PERIOD FOR.... BETWEEN temporal validity clauses available in Oracle 12c.

**References**


