DDGS Temporal Continuity

Table of Contents

Introduction

In the DDGS Generated History/Audit presentation (5th in the series of 5 presentations), temporal continuity was not addressed because it was too complex. This paper will attempt to define temporal continuity within the framework of DDGS Generated History/Audit. The simplifications that were made and the integrity enforcement inherent to the DDGS deliverables is included.

Background

The basic idea of the temporal and bi-temporal database is centered on time tracking, or specifically, the time that a transaction (or data change) occurred. Since a lag of time can occur between the time an event actually happened and the time it was recorded, 2 distinct time datum are defined:

1. Transaction Time - The time that the transaction (change, event, etc…) was recorded.
2. Valid Time - The time that the transaction (change, event, etc…) actually occurred (or was valid).

A temporal database is used to track one of these 2 time datum. A bi-temporal database tracks both of these time datum together. If a bi-temporal database is used to record events at the same instant they actually occur, then the Transaction Time will always be the same as the Valid Time.

Simplified Implementation

Temporal and bi-temporal database implementations can be terribly complex. This complexity occurs when the permutations of attribute update, unique key constraint, foreign key constraint, past/future Valid Time, and the mixing of temporal, bi-temporal, and non-temporal data are combined into one integrated system. The DDGS deliverables make use of some simplification of the temporal and bi-temporal database to reduce this complexity. This simplification includes

1. No Future Valid Time
2. Valid Time in Bi-Temporal Only
3. New Instances for Temporal Gaps
4. Native Oracle SQL and PL/SQL
5. Separation of Current/Historical Data

No Future Valid Time

The first concept of simplification is that no future Valid Time will be allowed for a transaction. A Valid Time that has yet to occur represents a plan or forecast for something. If a plan changes or a forecast is missed, the Valid Time may not get corrected, leading to unidentifiable errors in the database.

In practice, it is more accurate to record plans and forecasts as attributes instead of Valid Time. This approach allows plans and forecasts to be easily recognized. It also allows tracking of the changes made to plans and forecasts. For instance, if next week's work schedule is posted on Monday, and changes are made on Wednesday, the fact that the schedule changed can be captured with the temporal database. Furthermore, employee pay would not be based on a posted schedule. Rather, pay would be based on timecard data, which tracks what actually happened, not the plan or forecast.

Separation of Current/Historical Data

The second concept of simplification is to separate current/active data from historical/logged data. In this way, most transaction processing can occur against the current data without regard to history. Additionally, performance enhancing structures, like indexes, are smaller and faster when operating only on the current/active data. Finally, the separation of current/historical data is easier conceptually because all things past are in one place, while all things current are in another.

Valid Time in Bi-Temporal Only

The third concept of simplification is to limit the number of temporal and bi-temporal options for each table. This section also includes a discussion of the ASOF view that is used to query a past point-in-time.

NON Mode - non-temporal

A discussion of the NON mode is included to introduce the concept that not all tables are required to have Transaction Time and/or Valid Time. As shown later, these tables do participate in the temporal database if they are foreign keys from a table that has Transaction Time and/or Valid Time. Additionally, there is no ASOF View for these tables. The data that is in these tables is considered immutable and always current.

LOG Mode - temporal (Transaction Time only)

Among other attributes, a table that is configured in LOG Mode includes an AUD_BEG_DTM attribute. This is the starting Transaction Time for the data in the table. All records in this table are considered to be current/active.

These LOG Mode tables will always be paired with another table of the same name that includes an "_LOG" suffix in the name. This _LOG table collects the history of the table that is in LOG Mode and includes both the starting Transaction Time (AUD_BEG_DTM) and the ending Transaction Time (AUD_END_DTM). That is, the data that preceded any update or delete performed on the LOG Table will be captured with the correct AUD_END_DTM in this table. In essence, this table forms an audit log of all records that were in the database before being updated or deleted, along with their starting and ending Transaction Times.

In addition to the "_LOG" table, these LOG Mode tables will always be paired with a view of the same name that includes an "_ASOF" suffix in the name. This ASOF View is will return all records from both the LOG Mode table and the corresponding "_LOG" table that were current/active during a selected Transaction Time (past point-in-time). In this way, the ASOF view can be used to reconstruct the current/active data as it appeared at a given time in the past.

EFF Mode - bi-temporal

A table that is configured in EFF Mode includes the AUD_BEG_DTM attribute, exactly the same as a LOG Mode table. Additionally, and EFF Mode table includes an EFF_BEG_DTM attribute to record the Valid Time. Similar to the LOG Mode tables, the EFF Mode tables will always be paired with another table of the same name that includes "_EFF" suffix in the name. This _EFF table includes AUD_BEG_DTM, AUD_END_DTM, EFF_BEG_DTM, and EFF_END_DTM to capture start Transaction Time and Valid Time as well as end Transaction Time and Valid Time, respectively.

In the same way, a view with the same name that includes an "_ASOF" suffix will be paired with all EFF Mode tables. However, the ASOF point-in-time for EFF Mode tables will be taken from Valid Time, not Transaction Time. Transaction Time is ignored in these ASOF views.

New Instances for Temporal Gaps

The fourth concept of simplification is not as obvious in that it is centered on the idea of a temporal instance of something. A temporal instance is a time in which a something is current/active. For instance, if the same person is hired by a company, leaves the company, then returns, that person is then in their second instance of employment with the company. The first instance of employment has begin and end times and may include modifications like salary changes and promotions. The second instance of employment is different from the first instance in that a temporal gap occurred between the 2 instances. Continuing with the employment example, the second employment instance can include the same employee ID (natural key) for that employee. However, the surrogate primary key must be different for the second instance. In this way, no temporal gaps will exist for any instance of something.

The lack of temporal gaps allow all AUD_END_DTM attributes to have zero or one matching AUD_BEG_DTM. Conversely, all EFF_END_DTM attributes will have zero or one matching AUD_BEG_DTM. The lack of temporal gaps also allows the earliest *_BEG_DTM to be matched with the latest _END_DTM for the same Surrogate Primary Key without requiring a check for temporal continuity between those 2 times. This greatly eases the complexity of reporting and becomes more intuitive when identifying temporal gaps.

Native Oracle SQL and PL/SQL

The final concept of simplification is that these temporal and bi-temporal databases run natively in the Oracle database. This rules out the use of additional data cartridges like Spatial Data Option or calling external libraries. Instead, native tables, views, triggers, and procedures are implemented to allow native SQL to accomplish most every aspect of DML. The only exception is the setting of Valid Time during a delete, which requires a PL/SQL procedure call.

Temporal Integrity

Integrity constraints help keep data accurate and correct. These constraints are an expected part of the Oracle database. In this section, referential and unique integrity constraints are explored in the temporal domain.

Temporal Referential Integrity - Foreign Keys

One of the core constraints in any Oracle schema is the Foreign Key constraint. This constraint ensures that a referenced Foreign Key is available for all records. Within the temporal database, this constraint must be enforced across the temporal domain. The DDGS deliverables include this enforcement across all required permutations of table configurations:

- An EFF Mode table referencing another EFF Mode table
- An EFF Mode table referencing a LOG Mode table
- An EFF Mode table referencing a NON Mode table
- An EFF Mode table referencing itself.
- A LOG Mode table referencing another LOG Mode table
- A LOG Mode table referencing an EFF Mode table
- A LOG Mode table referencing a NON Mode table

- A LOG Mode table referencing itself

For instance, in demonstration 5 "DDGS Generated History/Audit", the DEPT table was configured for EFF Mode and the INV table was configured for LOG Mode. In the example below, the SALES department is removed. Since there are 513 BROCHURES in inventory in that department, the department cannot be removed before the inventory is removed. Since the inventory is in LOG Mode, only Transaction Time is recorded for the inventory delete. Now, suppose the department was removed with a Valid Time of 2 days ago. Let's see what happens.

```
C:\> sqlplus p5_usr/p5_usr
SQL*Plus: Release 11.2.0.2.0 Production on Sat Nov 9 19:41:36 2013
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production

SQL> select sysdate from dual;
SYSDATE
---------
09-NOV-13

SQL> select id, dname, loc from dept_act
  2>  where dname = 'SALES';
        ID DNAME          LOC
---------- -------------- -------------
         3 SALES          CHICAGO

SQL> select id, dept_id, dept_nk1, iname, onhand from inv_act
  2>  where dept_nk1 = 'SALES';
        ID    DEPT_ID DEPT_NK1       INAME                ONHAND
---------- ---------- -------------- ---------------- ----------
         1          3 SALES          BROCHURES               513

SQL> execute util.set_usr('TC Test');
PL/SQL procedure successfully completed.

SQL> delete from inv_act where dept_nk1 = 'SALES';
1 row deleted.

SQL> begin
  2     dept_dml.del (eff_end_dtm_in => sysdate - 2,
  3                   dname_in => 'SALES');
  4  end;
  5  /
begin
*
ERROR at line 1:
ORA-20018: inv_LOG.ID 3 child record exists at 2013-11-09 19:47:19
```

```
ORA-06512: at "P5.DEPT_TAB", line 641
ORA-06512: at "P5.DEPT_DML", line 338
ORA-06512: at line 2
```

Without the temporal foreign key constraint, this delete would have succeeded. However, the DDGS deliverables includes a constraint that found the temporal continuity problem with the Valid Time of the department being deleted 2 days before the Transaction Time of the inventory delete. The error message thrown by the constraint includes a Transaction Time for the deleted inventory record that would cause the temporal continuity constraint violation.

Temporal Uniqueness - Unique Keys

Another core constraint in any Oracle schema is the Unique Key Constraint. This constraint ensures that there is only 1 unique value or set of values for all records. Within the temporal database, this constraint must be enforced across the temporal domain.

Consider the situation where a unique key (like a department name) is re-used over time. If historical effective date/time (Valid Time) is used in a bi-termporal database, this unique key must remain unique across the temporal continuum.

```
C:\> sqlplus p5_usr/p5_usr
SQL*Plus: Release 11.2.0.2.0 Production on Tue Nov 12 11:40:15 2013
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production

SQL> execute util.set_usr('TC Test');
PL/SQL procedure successfully completed.

SQL> select sysdate from dual;
SYSDATE
---------
12-NOV-13

SQL> describe dept
 Name              Null?    Type
 ----------------- -------- --------------------------------
 ID                NOT NULL NUMBER(38)
 EFF_BEG_DTM       NOT NULL TIMESTAMP(9) WITH LOCAL TIME ZONE
 AUD_BEG_USR       NOT NULL VARCHAR2(30)
 AUD_BEG_DTM       NOT NULL TIMESTAMP(9) WITH LOCAL TIME ZONE
 DNAME             NOT NULL VARCHAR2(14)
 LOC               NOT NULL VARCHAR2(13)

SQL> select dname, eff_beg_dtm from dept;
DNAME          EFF_BEG_DTM
-------------- -------------------------
ACCOUNTING     31-DEC-79 11.00.00.000 PM
```

```
OPERATIONS     31-DEC-79 11.00.00.000 PM
RESEARCH       31-DEC-79 11.00.00.000 PM
SALES          31-DEC-79 11.00.00.000 PM

SQL> insert into dept (eff_beg_dtm, dname, loc)
  2  values (sysdate-3, 'HR', 'LOS VEGAS');
1 row created.

SQL> begin
  2     dept_dml.del (eff_end_dtm_in => sysdate - 1,
  3                   dname_in => 'HR');
  4  end;
  5  /
PL/SQL procedure successfully completed.

SQL> select dname, eff_beg_dtm from dept;
DNAME          EFF_BEG_DTM
-------------- -------------------------
ACCOUNTING     31-DEC-79 11.00.00.000 PM
OPERATIONS     31-DEC-79 11.00.00.000 PM
RESEARCH       31-DEC-79 11.00.00.000 PM
SALES          31-DEC-79 11.00.00.000 PM

SQL> select dname, eff_beg_dtm, eff_end_dtm from dept_all;
DNAME          EFF_BEG_DTM                EFF_END_DTM
-------------- -------------------------- --------------------------
ACCOUNTING     31-DEC-79 11.00.00.000 PM 31-DEC-99 05.59.59.000 PM
OPERATIONS     31-DEC-79 11.00.00.000 PM 31-DEC-99 05.59.59.000 PM
RESEARCH       31-DEC-79 11.00.00.000 PM 31-DEC-99 05.59.59.000 PM
SALES          31-DEC-79 11.00.00.000 PM 31-DEC-99 05.59.59.000 PM
HR             09-NOV-13 12.23.13.000 PM 11-NOV-13 12.27.53.000 PM

SQL> insert into dept (eff_beg_dtm, dname, loc)
  2  values (sysdate-2, 'HR', 'ANYWHERE');
insert into dept (eff_beg_dtm, dname, loc)
            *
ERROR at line 1:
ORA-20016: Duplicate dept NK Data found at 2013-11-11 12:27:53 for
ID: 23
ORA-06512: at "P5.DEPT_TAB", line 187
ORA-06512: at "P5.DEPT_TAB", line 320
ORA-06512: at "P5.DEPT_BI", line 47
ORA-04088: error during execution of trigger 'P5.DEPT_BI'
```

Without the temporal unique key constraint, this insert would have succeeded. However, the DDGS deliverables includes a constraint that found the temporal continuity problem with the Valid Time of the same department being inserted 1 day before the Valid Time of the previous delete. The error message thrown by the constraint includes a Valid Time for the deleted

department record that would cause the temporal continuity constraint violation.

Summary

The ability to track Transaction Time and Valid Time is a straight-forward implementation of the temporal and bi-temporal database. However, the ability to enforce temporal continuity of Foreign Key and Unique Key constraints is an important key to maintaining temporal continuity within that database.