DTGen ASOF Demonstration

Developed by DMSTEX (http://dmstex.com)

Table of Contents

Introduction:

The exercises in this demonstration are focused on the history and audit functionality of DTGen functionality. All functionality in these exercises is available through both command line and graphical user interface (GUI) mode. For simplicity in understanding the under-lying workings of DTGen, these exercises are conducted entirely in command-line mode.

The "basic" demonstration should be reviwed before running these exercises. Serveral concepts introduced in those exercises not explained here. Exercise #1 in this demonstration is similar to Exercise #1 in the basic demonstration.

The exercises in this directory are numbered and must be executed in sequential order. The demonstration users must be created with the "create_demo_users.sql" script in the parent directory before the first exercise is run. The demonstration users must be dropped with the "drop_demo_users.sql" script before the "create_demo_users.sql" script can be re-run. These exercises also assume that the default username/password (dtgen/dtgen) is still in use for the generator. Names and passwords are set in the "vars.sql" script and can be modified, if necessary. Also, the DTGen database objects must be installed in the database and the DTGen must be ready to generate code.

Exercise #1: Entity Based History and Audit

Command Line:

```
    sqlplus /nolog @e1
```

Exercise #1 modifies the database. The "drop_demo_users.sql" and "create_demo_users.sql" scripts must be used to reset the database before re-running this exercise.

Based on the demobld.sql script, this exercise implements the EMP and DEPT tables using DTGen. The script for this exercise performs the following functions:

1. Removes any old DEMO2 Items from DTGEN
2. Creates new DEMO2 Items in DTGEN
3. Generates the DEMO2 Application in DTGEN
4. Creates the "install_db.sql" script
5. Runs the "install_db.sql" script
6. Loads and Reports Data

Steps 1-3 are captured in the "e1.LST" file. Following is a example of e1.LST.

```
Login to dtgen
Connected.
Remove old DEMO2 Schema from DTGEN
create a DEMO2 Schema in DTGEN
Generate Demo2 Application
Capture install_db.sql Script
```

Step 4 is captured in the "install_db.sql" file. This file is about 79 kbytes and has over 3,000 lines. Due to its size, it is not listed here. It contains all the code generated by DTGen for this application.

Steps 5 and 6 are captured in the "install.LST" file. Step 5 is the execution of the install_db.sql script.

```
Login to dtgen_db_demo
Connected.

FILE_NAME
----------------
 -) create_glob

FILE_NAME
---------------
 -) create_ods

TABLE_NAME
--------------
***  dept  ***

TABLE_NAME
-------------
***  emp  ***

FILE_NAME
------------------
 -) create_integ

TABLE_NAME
--------------
***  dept  ***

TABLE_NAME
-------------
***  emp  ***

FILE_NAME
----------------
 -) create_oltp

TABLE_NAME
--------------
***  dept  ***
```

```
TABLE_NAME
-------------
***  emp  ***

FILE_NAME
-----------------
 -) create_mods
```

The above listing represents a successful installation of the application generated by DTGen. This application is small in that it only has 2 tables, 1 tier (the database tier), and no user schema.

The DEPT table is silently loaded with data. A query of column comments on the DEPT table from the data dictionary help identify what each column's data represents. Following the column comments is a report of all the data in the DEPT table (active view) for the selected columns.

```
COLUMN_NAME         COMMENTS
------------------- ------------------------------------------------------------
DEPTNO              Department Number
DNAME               Name of the Department
LOC                 Location for the Department
AUD_BEG_USR         User that created this record
AUD_BEG_DTM         Date/Time this record was created (must be in nanoseconds)


DEPTNO DNAME      LOC        AUD_BEG_USR AUD_BEG_DTM
------ ---------- ---------- ----------- ------------
    10 ACCOUNTING NEW YORK   Dataload    01-NOV-80 12
    20 RESEARCH   DALLAS     Dataload    01-NOV-80 12
    30 SALES      CHICAGO    THOMPSON    17-AUG-81 12
    40 OPERATIONS BOSTON     JAMES       12-FEB-82 12
```

The EMP table is also silently loaded with data. The same queries of column comments and data on the EMP table (active view) are shown.

```
COLUMN_NAME         COMMENTS
------------------- ------------------------------------------------------------
EMPNO               Employee Number
ENAME               Employee Name
JOB                 Job Title
MGR_EMP_NK1         EMP Natural Key Value 1: Employee Number
HIREDATE            Date the Employee was hired
SAL                 Employee's Salary
DEPT_NK1            DEPT Natural Key Value 1: Department Number
EFF_BEG_DTM         Date/Time this record became effective
AUD_BEG_USR         User that created this record


EMPNO ENAME  JOB        MGR_  HIREDATE    SAL  DEPT_  AUD_BEG_USR AUD_BEG_DTM
----- ------ --------- ----- --------- ----- ------ ----------- ------------
 7369 SMITH  CLERK      7902 17-DEC-80   800     20 SMITH       27-FEB-83 12
 7499 ALLEN  SALESMAN   7698 20-FEB-81  1600     30 THOMPSON    15-MAY-81 12
 7521 WARD   SALESMAN   7698 22-FEB-81  1250     30 THOMPSON    16-MAY-81 12
 7566 JONES  MANAGER    7839 02-APR-81  2975     20 SMITH       01-DEC-81 12
 7654 MARTIN SALESMAN   7698 28-SEP-81  1250     30 SMITH       28-SEP-81 12
 7698 BLAKE  MANAGER    7839 01-MAY-81  2850     30 SMITH       27-NOV-81 12
 7782 CLARK  MANAGER    7839 09-JUN-81  2450     10 SMITH       26-NOV-81 12
 7788 SCOTT  ANALYST    7566 09-DEC-82  3000     20 JAMES       07-DEC-82 12
 7839 KING   PRESIDENT       17-NOV-81  5000     10 SMITH       19-NOV-81 12
 7844 TURNER SALESMAN   7698 08-SEP-81  1500     30 SMITH       09-SEP-81 12
 7876 ADAMS  CLERK      7788 12-JAN-83  1100     20 SMITH       12-JAN-83 12
 7900 JAMES  CLERK      7698 03-DEC-81   950     30 SMITH       04-DEC-81 12
 7902 FORD   ANALYST    7566 03-DEC-81  3000     20 SMITH       05-DEC-81 12
 7934 MILLER CLERK      7782 23-JAN-82  1300     10 JAMES       23-JAN-82 12
```

In addition to the DEPT and EMP tables, this exercise also loaded a DEPT audit table called DEPT_AUD and an EMP history table called EMP_HIST.

```
COLUMN_NAME         COMMENTS
------------------- ------------------------------------------------------------
DEPT_ID             Surrogate Primary Key from the ACTIVE table
DEPTNO              Department Number
DNAME               Name of the Department
LOC                 Location for the Department
AUD_BEG_USR         User that created this record
AUD_BEG_DTM         Date/Time this record was created (must be in nanoseconds)
AUD_END_USR         User that modified/deleted this record
AUD_END_DTM         Date/Time this record was modified/deleted (must be in
                    nanoseconds)


DEPT_ID DEPTNO DNAME      LOC      AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
------- ------ ---------- -------- -------- --------- -------- ---------
      3     20 SALES      ST LOUIS Dataload 01-NOV-80 THOMPSON 17-AUG-82
      4     40 OPERATIONS BUFFALO  Dataload 01-NOV-80 JAMES    12-FEB-82
```

In the DEPT_AUD listing above, the column comments for the columns starting with "AUD_" indicate the user and date/time of modifications are being tracked, even after a record has been updated or deleted. The data records show that the SALES deparment was originally in St. Louis and the OERATIONS department was originally in Buffalo.

```
COLUMN_NAME         COMMENTS
------------------- ------------------------------------------------------------
EMP_ID              Surrogate Primary Key from the ACTIVE table
EMPNO               Employee Number
ENAME               Employee Name
JOB                 Job Title
MGR_EMP_ID          Surrogate Key of Employee's Manager
AUD_BEG_USR         User that created this record
AUD_BEG_DTM         Date/Time this record was created (must be in nanoseconds)
AUD_END_USR         User that modified/deleted this record
AUD_END_DTM         Date/Time this record was modified/deleted (must be in
                    nanoseconds)


EMP_ID EMPNO ENAME    JOB        MGR_ AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
------ ----- -------- --------- ----- -------- --------- -------- ---------
     1  7301 ELLISON  PRESIDENT       DAVIS    31-OCT-80 THOMPSON 28-JUN-81
     2  7344 DAVIS    CLERK         1 DAVIS    15-NOV-80 THOMPSON 21-JUN-81
     2  7344 DAVIS    CLERK        11 THOMPSON 22-JUN-81 THOMPSON 20-AUG-81
     2  7344 DAVIS    CLERK        12 SMITH    21-AUG-81 SMITH    30-NOV-81
     2  7344 DAVIS    CLERK        15 SMITH    28-NOV-81 SMITH    10-DEC-81
     3  7369 THOMPSON CLERK         1 DAVIS    18-DEC-80 SMITH    25-FEB-83
     3  7369 SMITH    CLERK        15 SMITH    29-NOV-81 SMITH    28-FEB-83
     3  7369 THOMPSON CLERK        11 THOMPSON 22-JUN-81 SMITH    22-AUG-81
     3  7369 SMITH    CLERK        12 SMITH    23-AUG-81 SMITH    30-NOV-81
     4  7499 ALLEN    SALESMAN      1 THOMPSON 18-FEB-81 THOMPSON 12-MAY-81
     5  7521 WARD     SALESMAN      1 THOMPSON 24-FEB-81 THOMPSON 16-MAY-81
     6  7566 JONES    MANAGER       1 THOMPSON 31-MAR-81 THOMPSON 21-JUN-81
     6  7566 JONES    MANAGER      11 THOMPSON 23-JUN-81 THOMPSON 19-AUG-81
     6  7566 JONES    MANAGER      12 THOMPSON 20-AUG-81 SMITH    29-NOV-81
     7  7654 MARTIN   SALESMAN      1 THOMPSON 17-APR-81 THOMPSON 14-MAY-81
     8  7698 BLAKE    MANAGER       1 THOMPSON 30-APR-81 THOMPSON 24-JUN-81
     8  7698 BLAKE    MANAGER      11 THOMPSON 21-JUN-81 SMITH    23-AUG-81
     8  7698 BLAKE    MANAGER      12 SMITH    22-AUG-81 SMITH    29-NOV-81
     9  7782 CLARK    MANAGER       1 THOMPSON 09-JUN-81 THOMPSON 24-JUN-81
     9  7782 CLARK    MANAGER      11 THOMPSON 22-JUN-81 SMITH    22-AUG-81
     9  7782 CLARK    MANAGER      12 SMITH    21-AUG-81 SMITH    27-NOV-81
    10  7788 SCOTT    ANALYST       6 THOMPSON 09-JUN-81 JAMES    09-MAR-82
    11  7839 KING     PRESIDENT       THOMPSON 18-JUN-81 SMITH    30-AUG-81
    12  7840 LANE     PRESIDENT       THOMPSON 12-AUG-81 SMITH    28-NOV-81
    16  7876 ADAMS    CLERK         6 SMITH    22-NOV-81 JAMES    14-JUN-82
```

In the EMP_HIST listing above, the column comments and data for selected columns are queried. The column comments for the columns show that both "_AUD" and "_HIST" tables include the audit columns that start with "AUD_". The data records show that many changes have occured to the data in the EMP table.

Exercise #2: EFF vs. LOG Table Types

Command Line:

```
sqlplus /nolog @e2
```

Exercise #2 does not modify the database. This exercise can be repeated without problem.

Exercise #1 included a brief introduction to the DEPT_AUD and EMP_HIST tables. Below are the DTGen settings that generated those tables.

```
Login to dtgen
Connected.

VALUE DESCRIPTION
----- ----------------------------------------------------------------------
EFF   A historical table type with effective/audit begin/end timestamps and
      begin/end user recording
LOG   An audit table type with audit only begin/end timestamps and begin/end
      user recording
NON   A none or nothing table type without begin/end timestamps or begin/end
      user recording


       SEQ NAME            TYP
---------- --------------- ---
        10 dept            LOG
        20 emp             EFF
```

For any table configured in DTGen, one of these 3 table types must be selected. In the "basics" demonstration, "NON" was set as the table type for both tables. This demonstration shows the use of LOG and EFF table types. An example of when to use LOG versus EFF is also represented with these tables. The DEPT table holds department information, which is slow moving (doesn't change often or rapidly) and is generally known well in advance of any changes to the applicaiton data (i.e. Adding a new department). Constrast the employee information, which can be fast moving and needs to be recorded as occuring at a specific time (effectivity) in addition to simple audit recording.

```
SQL>
SQL> select id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm
  3    from dept_act where deptno = 50;

no rows selected

SQL> select dept_id id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from dept_aud where deptno = 50;

no rows selected
```

There are no records for a department with a DEPTNO of "50".

```
SQL>
SQL> execute util.set_usr('USER1');

PL/SQL procedure successfully completed.

SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 04.35.54
```

```
1 row selected.

SQL> insert into dept_act (deptno, dname, loc)
  2    values (50, 'NEW_DEPT', 'LZ');

1 row created.

SQL>
SQL> select id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm
  3    from dept_act where deptno = 50;

 ID DEPT LOC AUD_BEG_USR AUD_BEG_DTM
--- ---- --- ----------- ------------------
  5   50 LZ  USER1       17-APR-12 04.35.54

1 row selected.

SQL> select dept_id id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from dept_aud where deptno = 50;

no rows selected
```

After setting the util.set_usr, the time is reported and a DEPT record is created. The last 2 queries above show that a DEPT_ACT record was created by the util.set_usr value at the current time, and here are no DEPT_AUD records.

```
SQL>
SQL> execute dbms_lock.sleep(1);

PL/SQL procedure successfully completed.

SQL> execute util.set_usr('USER2');

PL/SQL procedure successfully completed.

SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 04.35.55

1 row selected.

SQL> update dept_act
  2    set  loc = 'LA'
  3   where deptno = 50;

1 row updated.

SQL>
SQL> select id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm
  3    from dept_act where deptno = 50;

 ID DEPT LOC AUD_BEG_USR AUD_BEG_DTM
--- ---- --- ----------- ------------------
  5   50 LA  USER2       17-APR-12 04.35.55

1 row selected.

SQL> select dept_id id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from dept_aud where deptno = 50;

 ID DEPT LOC AUD_BEG_USR AUD_BEG_DTM        AUD_END_USR AUD_END_DTM
--- ---- --- ----------- ------------------ ----------- ------------------
  5   50 LZ  USER1       17-APR-12 04.35.54 USER2       17-APR-12 04.35.55

1 row selected.
```

In the sequence above, the DBMS_LOCK.SLEEP function is used to elapse one second before running the update with a different util.set_usr. The last 2 queries above show one record in each DEPT_ACT and DEPT_AUD. The DEPT_AUD record is the previous DEPT_ACT record. Also, the DEPT_AUD record has a matching AUD_END_DTM to the AUD_BEG_DTM of the new DEPT_ACT record. Notice that USER2 is recorded as the AUD_END_USR in DEPT_AUD and the AUD_BEG_USR in DEPT_ACT.

```
SQL>
SQL> execute dbms_lock.sleep(1);

PL/SQL procedure successfully completed.

SQL> execute util.set_usr('USER3');

PL/SQL procedure successfully completed.

SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 04.35.56

1 row selected.

SQL> delete from dept_act where deptno = 50;

1 row deleted.

SQL>
SQL> select id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm
  3    from dept_act where deptno = 50;

no rows selected

SQL> select dept_id id, deptno dept, loc,
  2    aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from dept_aud where deptno = 50;

 ID DEPT LOC AUD_BEG_USR AUD_BEG_DTM        AUD_END_USR AUD_END_DTM
--- ---- --- ----------- ------------------ ----------- ------------------
  5   50 LZ  USER1       17-APR-12 04.35.54 USER2       17-APR-12 04.35.55
  5   50 LA  USER2       17-APR-12 04.35.55 USER3       17-APR-12 04.35.56

2 rows selected.
```

In the sequence above, a delete is run on DEPT_ACT, resulting in no DEPT_ACT records and 2 DEPT_AUD records. This functionality works the same for the DEPT_DML API calls.

Below, the same set of processes will be repeated for an employee record.

```
SQL>
SQL> select id, empno, ename, to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  2    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm
  3    from emp_act where empno = 9999;

no rows selected

SQL> select emp_id id, empno, ename,
  2    to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  3    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm,
  4    to_char(eff_end_dtm,'DD HH24:MI:SS') eff_end_dtm,
  5    aud_end_usr, to_char(aud_end_dtm,'DD HH24:MI:SS') aud_end_dtm
  6    from emp_hist where empno = 9999;

no rows selected

SQL>
SQL> execute util.set_usr('USER1');

PL/SQL procedure successfully completed.
```

```
SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 08.25.17

1 row selected.

SQL> insert into emp_act (empno, ename, job, hiredate, sal, dept_nk1,
  2      eff_beg_dtm)
  3    values (9999, 'NEW_EMP', 'CLERK', sysdate, 100, 40,
  4      to_timestamp('1983-6-1 11', 'YYYY-MM-DD HH24'));

1 row created.

SQL>
SQL> select id, empno, ename, to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  2    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm
  3    from emp_act where empno = 9999;

 ID EMPNO ENAME     EFF_BEG_DTM AUD_B AUD_BEG_DTM
--- ----- --------- ----------- ----- -----------
 22  9999 NEW_EMP   01 11:00:00 USER1 17 20:25:17

1 row selected.

SQL> select emp_id id, empno, ename,
  2    to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  3    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm,
  4    to_char(eff_end_dtm,'DD HH24:MI:SS') eff_end_dtm,
  5    aud_end_usr, to_char(aud_end_dtm,'DD HH24:MI:SS') aud_end_dtm
  6    from emp_hist where empno = 9999;

no rows selected
```

Notice that the EMP_ACT active view has the EFF_BEG_DTM column that DEPT_ACT does not. This is a key difference between EFF and LOG. This additional effectivity column allows a date/time in the past to be entered and referenced as the effective time an event occured. Otherwise, the LOG table only records when an event was entered, as the AUD_BEG_DTM is acquired from the system clock and cannot be entered.

The first 2 queries show there are no EMP_ACT records with an EMPNO of 9999. After the user is set and the time is queried, a record is inserted into the EMP_ACT active view with an effectivity date/time of June 1st, 1983 at 11am. The record is returned from the EMP_ACT query and no records are returned from EMP_HIST.

```
SQL>
SQL> execute dbms_lock.sleep(1);

PL/SQL procedure successfully completed.

SQL> execute util.set_usr('USER2');

PL/SQL procedure successfully completed.

SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 08.25.18

1 row selected.

SQL> update emp_act
  2    set  ename = 'UPD_EMP',
  3         eff_beg_dtm = to_timestamp('1983-6-2 12', 'YYYY-MM-DD HH24')
  4   where empno = 9999;

1 row updated.

SQL>
SQL> select id, empno, ename, to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  2    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm
  3    from emp_act where empno = 9999;

 ID EMPNO ENAME     EFF_BEG_DTM AUD_B AUD_BEG_DTM
--- ----- --------- ----------- ----- -----------
 22  9999 UPD_EMP   02 12:00:00 USER2 17 20:25:18

1 row selected.

SQL> select emp_id id, empno, ename,
  2    to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  3    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm,
  4    to_char(eff_end_dtm,'DD HH24:MI:SS') eff_end_dtm,
  5    aud_end_usr, to_char(aud_end_dtm,'DD HH24:MI:SS') aud_end_dtm
  6    from emp_hist where empno = 9999;

 ID EMPNO ENAME     EFF_BEG_DTM AUD_B AUD_BEG_DTM EFF_END_DTM AUD_E AUD_END_DTM
--- ----- --------- ----------- ----- ----------- ----------- ----- -----------
 22  9999 NEW_EMP   01 11:00:00 USER1 17 20:25:17 02 12:00:00 USER2 17 20:25:18

1 row selected.
```

Again, the user is set and the time is queried. An update is issued against the active view with an effectivity date/time of June 2nd, 1983 at 12pm. Each of the last 2 queries return 1 record showing the effectivity that was entered. AUD_BEG_DTM and AUD_END_DTM will always have the current date/time as these values are not allowed to be entered.

```
SQL>
SQL> execute dbms_lock.sleep(1);

PL/SQL procedure successfully completed.

SQL> select systimestamp from dual;

SYSTIMESTAMP
------------------
17-APR-12 08.25.19

1 row selected.

SQL> declare
  2     eff_end_dtm  timestamp with local time zone;
  3     emp_id      number;
  4  begin
  5     util.set_usr('USER3');
  6     select id into emp_id
  7       from emp_act where empno = 9999;
  8     eff_end_dtm := to_timestamp('1983-6-3 13', 'YYYY-MM-DD HH24');
  9     emp_dml.del(emp_id, eff_end_dtm);
 10  end;
 11  /

PL/SQL procedure successfully completed.

SQL>
SQL> select id, empno, ename, to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  2    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm
  3    from emp_act where empno = 9999;

no rows selected

SQL> select emp_id id, empno, ename,
  2    to_char(eff_beg_dtm,'DD HH24:MI:SS') eff_beg_dtm,
  3    aud_beg_usr, to_char(aud_beg_dtm,'DD HH24:MI:SS') aud_beg_dtm,
  4    to_char(eff_end_dtm,'DD HH24:MI:SS') eff_end_dtm,
  5    aud_end_usr, to_char(aud_end_dtm,'DD HH24:MI:SS') aud_end_dtm
  6    from emp_hist where empno = 9999;

 ID EMPNO ENAME     EFF_BEG_DTM AUD_B AUD_BEG_DTM EFF_END_DTM AUD_E AUD_END_DTM
--- ----- --------- ----------- ----- ----------- ----------- ----- -----------
 22  9999 NEW_EMP   01 11:00:00 USER1 17 20:25:17 02 12:00:00 USER2 17 20:25:18
 22  9999 UPD_EMP   02 12:00:00 USER2 17 20:25:18 03 13:00:00 USER3 17 20:25:19
```

```
2 rows selected.

SQL>
SQL> rollback;

Rollback complete.
```

Notice the EMP_DML.DEL procedure was used to delete the record using an effectivity date/time of June 3rd, 1983 at 1pm. The DML package API is the only way to enter an EFF_END_DTM during a delete. There is no corresponding functionality available using the EMP_ACT active view. Otherwise, the EMP_DML.INS and EMP_DML_UPD procedures work the same as insert and delete SQL on the EMP_ACT active view.

Exercise #3: Point-in-Time ASOF Views

Command Line:

```
sqlplus /nolog @e3
```

Exercise #3 does not modify the database. This exercise can be repeated without problem.

In this exercise, indexes on foreign keys and natural keys are explored. Following is a query of the DTGen setup used to generate this application

```
SQL>
SQL> execute util.set_asof_dtm(to_timestamp('1981-06-01', 'YYYY-MM-DD'))

PL/SQL procedure successfully completed.

SQL>
SQL> select id, empno, ename, job, mgr_emp_nk1, hiredate,
  2         sal, dept_nk1, aud_beg_usr, aud_beg_dtm
  3   from  emp_asof
  4   order by empno;

 ID EMPNO ENAME    JOB        MGR_ HIREDATE    SAL DEPT AUD_BEG_USR AUD_BEG_DTM
--- ----- -------- --------- ----- --------- ----- ---- ----------- ------------
  1  7301 ELLISON  PRESIDENT       02-NOV-80  4000   10 DAVIS       31-OCT-80 12
  2  7344 DAVIS    CLERK      7301 16-NOV-80  1400   10 DAVIS       15-NOV-80 12
  3  7369 THOMPSON CLERK      7301 17-DEC-80   800   10 DAVIS       18-DEC-80 12
  4  7499 ALLEN    SALESMAN   7698 20-FEB-81  1600   30 THOMPSON    15-MAY-81 12
  5  7521 WARD     SALESMAN   7698 22-FEB-81  1250   30 THOMPSON    16-MAY-81 12
  6  7566 JONES    MANAGER    7301 02-APR-81  2975   20 THOMPSON    31-MAR-81 12
  8  7698 BLAKE    MANAGER    7301 01-MAY-81  2850   30 THOMPSON    30-APR-81 12

7 rows selected.

SQL>
SQL> execute util.set_asof_dtm(to_timestamp('1981-09-01', 'YYYY-MM-DD'))

PL/SQL procedure successfully completed.

SQL>
SQL> select id, empno, ename, job, mgr_emp_nk1, hiredate,
  2         sal, dept_nk1, aud_beg_usr, aud_beg_dtm
  3   from  emp_asof
  4   order by empno;

 ID EMPNO ENAME    JOB        MGR_ HIREDATE    SAL DEPT AUD_BEG_USR AUD_BEG_DTM
--- ----- -------- --------- ----- --------- ----- ---- ----------- ------------
  2  7344 DAVIS    CLERK      7840 16-NOV-80  1400   10 SMITH       21-AUG-81 12
  3  7369 SMITH    CLERK      7840 17-DEC-80   800   10 SMITH       23-AUG-81 12
  3  7369 THOMPSON CLERK           17-DEC-80   800   10 DAVIS       18-DEC-80 12
  4  7499 ALLEN    SALESMAN   7698 20-FEB-81  1600   30 THOMPSON    15-MAY-81 12
  5  7521 WARD     SALESMAN   7698 22-FEB-81  1250   30 THOMPSON    16-MAY-81 12
  6  7566 JONES    MANAGER    7840 02-APR-81  2975   20 THOMPSON    20-AUG-81 12
  8  7698 BLAKE    MANAGER    7840 01-MAY-81  2850   30 SMITH       22-AUG-81 12
  9  7782 CLARK    MANAGER    7840 09-JUN-81  2450   10 SMITH       21-AUG-81 12
 10  7788 SCOTT    ANALYST    7566 12-JUN-81  3000   20 THOMPSON    09-JUN-81 12
 12  7840 LANE     PRESIDENT       14-AUG-81  6000   10 THOMPSON    12-AUG-81 12

10 rows selected.

SQL>
SQL> execute util.set_asof_dtm(to_timestamp('1982-01-01', 'YYYY-MM-DD'))

PL/SQL procedure successfully completed.

SQL>
SQL> select id, empno, ename, job, mgr_emp_nk1, hiredate,
  2         sal, dept_nk1, aud_beg_usr, aud_beg_dtm
  3   from  emp_asof
  4   order by empno;

 ID EMPNO ENAME    JOB        MGR_ HIREDATE    SAL DEPT AUD_BEG_USR AUD_BEG_DTM
--- ----- -------- --------- ----- --------- ----- ---- ----------- ------------
  3  7369 SMITH    CLERK      7839 17-DEC-80   800   10 SMITH       29-NOV-81 12
  3  7369 THOMPSON CLERK           17-DEC-80   800   10 DAVIS       18-DEC-80 12
  4  7499 ALLEN    SALESMAN   7698 20-FEB-81  1600   30 THOMPSON    15-MAY-81 12
  5  7521 WARD     SALESMAN   7698 22-FEB-81  1250   30 THOMPSON    16-MAY-81 12
  6  7566 JONES    MANAGER    7839 02-APR-81  2975   20 SMITH       01-DEC-81 12
 14  7654 MARTIN   SALESMAN   7698 28-SEP-81  1250   30 SMITH       28-SEP-81 12
  8  7698 BLAKE    MANAGER    7839 01-MAY-81  2850   30 SMITH       27-NOV-81 12
  9  7782 CLARK    MANAGER    7839 09-JUN-81  2450   10 SMITH       26-NOV-81 12
 10  7788 SCOTT    ANALYST    7566 12-JUN-81  3000   20 THOMPSON    09-JUN-81 12
 15  7839 KING     PRESIDENT       17-NOV-81  5000   10 SMITH       19-NOV-81 12
 13  7844 TURNER   SALESMAN   7698 08-SEP-81  1500   30 SMITH       09-SEP-81 12

 ID EMPNO ENAME    JOB        MGR_ HIREDATE    SAL DEPT AUD_BEG_USR AUD_BEG_DTM
--- ----- -------- --------- ----- --------- ----- ---- ----------- ------------
 16  7876 ADAMS    CLERK      7566 22-NOV-81  1100   20 SMITH       22-NOV-81 12
 17  7900 JAMES    CLERK      7698 03-DEC-81   950   30 SMITH       04-DEC-81 12
 18  7902 FORD     ANALYST    7566 03-DEC-81  3000   20 SMITH       05-DEC-81 12

14 rows selected.
```

Exercise #4: Audited POP Functions

Command Line:

```
sqlplus /nolog @e4
```

Exercise #4 modifies the database. The "drop_demo_users.sql", "create_demo_users.sql", and "e1.sql" scripts must be used to reset the database before re-running this exercise.

Each table defined in DTGen is generated with a corresponding "active view". The DEPT and EMP tables have an active view called "DEPT_ACT" and "EMP_ACT", respectively. In most cases, these views should be used for all DML (Data

Exercise #5: Comprehensive OMNI Views

Command Line:

```
sqlplus /nolog @e5
```

Exercise #5 does not modify the database. This exercise can be repeated without problem.

The EMP table has a self-referencing foreign key. It is the relationship between employees and managers. Since managers are also employees, they have managers as well, with the exception of the PRESIDENT. This self-referencing foreig

Exercise #6: Transportable ASOF Data

Command Line:

```
sqlplus /nolog @e6
```

Exercise #6 does not modify the database. This exercise can be repeated without problem.

Unlike the original demobld.sql, this demonstration includes built in domain checking on the JOB column in the EMP table. The configuration of DTGen included a domain specification for all possible company jobs. Unlike a foreign key table, a domain is embedded into the error checking

Oracle introduced a mechanism to hold rollback entries for use by "flashback" queries to provide data as it was at some previous point in time. Since the source of this data is the rollback segments, transporting this data has limitations. DTGen provides a schema based approach to retrieving data at some previous point in time. The data in the Dtgen generated schema are more easily transported to different databases and servers than the rollback segment data. The loading of data for this demonstration demonstrates some of the capability of transporting historical data. These exercises would not have been more difficult and abstract using Oracle's rollback segment data.

Additionally, it is impossible to edit historical mistakes in the rollback segments.