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 are 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 161 kbytes and has over 5,000 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
------------------- ----------------------------------------------------------------
ID                  Surrogate Primary Key for this 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)


 ID DEPTNO DNAME      LOC      AUD_BEG_USR AUD_BEG_DTM
--- ------ ---------- -------- ----------- ------------
  1     10 ACCOUNTING NEW YORK Dataload    01-NOV-80 12
  2     20 RESEARCH   DALLAS   Dataload    01-NOV-80 12
  3     30 SALES      CHICAGO  THOMPSON    17-AUG-82 12
  4     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
------------------- ----------------------------------------------------------------
ID                  Surrogate Primary Key for this table
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
AUD_BEG_USR         User that created this record
AUD_BEG_DTM         Date/Time this record was created (must be in nanoseconds)


 ID EMPNO ENAME  JOB        MGR_ HIREDATE    SAL  DEPT_ AUD_BEG_USR AUD_BEG_DTM
--- ----- ------ --------- ----- --------- ----- ------ ----------- ------------
  3  7369 SMITH  CLERK      7902 17-DEC-80   800     20 SMITH       26-FEB-83 12
  4  7499 ALLEN  SALESMAN   7698 20-FEB-81  1600     30 THOMPSON    12-MAY-81 12
  5  7521 WARD   SALESMAN   7698 22-FEB-81  1250     30 THOMPSON    14-MAY-81 12
  6  7566 JONES  MANAGER    7839 02-APR-81  2975     20 SMITH       29-NOV-81 12
 14  7654 MARTIN SALESMAN   7698 28-SEP-81  1250     30 SMITH       26-SEP-81 12
  8  7698 BLAKE  MANAGER    7839 01-MAY-81  2850     30 SMITH       30-NOV-81 12
  9  7782 CLARK  MANAGER    7839 09-JUN-81  2450     10 SMITH       26-NOV-81 12
 20  7788 SCOTT  ANALYST    7566 09-DEC-82  3000     20 JAMES       11-DEC-82 12
 15  7839 KING   PRESIDENT       17-NOV-81  5000     10 SMITH       18-NOV-81 12
 13  7844 TURNER SALESMAN   7698 08-SEP-81  1500     30 SMITH       06-SEP-81 12
 21  7876 ADAMS  CLERK      7788 12-JAN-83  1100     20 SMITH       12-JAN-83 12
 17  7900 JAMES  CLERK      7698 03-DEC-81   950     30 SMITH       05-DEC-81 12
 18  7902 FORD   ANALYST    7566 03-DEC-81  3000     20 SMITH       01-DEC-81 12
 19  7934 MILLER CLERK      7782 23-JAN-82  1300     10 JAMES       21-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
AUD_BEG_USR          User that created this record
AUD_END_USR          User that modified/deleted this record
AUD_BEG_DTM          Date/Time this record was created (must be in nanoseconds)
AUD_END_DTM          Date/Time this record was modified/deleted (must be in
                     nanoseconds)
DEPTNO               Department Number
DNAME                Name of the Department
LOC                  Location for the Department


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
AUD_BEG_USR          User that created this record
AUD_END_USR          User that modified/deleted this record
AUD_BEG_DTM          Date/Time this record was created (must be in nanoseconds)
AUD_END_DTM          Date/Time this record was modified/deleted (must be in
                     nanoseconds)
EMPNO                Employee Number
ENAME                Employee Name
JOB                  Job Title
MGR_EMP_ID           Surrogate Key of Employee's Manager


EMPNO EMP_ID ENAME    JOB        MGR_ AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
----- ------ -------- --------- ----- -------- --------- -------- ---------
 7301      1 ELLISON  PRESIDENT       DAVIS    04-NOV-80 THOMPSON 28-JUN-81
 7344      2 DAVIS    CLERK         1 DAVIS    14-NOV-80 THOMPSON 25-JUN-81
 7344      2 DAVIS    CLERK        11 THOMPSON 25-JUN-81 THOMPSON 20-AUG-81
 7344      2 DAVIS    CLERK        12 THOMPSON 20-AUG-81 SMITH    29-NOV-81
 7344      2 DAVIS    CLERK        15 SMITH    29-NOV-81 SMITH    06-DEC-81
 7369      3 THOMPSON CLERK         1 DAVIS    15-DEC-80 THOMPSON 25-JUN-81
 7369      3 THOMPSON CLERK        11 THOMPSON 25-JUN-81 SMITH    21-AUG-81
 7369      3 SMITH    CLERK        12 SMITH    21-AUG-81 SMITH    01-DEC-81
 7369      3 SMITH    CLERK        15 SMITH    01-DEC-81 SMITH    26-FEB-83
 7499      4 ALLEN    SALESMAN      1 THOMPSON 17-FEB-81 THOMPSON 12-MAY-81
 7521      5 WARD     SALESMAN      1 THOMPSON 24-FEB-81 THOMPSON 14-MAY-81
 7566      6 JONES    MANAGER       1 THOMPSON 03-APR-81 THOMPSON 24-JUN-81
 7566      6 JONES    MANAGER      11 THOMPSON 24-JUN-81 SMITH    22-AUG-81
 7566      6 JONES    MANAGER      12 SMITH    22-AUG-81 SMITH    29-NOV-81
 7654      7 MARTIN   SALESMAN      1 THOMPSON 16-APR-81 THOMPSON 13-MAY-81
 7698      8 BLAKE    MANAGER       1 THOMPSON 02-MAY-81 THOMPSON 24-JUN-81
 7698      8 BLAKE    MANAGER      11 THOMPSON 24-JUN-81 THOMPSON 19-AUG-81
 7698      8 BLAKE    MANAGER      12 THOMPSON 19-AUG-81 SMITH    30-NOV-81
 7782      9 CLARK    MANAGER       1 THOMPSON 07-JUN-81 THOMPSON 23-JUN-81
 7782      9 CLARK    MANAGER      11 THOMPSON 23-JUN-81 SMITH    23-AUG-81
 7782      9 CLARK    MANAGER      12 SMITH    23-AUG-81 SMITH    26-NOV-81
 7788     10 SCOTT    ANALYST       6 THOMPSON 10-JUN-81 JAMES    09-MAR-82
 7839     11 KING     PRESIDENT       THOMPSON 18-JUN-81 SMITH    30-AUG-81
 7840     12 LANE     PRESIDENT       THOMPSON 12-AUG-81 SMITH    29-NOV-81
 7876     16 ADAMS    CLERK         6 SMITH    20-NOV-81 JAMES    13-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_". Each record from EMP_HIST represents a record that was in EMP_ACT from AUD_BEG_DTM to AUD_END_DTM. Notice that EMPNO 7369 changes

names from THOMPSON to SMITH. This name change was entered into EMP_ACT on August 2[1st,] 1981. (i.e. Both AUD_BEG_DTM of the record with the new ENAME and the AUD_END_DTM of the previous record have the same date.)

Exercise #2: EFF vs. LOG Table Types

Command Line:

```
sqlplus /nolog @e2
```

Exercise #2 does not modify the database. (The DEPT_SEQ and EMP_SEQ sequences are incremented by one). 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.

SQL>
SQL> rollback;

Rollback complete.
```

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, we take a trip back in time using the DTGen version of flashback query. The ASOF view on each table returns data that was current during the "util.set_asof_dtm" date/time specified. Referential integrity is not specifically enforced in the AUD and HIST data. However, since referential integrity was enforced at any given time, the data for any point in time should have integrity. (NOTE: Referential integrity is not guaranteed until Issue #2 is resolved. See "http://code.google.com/p/dtgen/issues/detail?id=2 ")

The following query joins the EMP_ASOF and DEPT_ASOF views at a point in time of January 1st, 1983 (midnight).

```
SQL>
SQL> execute glob.set_asof_dtm(to_timestamp('1983-01-01', 'YYYY-MM-DD'))

PL/SQL procedure successfully completed.

SQL>
SQL> select empno, ename, job, mgr_emp_nk1, hiredate, sal, deptno, dname, loc
  2   from  emp_asof e, dept_asof d where e.dept_id = d.id
  3   order by empno;

EMPNO ENAME    JOB       MGR_EMP_NK1 HIREDATE    SAL DEPTNO DNAME      LOC
----- -------- --------- ----------- --------- ----- ------ ---------- --------
 7369 SMITH    CLERK            7839 17-DEC-80   800     10 ACCOUNTING NEW YORK
 7499 ALLEN    SALESMAN         7698 20-FEB-81  1600     30 SALES      CHICAGO
 7521 WARD     SALESMAN         7698 22-FEB-81  1250     30 SALES      CHICAGO
 7566 JONES    MANAGER          7839 02-APR-81  2975     20 RESEARCH   DALLAS
 7654 MARTIN   SALESMAN         7698 28-SEP-81  1250     30 SALES      CHICAGO
```

```
 7698 BLAKE    MANAGER          7839 01-MAY-81  2850     30 SALES      CHICAGO
 7782 CLARK    MANAGER          7839 09-JUN-81  2450     10 ACCOUNTING NEW YORK
 7788 SCOTT    ANALYST          7566 09-DEC-82  3000     20 RESEARCH   DALLAS
 7839 KING     PRESIDENT             17-NOV-81  5000     10 ACCOUNTING NEW YORK
 7844 TURNER   SALESMAN         7698 08-SEP-81  1500     30 SALES      CHICAGO
 7900 JAMES    CLERK            7698 03-DEC-81   950     30 SALES      CHICAGO
 7902 FORD     ANALYST          7566 03-DEC-81  3000     20 RESEARCH   DALLAS
 7934 MILLER   CLERK            7782 23-JAN-82  1300     10 ACCOUNTING NEW YORK

13 rows selected.
```

Data from the original dmobld.sql script showed that ADAMS the CLERK was hired on January 12th, 1983. Since the query above is for data some 11 days earlier, ADAMS is missing from the list. The following is the same query with "asof_dtm" set one year earler.

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

PL/SQL procedure successfully completed.

SQL>
SQL> select empno, ename, job, mgr_emp_nk1, hiredate, sal, deptno, dname, loc
  2   from  emp_asof e, dept_asof d where e.dept_id = d.id
  3   order by empno;

EMPNO ENAME    JOB       MGR_EMP_NK1 HIREDATE    SAL DEPTNO DNAME      LOC
----- -------- --------- ----------- --------- ----- ------ ---------- --------
 7369 SMITH    CLERK            7839 17-DEC-80   800     10 ACCOUNTING NEW YORK
 7499 ALLEN    SALESMAN         7698 20-FEB-81  1600     20 SALES      ST LOUIS
 7521 WARD     SALESMAN         7698 22-FEB-81  1250     20 SALES      ST LOUIS
 7566 JONES    MANAGER          7839 02-APR-81  2975     20 RESEARCH   DALLAS
 7654 MARTIN   SALESMAN         7698 28-SEP-81  1250     20 SALES      ST LOUIS
 7698 BLAKE    MANAGER          7839 01-MAY-81  2850     20 SALES      ST LOUIS
 7782 CLARK    MANAGER          7839 09-JUN-81  2450     10 ACCOUNTING NEW YORK
 7788 SCOTT    ANALYST          7566 12-JUN-81  3000     20 RESEARCH   DALLAS
 7839 KING     PRESIDENT             17-NOV-81  5000     10 ACCOUNTING NEW YORK
 7844 TURNER   SALESMAN         7698 08-SEP-81  1500     20 SALES      ST LOUIS
 7876 ADAMS    CLERK            7566 22-NOV-81  1100     20 RESEARCH   DALLAS
 7900 JAMES    CLERK            7698 03-DEC-81   950     20 SALES      ST LOUIS
 7902 FORD     ANALYST          7566 03-DEC-81  3000     20 RESEARCH   DALLAS

13 rows selected.
```

Notice is that the SALES department is in St. Louis. Sometime during 1982, the SALES department must have been moved to Chicago. With a HIREDATE of January 23rd, 1982, MILLER is not on the list. However, ADAMS the CLERK is back on the list. Given the sequential assignment of EMPNO 7876, ADAMS was hired, left, and re-hired with the same EMPNO (but a new EMP record). More interesting data can be seen on September 1st, 1981 and June 1st, 1981.

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 as a "LOG" or "EFF" table type has a "POP" or undo package generated for it. The POP function can undo all DML on a record. Below, department 40 will be staffed with new hires and transfers.

```
SQL>
```

```
SQL> select id did, deptno, dname, loc from dept_act;

DID     DEPTNO DNAME          LOC
--- ---------- -------------- -------------
  1         10 ACCOUNTING     NEW YORK
  2         20 RESEARCH       DALLAS
  3         30 SALES          CHICAGO
  4         40 OPERATIONS     BOSTON

4 rows selected.

SQL>
SQL> select empno, ename, job, dept_id did, dept_nk1 dept, aud_beg_usr, aud_beg_dtm
  2   from emp_act where dept_nk1 = 40;

no rows selected

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

PL/SQL procedure successfully completed.

SQL>
SQL> -- Add a new manager MCMURRY to the Operations Department
SQL> insert into emp_act (empno, ename, job, mgr_emp_nk1, hiredate, sal, dept_id)
  2   values (8156, 'MCMURRY', 'MANAGER', 7839, sysdate, 2975, 4);

1 row created.

SQL>
SQL> -- Add a new analyst WALKER to the Operations Department
SQL> insert into emp_act (empno, ename, job, mgr_emp_nk1, hiredate, sal, dept_nk1)
  2   values (8157, 'WALKER', 'ANALYST', 8156, sysdate, 3000, 40);

1 row created.

SQL>
SQL> -- Transfer an analyst SCOTT to the Operations Department
SQL> update emp_act
  2    set  dept_id   = 4
  3       ,mgr_emp_nk1 = 8156
  4   where empno = 7788;

1 row updated.

SQL>
SQL> -- Transfer a clerk JAMES to the Operations Department
SQL> update emp_act
  2    set  dept_nk1  = 40
  3       ,mgr_emp_nk1 = 8156
  4   where empno = 7902;

1 row updated.

SQL>
SQL> commit;

Commit complete.
```

In the sequence of steps above, department 40 is confirmed to have no employees. SMITH adds two new employees to the department and transfers 2 existing employees . The transaction is commited. However, something went wrong.

```
SQL>
SQL> select id eid, empno, ename, job, dept_id did, dept_nk1 dept,
  2        aud_beg_usr, aud_beg_dtm
  3   from emp_act where dept_nk1 = 40;

EID EMPNO ENAME    JOB       DID DEPT AUD_BEG AUD_BEG_D
--- ----- -------- --------- --- ---- ------- ---------
 18  7902 FORD     ANALYST     4   40 SMITH   18-APR-12
 20  7788 SCOTT    ANALYST     4   40 SMITH   18-APR-12
 23  8156 MCMURRY  MANAGER     4   40 SMITH   18-APR-12
 24  8157 WALKER   ANALYST     4   40 SMITH   18-APR-12
```

```
4 rows selected.
```

In the department listng above, FORD has been mistakenly transfered to department 40. MILLER will use the POP procedure to undo the "committed" error and re-transfer the correct employee.

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

PL/SQL procedure successfully completed.

SQL>
SQL> select emp_id eid, empno, ename, dept_id did,
  2        aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3   from emp_hist where empno = 7902;

EID EMPNO ENAME    DID AUD_BEG AUD_BEG_D AUD_END AUD_END_D
--- ----- -------- --- ------- --------- ------- ---------
 18  7902 FORD       2 SMITH   02-DEC-81 SMITH   18-APR-12

1 row selected.

SQL>
SQL> -- Undo the transfer of FORD to the Operations Department
SQL> declare
  2    emp_id number;
  3  begin
  4    select id into emp_id from emp_act where empno = 7902;
  5    emp_pop.at_server(emp_id);
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL> -- Transfer a clerk JAMES to the Operations Department
SQL> update emp_act
  2    set  dept_nk1  = 40
  3       ,mgr_emp_nk1 = 8156
  4   where empno = 7900;

1 row updated.

SQL>
SQL> select id eid, empno, ename, job, dept_id did, dept_nk1 dept,
  2        aud_beg_usr, aud_beg_dtm
  3   from emp_act where dept_nk1 = 40 or empno = 7902;

EID EMPNO ENAME    JOB       DID DEPT AUD_BEG AUD_BEG_D
--- ----- -------- --------- --- ---- ------- ---------
 17  7900 JAMES    CLERK       4   40 MILLER  18-APR-12
 18  7902 FORD     ANALYST     2   20 SMITH   01-DEC-81
 20  7788 SCOTT    ANALYST     4   40 SMITH   18-APR-12
 23  8156 MCMURRY  MANAGER     4   40 SMITH   18-APR-12
 24  8157 WALKER   ANALYST     4   40 SMITH   18-APR-12

5 rows selected.

SQL>
SQL> select emp_id eid, empno, ename, dept_id did,
  2        aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3   from emp_hist where empno = 7902;

no rows selected

SQL>
SQL> commit;

Commit complete.
```

In the first query above, FORD's history record is displayed, showing that he was in DEPT_ID 2. MILLER uses the EMP_POP.AT_SERVER procedures to "undo" the last DML on FORD's employment record. Then, MILLER correctly transfers JAMES to department 40 (DEPT_ID 4).

Notice that the EMP_ACT record for FORD in the last query shows the original AUD_BEG_DTM of December 1st, 1981. Even though the EMP_ACT and EMP_HIST tables dont' show any record of the POP occuring, the POP was recorded in the EMP_PDAT table.

```
SQL>
SQL> select emp_id eid, pop_dml, pop_usr, pop_dtm, empno, ename,
  2         aud_beg_usr, aud_beg_dtm, aud_prev_beg_usr, aud_prev_beg_dtm
  3    from emp_pdat;

EID POP_DM POP_USR POP_DTM   EMPNO ENAME    AUD_BEG AUD_BEG_D AUD_PRE AUD_PREV_
--- ------ ------- --------- ----- -------- ------- --------- ------- ---------
 18 UPDATE MILLER  18-APR-12  7902 FORD     SMITH   18-APR-12 SMITH   01-DEC-81

1 row selected.
```

The query above shows that MILLER "popped" an UPDATE statement on FORD's employment record (ID 18) that was performed by SMITH on April 12th, 2012. Note that the AUD_PREV_BEG_USR of SMITH is the value restored to the AUD_BEG_USR in the EMP_ACT record. Also note that the AUD_PREV_BEG_DTM of December 1st, 1981 is the value restored to the AUD_BEG_DTM in the EMP_ACT record. These values, along with the ID from the original EMP_ACT record, can be used to trace the audit trail back to the original EMP_ACT or EMP_HIST records, regardless of how many times a record is "popped".

Exercise #5: All Instances View

Command Line:

```
sqlplus /nolog @e5
```

Exercise #5 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 record that was originally created in the EMP table is tracked through all the DML performed on it (including the use of the POP procedure). The original record is tracked via the ID it was originally assigned in the EMP table. In the EMP table, each record represents an instance of employment.

```
SQL>
SQL> select empno, ename, id eid, stat, dept_id did,
  2         aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from emp_all order by empno, id;

EMPNO ENAME    EID STAT DID AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
----- -------- --- ---- --- -------- --------- -------- ---------
 7301 ELLISON    1 HIST   1 DAVIS    30-OCT-80 THOMPSON 30-JUN-81
 7344 DAVIS      2 HIST   1 SMITH    28-NOV-81 SMITH    10-DEC-81
 7369 SMITH      3 ACT    2 SMITH    28-FEB-83
 7499 ALLEN      4 ACT    3 THOMPSON 14-MAY-81
 7521 WARD       5 ACT    3 THOMPSON 14-MAY-81
 7566 JONES      6 ACT    2 SMITH    30-NOV-81
 7654 MARTIN     7 HIST   3 THOMPSON 18-APR-81 THOMPSON 14-MAY-81
 7654 MARTIN    14 ACT    3 SMITH    26-SEP-81
 7698 BLAKE      8 ACT    3 SMITH    28-NOV-81
 7782 CLARK      9 ACT    1 SMITH    01-DEC-81
 7788 SCOTT     10 HIST   2 THOMPSON 10-JUN-81 JAMES    07-MAR-82
 7788 SCOTT     20 ACT    4 SMITH    18-APR-12
 7839 KING      11 HIST   1 THOMPSON 14-JUN-81 SMITH    29-AUG-81
 7839 KING      15 ACT    1 SMITH    15-NOV-81
 7840 LANE      12 HIST   1 THOMPSON 15-AUG-81 SMITH    28-NOV-81
 7844 TURNER    13 ACT    3 SMITH    09-SEP-81
 7876 ADAMS     16 HIST   2 SMITH    24-NOV-81 JAMES    15-JUN-82
```

```
 7876 ADAMS     21 ACT    2 SMITH    09-JAN-83
 7900 JAMES     17 ACT    4 MILLER   18-APR-12
 7902 FORD      18 ACT    2 SMITH    02-DEC-81
 7934 MILLER    19 ACT    1 JAMES    22-JAN-82
 8156 MCMURRY   23 ACT    4 SMITH    18-APR-12
 8157 WALKER    24 ACT    4 SMITH    18-APR-12

23 rows selected.
```

In the EMP_ALL query above, EMPNO 7654 MARTIN has 2 instances of employment. The first instance was last modified/entered on April 18th, 1981 and deleted on May 14th, 1981 as captured by EMP ID 7. Later, MARTIN returned to the company, maintained the same EMPNO, but received a new instance of employment as captured by EMP ID 14. The STAT data shows that EMP ID is "ACT" or active, so MARTIN is still employed on this, the second instance of employment. In the example below, SMITH will be retired from the company by deleting the current employment instance from EMP_ACT.

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

PL/SQL procedure successfully completed.

SQL>
SQL> -- SMITH retires today
SQL> delete from emp_act
  2   where empno = 7369;

1 row deleted.

SQL>
SQL> select empno, ename, id eid, stat, dept_id did,
  2         aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3   from emp_all order by empno, id;

EMPNO ENAME    EID STAT DID AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
----- -------- --- ---- --- -------- --------- -------- ---------
 7301 ELLISON    1 HIST   1 DAVIS    30-OCT-80 THOMPSON 30-JUN-81
 7344 DAVIS      2 HIST   1 SMITH    28-NOV-81 SMITH    10-DEC-81
 7369 SMITH      3 HIST   2 SMITH    28-FEB-83 MILLER   18-APR-12
 7499 ALLEN      4 ACT    3 THOMPSON 14-MAY-81
 7521 WARD       5 ACT    3 THOMPSON 14-MAY-81
 7566 JONES      6 ACT    2 SMITH    30-NOV-81
 7654 MARTIN     7 HIST   3 THOMPSON 18-APR-81 THOMPSON 14-MAY-81
 7654 MARTIN    14 ACT    3 SMITH    26-SEP-81
 7698 BLAKE      8 ACT    3 SMITH    28-NOV-81
 7782 CLARK      9 ACT    1 SMITH    01-DEC-81
 7788 SCOTT     10 HIST   2 THOMPSON 10-JUN-81 JAMES    07-MAR-82
 7788 SCOTT     20 ACT    4 SMITH    18-APR-12
 7839 KING      11 HIST   1 THOMPSON 14-JUN-81 SMITH    29-AUG-81
 7839 KING      15 ACT    1 SMITH    15-NOV-81
 7840 LANE      12 HIST   1 THOMPSON 15-AUG-81 SMITH    28-NOV-81
 7844 TURNER    13 ACT    3 SMITH    09-SEP-81
 7876 ADAMS     16 HIST   2 SMITH    24-NOV-81 JAMES    15-JUN-82
 7876 ADAMS     21 ACT    2 SMITH    09-JAN-83
 7900 JAMES     17 ACT    4 MILLER   18-APR-12
 7902 FORD      18 ACT    2 SMITH    02-DEC-81
 7934 MILLER    19 ACT    1 JAMES    22-JAN-82
 8156 MCMURRY   23 ACT    4 SMITH    18-APR-12
 8157 WALKER    24 ACT    4 SMITH    18-APR-12

23 rows selected.

SQL>
SQL> select empno, ename, id eid, dept_id did, aud_beg_usr, aud_beg_dtm
  2    from emp_act where empno = 7369;

no rows selected

SQL>
SQL> select empno, ename, emp_id eid, last_active, dept_id did,
  2         aud_beg_usr, aud_beg_dtm, aud_end_usr, aud_end_dtm
  3    from emp_hist where empno = 7369 order by aud_beg_dtm;
```

```
EMPNO ENAME    EID LAST MGR_EMP_ID DID AUD_BEG_ AUD_BEG_D AUD_END_ AUD_END_D
----- -------- --- ---- ---------- --- -------- --------- -------- ---------
 7369 THOMPSON   3               1   1 DAVIS    15-DEC-80 THOMPSON 25-JUN-81
 7369 THOMPSON   3              11   1 THOMPSON 25-JUN-81 SMITH    21-AUG-81
 7369 SMITH      3              12   1 SMITH    21-AUG-81 SMITH    01-DEC-81
 7369 SMITH      3              15   1 SMITH    01-DEC-81 SMITH    26-FEB-83
 7369 SMITH      3 Y            18   2 SMITH    26-FEB-83 MILLER   18-APR-12

5 rows selected.

SQL>
SQL> commit;

Commit complete.
```

Notice that the STAT of SMITH (EMP ID 3) has changed from ACT to HIST. This shows that EMP ID 3 is no longer an active instance of employment (no longer in EMP_ACT), which is confirmed by the EMP_ACT query. The last query shows that only 1 record, the last record, has a "Y" for LAST_ACTIVE. When the delete was performed on EMP_ACT, this flag was set to recognize there are no more active view records for this entity. Though EMP_ACT has many records for EID 3, the EMP_ALL view only returns the last active record.

Exercise #6: Transportable ASOF Data

Command Line:

```
sqlplus /nolog @e6
```

Exercise #6 modifies the database. The "drop_demo_users.sql", "create_demo_users.sql", "e1.sql", and "e5.sql" scripts must be used to reset the database before re-running this exercise.

Oracle has a mechanism that will hold rollback entries (prevent them from being re-used) for use by "flashback" queries to provide data as it was at some previous point in time. Because flashback query used the rollback segments, transporting this flashback query data has limitations. DTGen provides a schema based approach to flashback query. This schema based data is more easily transported to different databases and servers than rollback segment data. The loading of data for this demonstration is a transport of ASOF data to this schema instance. Using native flashback query would have required rebuilding Oracle's rollback segment data. Additionally, the data in this exercise was not created perfectly, i.e. It was edited during the production process. It is impossible to edit any historical mistakes in the rollback segments.

```
Login to dtgen_db_demo
Connected.
SQL>
SQL> REM Export the 4 tables
SQL> REM
SQL> host exp &DB_NAME./&DB_PASS. LOG=e6_exp.log FILE=e6.dmp
TABLES=dept,dept_aud,emp,emp_hist
```

In this exercise, the "original" export and import utilities will be used. Oracle recommends the use of the data pump versions of export and import. However, the data pump versions require the database kernel to write and read O/S files, which require directory and file setup in the kernel (data dictionary). Alternatively, the original export and import utilities operate as database clients on behalf of the O/S user and do not require this special kernel configuration to execute. Though they are not used in this exercise, the data pump version of these ultilities is preferred.

```
Connected to: Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set

About to export specified tables via Conventional Path ...
. . exporting table                           DEPT          4 rows exported
. . exporting table                       DEPT_AUD          2 rows exported
. . exporting table                            EMP         13 rows exported
. . exporting table                       EMP_HIST         26 rows exported
Export terminated successfully without warnings.
```

The output listing above is from the "e6_exp.log" LOG file created by the export. It shows a successfull export of the 4 main data tables from the database to the local "e6.dmp" export file.

```
SQL> REM  These constraints are only used to assist Data Dictionary Queries
SQL> REM  Since they were created, these constraints were never enabled
SQL> REM
SQL> alter view emp_act  drop constraint emp_act_fk1;

View altered.

SQL> alter view emp_act  drop constraint emp_act_fk2;

View altered.

SQL> alter view emp_l  drop constraint emp_l_fk1;

View altered.

SQL> alter view emp_l  drop constraint emp_l_fk2;

View altered.

SQL> alter view emp_all  drop constraint emp_all_fk1;

View altered.

SQL> alter view emp_all  drop constraint emp_all_fk2;

View altered.

SQL> alter view emp_f  drop constraint emp_f_fk1;

View altered.

SQL> alter view emp_f  drop constraint emp_f_fk2;

View altered.

SQL> alter view emp_asof drop constraint emp_asof_fk1;

View altered.

SQL> alter view emp_asof drop constraint emp_asof_fk2;

View altered.
```

The constraints listed above must be removed before the 4 tables can be dropped.

```
SQL> REM Drop the tables
SQL> REM
SQL> drop table emp_hist;

Table dropped.

SQL> drop table emp;

Table dropped.

SQL> drop table dept_aud;

Table dropped.

SQL> drop table dept;
```

```
Table dropped.
```

With the 4 tables dropped, the original ASOF data from the previous exercises no longer exists in the database.

```
SQL> REM Import the 4 tables
SQL> REM
SQL> host imp &DB_NAME./&DB_PASS. LOG=e6_imp.log FILE=e6.dmp
```

The import listed above re-creates the 4 tables.

```
Connected to: Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

Export file created by EXPORT:V10.02.01 via conventional path
import done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
. importing DTGEN_DB_DEMO's objects into DTGEN_DB_DEMO
. importing DTGEN_DB_DEMO's objects into DTGEN_DB_DEMO
. . importing table                          "DEPT"          4 rows imported
. . importing table                     "DEPT_AUD"          2 rows imported
. . importing table                          "EMP"         13 rows imported
. . importing table                     "EMP_HIST"         26 rows imported
About to enable constraints...
Import terminated successfully without warnings.
```

The output listing above is from the "e6_imp.log" LOG file created by the export. It shows a successfull import of the 4 main data tables back into the database from the local "e6.dmp" export file.

```
SQL> REM While not necessary in this exercise, it is important to demonstrate
SQL> REM   a possible method of transporting the sequence generators
SQL>
SQL> drop   sequence dept_seq;

Sequence dropped.

SQL> create sequence dept_seq;

Sequence created.

SQL> declare
  2    max_id number;
  3    junk   number;
  4  begin
  5    dbms_output.enable;
  6    select max(id) into max_id
  7    from (select max(id) id from dept
  8          union
  9                 select max(dept_id) id from dept_aud
 10                 );
 11    for i in 1 .. max_id
 12    loop
 13       select dept_seq.nextval into junk from dual;
 14    end loop;
 15    dbms_output.put_line('DEPT_SEQ incremented to ' || max_id);
 16  end;
 17  /
DEPT_SEQ incremented to 4

PL/SQL procedure successfully completed.

SQL>
SQL> drop   sequence emp_seq;

Sequence dropped.

SQL> create sequence emp_seq;

Sequence created.

SQL> declare
```

```
  2    max_id number;
  3    junk   number;
  4  begin
  5    dbms_output.enable;
  6    select max(id) into max_id
  7    from (select max(id) id from emp
  8          union
  9                 select max(emp_id) id from emp_hist
 10                 );
 11    for i in 1 .. max_id
 12    loop
 13       select emp_seq.nextval into junk from dual;
 14    end loop;
 15    dbms_output.put_line('EMP_SEQ incremented to ' || max_id);
 16  end;
 17  /
EMP_SEQ incremented to 21

PL/SQL procedure successfully completed.

SQL>
SQL> commit;

Commit complete.
```

A review of the "e1.sql" script will show a similar technique for creating the schema for this demonstration. These sequences must be considered only if data modification will continue at the new location. Note that this technique was not necessary for this exercise because the sequences remained unchanged through the export, table drop, and import.

```
SQL> execute util.set_asof_dtm(to_timestamp('1983-01-01', 'YYYY-MM-DD'))

PL/SQL procedure successfully completed.

SQL>
SQL> select empno, ename, job, mgr_emp_nk1, hiredate, sal, deptno, dname, loc
  2   from  emp_asof e, dept_asof d where e.dept_id = d.id
  3   order by empno;

EMPNO ENAME    JOB       MGR_EMP_NK1 HIREDATE    SAL DEPTNO DNAME      LOC
----- -------- --------- ----------- --------- ----- ------ ---------- --------
 7369 SMITH    CLERK            7839 17-DEC-80   800     10 ACCOUNTING NEW YORK
 7499 ALLEN    SALESMAN         7698 20-FEB-81  1600     30 SALES      CHICAGO
 7521 WARD     SALESMAN         7698 22-FEB-81  1250     30 SALES      CHICAGO
 7566 JONES    MANAGER          7839 02-APR-81  2975     20 RESEARCH   DALLAS
 7654 MARTIN   SALESMAN         7698 28-SEP-81  1250     30 SALES      CHICAGO
 7698 BLAKE    MANAGER          7839 01-MAY-81  2850     30 SALES      CHICAGO
 7782 CLARK    MANAGER          7839 09-JUN-81  2450     10 ACCOUNTING NEW YORK
 7788 SCOTT    ANALYST          7566 09-DEC-82  3000     20 RESEARCH   DALLAS
 7839 KING     PRESIDENT             17-NOV-81  5000     10 ACCOUNTING NEW YORK
 7844 TURNER   SALESMAN         7698 08-SEP-81  1500     30 SALES      CHICAGO
 7900 JAMES    CLERK            7698 03-DEC-81   950     30 SALES      CHICAGO
 7902 FORD     ANALYST          7566 03-DEC-81  3000     20 RESEARCH   DALLAS
 7934 MILLER   CLERK            7782 23-JAN-82  1300     10 ACCOUNTING NEW YORK

13 rows selected.
```

The above listing is the first of several ASOF queries that are repeated from Exercise #3. These reports show that thie exercise successfully transported ASOF data out of and back into the database.