DTGen Tiers Demonstration

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

Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates.

Table of Contents

Introduction:

The exercises in this demonstration are focused on DTGen functionality that enables and enhances tiered deployment. 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.

Tiered deployment as described here includes 2 forms of deployment:

- database/mid-tier server tiers
- database user/schema tiers

Multi-tiered hardware deployment is a common aspect of many systems. The database and mid-tier servers use a software deployment stragey that increases capacity and improves security. Capacity can be increased by adding servers to existing systems instead of replacing centralized servers with larger servers. Security can be improved through the user of layered access to applications and data.

Layered security can also be found in many systems. Oracle e-Business Suite layers application schema behind a common database applicaiton login. The Defense Information Services (DISA) Oracle Database Security Readiness Review ("http://iase.disa.mil/stigs/downloads/zip/unclassified_oracle10_v8r1.8_checklist_20100827.zip") includes checklist items that require layered security. Specific references in the " U_DB_oracle10_v8r1.8_Checklist_20100827.pdf" file include:

- V0005683 - Application object owner accounts should be disabled when not performing installation or maintenance actions.
- V0015613 - Each database user, application or process should have an individually assigned

account.
- V0015629 - Application users privileges should be restricted to assignment using application user roles.
- V0003847 - Database application user accounts should be denied storage usage for object creation within the database.

The "basic" and "asof" demonstrations 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 asof demonstration. Additionally, the "DML & API Calls" and "Other Object Location" diagrams in the "DTGen_Notes.pdf" document in the "docs" directory provide a graphical layout of the multi-tier deployments created by DTGen.

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: Simple Mid-Tier

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 DEMO1 Items from DTGEN
2. Creates new DEMO1 Items in DTGEN
3. Generates the DEMO1 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 DEMO1 Schema from DTGEN
create a DEMO1 Schema in DTGEN
Generate Demo1 Application
Capture install_db.sql Script
```

Step 4 is captured in the "install_db.sql" file. This file is about 78 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

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

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
```

```
     EMPNO ENAME            JOB       MGR_EMP_NK1 HIREDATE     SAL   DEPT_NK1
---------- ---------------- --------- ----------- --------- ------ ----------
      7369 SMITH            CLERK            7902 17-DEC-80    800         20
      7499 ALLEN            SALESMAN         7698 20-FEB-81   1600         30
      7521 WARD             SALESMAN         7698 22-FEB-81   1250         30
      7566 JONES            MANAGER          7839 02-APR-81   2975         20
      7654 MARTIN           SALESMAN         7698 28-SEP-81   1250         30
      7698 BLAKE            MANAGER          7839 01-MAY-81   2850         30
      7782 CLARK            MANAGER          7839 09-JUN-81   2450         10
      7788 SCOTT            ANALYST          7566 09-DEC-82   3000         20
      7839 KING             PRESIDENT             17-NOV-81   5000         10
      7844 TURNER           SALESMAN         7698 08-SEP-81   1500         30
      7876 ADAMS            CLERK            7788 12-JAN-83   1100         20
      7900 JAMES            CLERK            7698 03-DEC-81    950         30
      7902 FORD             ANALYST          7566 03-DEC-81   3000         20
      7934 MILLER           CLERK            7782 23-JAN-82   1300         10
```

With the completion of exercise 1, a new application was defined in DTGen, generated, and loaded into the database.

Exercise #2: Materialized Views

Command Line:

```
sqlplus /nolog @e2
```

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

In the exercise #1, a basic generation was completed. The results of that generation were loaded into a new schema. This exercise, and the following exercises, will examine more closely what was generated.

Exercise #3: User Security

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

Exercise #4: Global Locks

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 Manipulation Language - insert, update, and delete)

Exercise #5: DB Tier Streamlining

Command Line:

```
    sqlplus /nolog @e5
```

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

As stated in the Introduction, system capacity can be increased by adding servers to existing systems instead of replacing centralized servers with larger servers. A chief goal in this method of capacity increase is to isolate application logic from database storage (i.e. DML).