CSC 453 Database Technologies

Assignment 5 (5/15)

Due 5:45pm, Friday 5/30. (NOTE: There has been some confusion over assignment submission deadlines. The dropbox for Assignment 5 will close promptly at 5:45pm on 5/30 and no late or emailed submissions will be accepted. All Assignment 5 submissions must be uploaded to the dropbox by 5:45pm on 5/30 – no exceptions.)

Reading: In Elmasri and Navathe, Sections 13.1, 26.1, and 13.3.2. For PL/SQL, I recommend Chapters 1, 3-5, and 8-10 of *Oracle Database 10g PL/SQL Programming* by Urman, Hardman, and McLaughlin. (If you want to read ahead on JDBC, I recommend Chapter 15 of *Oracle Database 10g SQL*, by Price.) Both of these books are available through the DePaul library's E-Books collection at library.depaul.edu (search on title and/or author, and restrict the search results to E-Books).

Problems:

1 (PL/SQL). Consider the table STUDENT with attributes ID, Name, Midterm, Final, and Homework, and the table WEIGHTS with attributes MidPct, FinPct, and HWPct defined and populated by the following script:

```
DROP TABLE STUDENT CASCADE CONSTRAINTS;
CREATE TABLE STUDENT
(
     ID         CHAR(3),
     Name       VARCHAR2(20),
     Midterm    NUMBER(3,0)     CHECK (Midterm>=0 AND Midterm<=100),
     Final      NUMBER(3,0)     CHECK (Final>=0 AND Final<=100),
     Homework   NUMBER(3,0)     CHECK (Homework>=0 AND Homework<=100),

     PRIMARY KEY (ID)
);
INSERT INTO STUDENT VALUES ( '445', 'Seinfeld', 85, 90, 99 );
INSERT INTO STUDENT VALUES ( '909', 'Costanza', 74, 72, 86 );
INSERT INTO STUDENT VALUES ( '123', 'Benes', 93, 89, 91 );
INSERT INTO STUDENT VALUES ( '111', 'Kramer', 99, 91, 93 );
INSERT INTO STUDENT VALUES ( '667', 'Newman', 78, 82, 83 );
INSERT INTO STUDENT VALUES ( '888', 'Banya', 50, 65, 50 );
SELECT * FROM STUDENT;

DROP TABLE WEIGHTS CASCADE CONSTRAINTS;
CREATE TABLE WEIGHTS
(
     MidPct     NUMBER(2,0) CHECK (MidPct>=0 AND MidPct<=100),
     FinPct     NUMBER(2,0) CHECK (FinPct>=0 AND FinPct<=100),
     HWPct      NUMBER(2,0) CHECK (HWPct>=0 AND HWPct<=100)
);
```

```
INSERT INTO WEIGHTS VALUES ( 30, 30, 40 );
SELECT * FROM WEIGHTS;
```

Write an anonymous PL/SQL block that will do the following:

First, report the three weights found in the WEIGHTS table. (You may assume that the WEIGHTS table contains only one record.) Next, output the name of each student in the STUDENT table and their overall score, computed as x percent Midterm, y percent Final, and z percent Homework, where x, y, and z are the corresponding percentages found in the WEIGHTS table. (You may assume that x+y+z=100.) Also convert each student's overall score to a letter grade by the rule 90-100=A, 80-89.99=B, 65-79.99=C, 0-64.99=F, and include the letter grade in the output. Output each student's information on a separate line. For the sample data given above, the output should be

```
Weights are 30, 30, 40
445 Seinfeld 92.1 A
909 Costanza 78.2 C
123 Benes 91 A
111 Kramer 94.2 A
667 Newman 81.2 B
888 Banya 54.5 F
```

(Of course, this is just an example – your PL/SQL block should work in general, not just for the given sample data.)

2 (Triggers). Consider the PROJECT and ASSIGNMENT tables defined and populated by the following script:

```
DROP TABLE ASSIGNMENT CASCADE CONSTRAINTS;
DROP TABLE PROJECT CASCADE CONSTRAINTS;

CREATE TABLE PROJECT
(
 Code        NUMBER(3),
 Name        VARCHAR2(30),

 CONSTRAINT PK_PROJECT PRIMARY KEY (Code)
);

CREATE TABLE ASSIGNMENT
(
 ID          CHAR(5),
 Name        VARCHAR2(20),
 ProjCode    NUMBER(3),
 Hours       NUMBER(*,0) CHECK (Hours>0),

 CONSTRAINT PK_ASSIGNMENT PRIMARY KEY (ID, ProjCode),

 CONSTRAINT FK_ASSIGNMENT_PROJECT FOREIGN KEY (ProjCode) REFERENCES PROJECT (Code)
);

INSERT INTO PROJECT VALUES ( 101, 'Alpha' );
INSERT INTO PROJECT VALUES ( 222, 'Beta' );
INSERT INTO PROJECT VALUES ( 355, 'Gamma' );
INSERT INTO PROJECT VALUES ( 973, 'Delta' );
```

```
INSERT INTO ASSIGNMENT VALUES ( '55055', 'Smith', 101, 20 );
INSERT INTO ASSIGNMENT VALUES ( '55055', 'Smith', 222, 10 );
INSERT INTO ASSIGNMENT VALUES ( '39002', 'Hammond', 973, 25 );
INSERT INTO ASSIGNMENT VALUES ( '00001', 'Preston', 355, 5 );
INSERT INTO ASSIGNMENT VALUES ( '10000', 'Logan', 355, 5 );
INSERT INTO ASSIGNMENT VALUES ( '00777', 'Bond', 222, 20 );

SELECT * FROM PROJECT;
SELECT * FROM ASSIGNMENT;
```

Suppose we did not want to store information on projects that no longer have any employees assigned to them. Write a trigger that will do the following:

Any time a change is made to the ASSIGNMENT table that may cause the last employee assigned to a project to be removed from that project, display a list of all projects that have at least one employee assigned to them, and update the PROJECT table by removing any projects that no longer have any employees assigned to them. (Hint: Any time an UPDATE or DELETE is done to the ASSIGNMENT table, query the table to find a list of projects that have at least one employee working on them, output this list to DBMS_OUTPUT to verify it, and remove from the PROJECT table all projects that are not in this list...)

For example, for the initial tables:

```
CODE         NAME
------------ --------------
101          Alpha
222          Beta
355          Gamma
973          Delta

ID    NAME         PROJCODE       HOURS
----- ------------ -------------- ------------
55055 Smith        101            20
55055 Smith        222            10
39002 Hammond      973            25
00001 Preston      355            5
10000 Logan        355            5
00777 Bond         222            20
```

If Preston is removed from the ASSIGNMENT table with a DELETE, there should be no change to PROJECT, since Logan is still assigned to project 355:

```
CODE         NAME
------------ --------------
101          Alpha
222          Beta
355          Gamma
973          Delta

ID    NAME         PROJCODE       HOURS
----- ------------ -------------- ------------
55055 Smith        101            20
55055 Smith        222            10
39002 Hammond      973            25
10000 Logan        355            5
00777 Bond         222            20
```

However, if Smith is then moved from project 101 to project 973 with an UPDATE, then there will be no longer be any employees assigned to project 101 and the tables should become:

```
CODE         NAME
------------ --------------
222          Beta
355          Gamma
973          Delta

ID    NAME         PROJCODE       HOURS
----- ------------ -------------- ------------
55055 Smith        973            20
55055 Smith        222            10
39002 Hammond      973            25
10000 Logan        355            5
00777 Bond         222            20
```

(Be sure that your trigger will work in general, not just for the given sample data.)

Remarks:

1. Submit a .doc, .txt or other electronic document with your answers under "Assignment 5". You do not have to submit any output for this assignment.

2. Copy and paste your anonymous PL/SQL block marked as "Problem 1" and your PL/SQL trigger definition marked as "Problem 2" into a single .doc or .txt file and submit the file under "Assignment 5". You may cut and paste the scripts I have supplied into SQLDeveloper to set up the tables so that you can test your programs, but your submitted answers should include only the code you have written to solve the problems – not any of my code. Be sure that your submitted code is in a plain text form that can be cut and pasted into a script file or SQLDeveloper for testing.

3. It is your responsibility to make sure that the file you have uploaded is readable and in the correct location. You should check that you can successfully download your submitted file back from the course web site immediately after submitting it to be sure that it has been uploaded correctly.

4. Remember that all work must be completed individually.

Eric J. Schwabe – 05/15/14