Assignment 2 (10% of total marks)

Due date: Saturday, 16 November 2019

Scope:

The tasks of this assignment cover topics on **PLSQL as well as Transaction processing and concurrency**.

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

Please read carefully information listed below.

This assignment contributes to 10% of the total assessment mark for the subject CSCI235.

A submission procedure is explained at the end of specification.

This assignment consists of 4 tasks and specification of each task starts from a new page.

A policy regarding late submissions is included in the subject outline.

For all the implemented tasks, your report or output must include a listing of all PL/SQL statements processed. To achieve that put the following SQL*Plus commands in all your scripts:

```
SPOOL file-name
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 100
SET PAGESIZE 200
SET SERVEROUTPUT ON
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Assignment Specification:

Preliminary actions
The Assignment 2 folder contains the SQL scripts dbCreate.sql and dbDrop.sql. Execute the script dbCreate.sql to create and to load a sample database.

Task 1 (2.0 marks)

PL/SQL Procedure

Implement a stored PL/SQL procedure PROJECTGROUPS to list the project numbers, titles and the names of employees who work on each project.

The names of employees must be listed in the ascending order. Execute the stored PL/SQL procedure PROJECT GROUPS. A fragment of expected sample printout is given below.

```
1001 Computation: Alvin, Peter
1002 Study methods: Bob, Robert
1003 Racing car: Robert
…
```

When ready append all solutions for the questions above into an SQL script file solution1.sql. Next, process the entire script and spool the output to a spool file solution1.lst.

Deliverables
Submit your spooled file solution1.lst that contains your SQL script and the output from the execution of the script. The report must have no errors related to the implementation of your task and it must list all PL/SQL and SQL statements processed.

Remember to set ECHO option of SQL*Plus to ON!

Task 2 (2.0 marks)

PL/SQL Function

Implement a stored PL/SQL function FINDDEPENDENTS that finds the names of dependents of an employee by given employee's number(e#). The function must return a string of characters that contains all dependents' names. All names of dependents must be separated by commas (,).

Execute the stored PL/SQL function FINDEPENDENTS. A fragment of sample printout is given below:

```
00100 Albert: Bolt, Edee, Judy
00110 Alvin:
00120 Alice: Blures, Edee, Kadi
…
```

When ready append all solutions for the questions above into an SQL script file solution2.sql. Next, process the entire script and spool the output to a spool file solution2.lst.

Deliverables

Submit your spooled file solution2.lst that contains your SQL script and the output from the execution of the script. The report must have no errors related to the implementation of your task and it must list all PL/SQL and SQL statements processed.

Remember to set ECHO option of SQL*Plus to ON!

Task 3 (2.0 marks)

Statement trigger

Implement and comprehensively test a **statement trigger** that enforces the following data access constraint.

A user is allowed to insert a row into a relational table Employee only one time per day.

When ready, create an SQL script solution3.sql with CREATE OR REPLACE TRIGGER statement and with SQL statements used for implementation of the constraint and for comprehensive testing of the trigger. A comprehensive test must consist of at least two cases: one when insertion is accepted and one when insertion is rejected.

Hints:

(1) SYSDATE function returns a value of type DATE that contains the current year, month, day, time, hour, and second.

(2) Time measurement with a precision of up to a single second is acceptable.

(3) A value of an expression SYSDATE+1 is tomorrow the same time as now.

(4) You are allowed to change the structures of a sample database.

Deliverables

Submit a file solution3.lst with a report from processing of SQL script solution3.sql. The report MUST have no errors other than reported by a trigger and the report MUST list all SQL statements processed. The report MUST include ONLY SQL statements and control statements that implement the specifications of Task 3 and NO OTHER statements.

Remember to set ECHO option of SQL*Plus to ON!

Task 4 (4.0 marks)

Transaction Processing and Concurrency Control

This task does not need any implementation. It is a pure "pen&paper" or rather "keyboard&pdf " exercise.

Assume that the structures and the contents of a sample database have been changed in the following way.

```
ALTER TABLE DRIVER ADD (TOTAL_TRIPS NUMBER(7));
UPDATE DRIVER
SET TOTAL_TRIPS = ( SELECT count(*)
          FROM TRIP
          WHER L# = DRIVER.L#);

CREATE TABLE TOTALTRIPS(
 TOTAL_TRIPS NUMBER(7) NOT NULL,
 CONSTRAINT TOTALTRIPS_PKEY PRIMARY KEY(TOTAL_TRIPS) );

INSERT INTO TOTALTRIPS (SELECT SUM(TOTAL_TRIPS)
            FROM DRIVER);
```

A database developer implemented the following stored procedure.

```
CREATE OR REPLACE PROCEDURE INSERT_TRIP(trip_number IN NUMBER,
                  licence_number IN NUMBER,
                  registration_number IN VARCHAR ) IS
total  TOTALTRIPS.TOTAL_TRIPS%TYPE;
BEGIN
 INSERT INTO TRIP VALUES(trip_number, licence_number,
   read and write         registration_number, SYSDATE);
 UPDATE DRIVER                                  write
 SET TOTAL_TRIPS = TOTAL_TRIPS + 1
 WHERE L# = licence_number;
  read
 SELECT TOTAL_TRIPS
 INTO total                               10
 FROM TOTALTRIPS;

 total := total + 1;                      11
   write
 UPDATE TOTALTRIPS
 SET TOTAL_TRIPS = total;                 11

 COMMIT;

EXCEPTION
 WHEN OTHERS THEN
```

```
  DBMS_OUTPUT.PUT_LINE( SQLERRM );
  ROLLBACK;
END;
```

Assume that the procedure INSERT_TRIP has been concurrently processed by many different users that change information about the trips performed by the drivers and the detailed descriptions of the trips.

Decide what isolation level the procedure UPD_AVGYEAR should be processed at and justify your decision.

You have the following two options: READ COMMITTED or SERIALIZABLE.

If you decide that the procedure can be processed at READ COMMITTED level, then as a justification of your decision provide a proof that any concurrent processing of the procedure at READ COMMITTED level does not corrupt a sample database.

If you decide that the procedure can be processed at SERIALIZABLE level, then as a justification provide a sample concurrent processing of the procedure at READ COMMITTED level and the other transactions that corrupts a database. You can apply the two-dimensional visualisation of concurrent processing of database transactions used in the lecture slides.

Submissions

This assignment is due by 9:00 pm (21:00 hours) Saturday, 16 November 2019, **Singapore time**.

Submit the files **solution1.pdf**, **solution2.pdf**, **solutions3.pdf**, and **solution4.pdf** through Moodle in the following way:

1) Zip all the files (Solution1.pdf, solution2.pdf, solution3.pdf, solution4.pdf into one zipped folder.)
2) Access Moodle at **http://moodle.uowplatform.edu.au/**
3) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
4) When successfully logged in, select a site CSCI235 (SP419) Database Systems
5) Scroll down to a section Submissions of Assignments
6) Click at Submit your Assignment 1 here link.
7) Click at a button Add Submission
8) Move the zipped file created in Step 1 above into an area provided in Moodle. You can drag and drop files here to add them. You can also use a link *Add…*
9) Click at a button Save changes,
10) Click at check box to confirm authorship of a submission,
11) When you are satisfied, remember to click at a button Submit assignment.

A policy regarding late submissions is included in the subject outline.

Only one submission per student is accepted.

Assignment 2 is an individual assignment and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

End of specification