Assignment 2 (15% of total marks)

Due date: Sunday, 10 May 2020

Scope:

The tasks of this assignment cover **stored PL/SQL procedure, function, and trigger**. The assignment covers the topics discussed in lecture 7, 8, and 9.

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

Assignment Specification:

Preliminary actions

Download SQL script dbCreateTruck-2020SP20.sql from Moodle. Execute the script to create and to load a sample database for trips made by drivers. The database contains information about the employees working for a transportation company, drivers employed by the company, trucks owned by the company, trips made by the drivers and trucks, and all legs of each trip. It is strongly recommended to discover a conceptual schema (UML diagram) of the database. However, there will be no mark awarded for producing the conceptual schema.

Task 1 (7.0 marks)
Stored PL/SQL Procedure

Implement a stored PL/SQL procedure that insert into a database full information about an employee i.e., the values of the following attributes E#, NAME, DOB, ADDRESS, HIREDATE, L#, STATUS, EXPERIENCE (only for mechanics).

Your procedure should enforce a logical consistency constraint such that "the sets of drivers and mechanics must be disjoint". It means that it is not allowed to have in the tables MECHANIC and DRIVER the rows with the same employee number (E#) and/or the same driving license number (L#).

Execute your procedure twice. The first execution should insert full information about a new employee. The second execution should fail due to the violation of logical consistency constraint given above.

Deliverables

Hand in your solution1.lst in pdf format. Your report MUST have no errors and the report MUST list all SQL statements processed. All SQL statements of the script must be executed with SET ECHO ON, SET FEEEDBACK ON and SET SERVEROUTPUT ON options of SQL*Plus. It is a good idea to set the options at the beginning of the script. The script must be implemented with SQL*Plus. Oracle database server used for the testing and the final execution of the script is up to you. **The printouts that do not satisfy the requirements listed above will score NO MARKS!**

```
SQL> set feedback on
SQL> set line 132
SQL> set pagesize 100
SQL> column name format a20
SQL> column address format a50
SQL> column e# format 99999
SQL> --
SQL> CREATE OR REPLACE PROCEDURE INSEMP( ENUM IN NUMBER,
  2                            ENAME VARCHAR,
  3                            EDOB VARCHAR,
  4                            EADDRESS VARCHAR,
  5                            EHIREDATE VARCHAR,
  6                            ELNUM NUMBER,
  7                            ESTATUS VARCHAR,
  8                            EEXP IN VARCHAR ) IS
  9  anEmpeNum        TRKEMPLOYEE.E#%TYPE;
 10  errorNum         NUMBER(5);
 11  errorMessage     VARCHAR2(200);
 12  BEGIN
 13    INSERT INTO TRKEMPLOYEE VALUES( ENUM, ENAME, EDOB, EADDRESS,
EHIREDATE);
 14
 15  /* if experience is null, then the TRKEMPLOYEE is a driver */
 16    IF EEXP IS NULL THEN -- DRIVER !!!
 17   BEGIN
 18     SELECT E#
 19     INTO anEmpeNum
 20     FROM MECHANIC
 21     WHERE E# = ENUM OR
 22         L# = ELNUM;
 23     DBMS_OUTPUT.PUT_LINE('ATTEMPTED TO INSERT A DRIVER: CONFLICTING
E# OR L# FOUND IN MECHANIC TABLE');
 24     ROLLBACK;
 25   EXCEPTION
 26     WHEN NO_DATA_FOUND THEN
 27       INSERT INTO DRIVER VALUES(ENUM, ELNUM, ESTATUS, null);
 28       DBMS_OUTPUT.PUT_LINE('Employee '||ENUM || ', ' || ENAME ||
 29            ' is successfully created as a driver.');
 30       COMMIT;
 31     WHEN OTHERS THEN
```

```
 32        DBMS_OUTPUT.PUT_LINE('OTHER ERROR');
 33        ROLLBACK;
 34    END;
 35     ELSE -- MECHANIC !!!
 36    BEGIN
 37      SELECT E#
 38      INTO anEmpeNum
 39      FROM DRIVER
 40      WHERE E# = ENUM OR
 41          L# = ELNUM;
 42      DBMS_OUTPUT.PUT_LINE('ATTEMPTED TO INSERT A MECHANIC:
CONFLICTING E# OR L# FOUND IN DRIVER TABLE');
 43      ROLLBACK;
 44    EXCEPTION
 45      WHEN NO_DATA_FOUND THEN
 46        INSERT INTO MECHANIC VALUES(ENUM, ELNUM, ESTATUS, EEXP);
 47        DBMS_OUTPUT.PUT_LINE('Employee '||ENUM || ', ' || ENAME ||
 48             ' is successfully created as a mechanic.');
 49        COMMIT;
 50      WHEN OTHERS THEN
 51        DBMS_OUTPUT.PUT_LINE('OTHER ERROR');
 52        ROLLBACK;
 53    END;
 54     END IF;
 55     EXCEPTION
 56     WHEN OTHERS THEN
 57    errorNum := SQLCODE;
 58    errorMessage := SQLERRM;
 59    DBMS_OUTPUT.PUT_LINE('ERROR IN INSEMP: ');
 60    DBMS_OUTPUT.PUT_LINE(rpad(errorNum,7) || ':' || errorMessage);
 61    ROLLBACK;
 62  END INSEMP;
 63  /

Procedure created.

SQL>
SQL> SHOW ERRORS;
No errors.
SQL>
SQL> EXECUTE INSEMP(777, 'James Bond', '12-DEC-48', 'London', '01-JAN-
66', 666, 'AVAILABLE', NULL);
Employee 777, James Bond is successfully created as a driver.

PL/SQL procedure successfully completed.

SQL> EXECUTE INSEMP(778, 'John Smith', '12-DEC-78', 'LA', '23-MAR-02',
666, 'AVAILABLE', 'EXPERT');
ATTEMPTED TO INSERT A MECHANIC: CONFLICTING E# OR L# FOUND IN DRIVER
TABLE

PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE INSEMP(779, 'Roger Brown', '29-JUL-50', 'USA', '01-MAR-20',
779000, 'AVAILABLE', 'EXPERT');
Employee 779, Roger Brown is successfully created as a mechanic.

PL/SQL procedure successfully completed.

SQL> EXECUTE INSEMP(780, 'John Clark', '22-FEB-74', 'LA', '28-APR-20',
779000, 'AVAILABLE', NULL);
ATTEMPTED TO INSERT A DRIVER: CONFLICTING E# OR L# FOUND IN MECHANIC
TABLE

PL/SQL procedure successfully completed.

SQL> --
SQL> select * from trkemployee;

    E# NAME                 DOB       ADDRESS
HIREDATE
------ -------------------- --------- ----------------------------------
---------------- ---------
     1 John Smith                      42 Victoria St. Hurstville, NSW
2456               03-APR-20
     2 Peter Taylor         12-JAN-60 42 Victoria St. Hurstville, NSW
2456               03-APR-20
     3 John Doe             23-MAR-56 12 Station St. Dapto, NSW 2530
22-JAN-20
     4 John Gray            05-MAY-78 16 Station St. Dapto, NSW 2530
15-FEB-20
     5 Adam Taylor          01-JAN-70 42 Church St. City, NSW 2300
31-DEC-19
     6 Michael Jones        05-OCT-65 23 Waterloo Ave. Surry Hills, NSW
2502             12-SEP-19
     7 Frederic Jones       28-FEB-73 23 Victoria St. Redfern, NSW 2420
13-AUG-19
     8 Peter O'Brien        28-FEB-73 19 Lucas Dr. Horsley, NSW 2530
04-APR-18
     9 John Lucas           16-DEC-56 20 Huxley St. Horsley, NSW 2530
26-FEB-17
    10 John Fox             25-OCT-65 18 Victoria St. Hurstville, NSW
2456               28-AUG-19
    11 Adam Fox             04-MAY-70 45 Victoria St. Hurstville, NSW
2456               22-JUL-19
    12 Phillip Cox          12-DEC-74 5 The Avenue, Rockdale, NSW 2300
04-APR-05
    13 Andrew K. Smith      04-APR-59 42 Bambaramba Ave. Pennant Hills,
NSW 2556         09-NOV-18
    14 Andrew R. Smith      01-APR-83 67 King Cr. Hurstville, NSW 2456
21-JAN-19
    15 Michael Potter       01-APR-85 568 Bong Bong St. Horsley, NSW
2530               03-OCT-18
```

```
   777 James Bond           12-DEC-48 London
01-JAN-66
   779 Roger Brown          29-JUL-50 USA
01-MAR-20

17 rows selected.

SQL> select * from driver;

    E#         L# STATUS     TOTALTRIPMADE
------ ---------- ---------- -------------
     1      10001 AVAILABLE
     3      10002 AVAILABLE
     5      10003 ON LEAVE
     7      20002 BUSY
     9      30005 BUSY
    11      20005 ON LEAVE
    13      10008 ON LEAVE
   777        666 AVAILABLE

8 rows selected.

SQL> select * from mechanic;

    E#         L# STATUS     EXPERIENCE
------ ---------- ---------- ----------
     2      10345 AVAILABLE  EXPERT
     4      10452 AVAILABLE  STANDARD
     6       7773 ON_LEAVE   STANDARD
     8      23302 BUSY       BEGINNER
    10      22205 BUSY       EXPERT
    12      10005 AVAILABLE  BEGINNER
    14      10000 AVAILABLE  BEGINNER
   779     779000 AVAILABLE  EXPERT

8 rows selected.

SQL>
SQL> --
SQL> spool off
```

Task 2 (5.0 marks)
Stored PL/SQL Trigger

Implement **a row trigger** that enforces the following consistency constraint.

A column totalTripMade in the relational table DRIVER is currently does not contain any value. Create a row trigger that automatically updates the values in the column

(totalTripMade) when a new trip made by a driver is inserted into the relational table TRIP. Your trigger, once activated, will compute the total number of trips made by the driver and update the totalTripMade column in the relational table DRIVER. NOTE: You do not need to consider any other cases that may change the value in the column totalTripMade; that is, NO NEED to consider delete and update cases.

When ready, process the SQL script solution2.sql and record the results of processing in a file solution2.lst.

Deliverables

Hand in your solution2.lst in pdf format. Your report MUST have no errors and the report MUST list all SQL statements processed. All SQL statements of the script must be executed with SET ECHO ON, SET FEEEDBACK ON and SET SERVEROUTPUT ON options of SQL*Plus. It is a good idea to set the options at the beginning of the script. The script must be implemented with SQL*Plus. Oracle database server used for the testing and the final execution of the script is up to you. **The printouts that do not satisfy the requirements listed above will score NO MARKS!**

```
SQL> set echo on
SQL> set feedback on
SQL> set serveroutput on
SQL> create or replace trigger totalTripMade
  2  before insert on TRIP
  3  for each row
  4  declare
  5    drvLicense DRIVER.l#%type;
  6  begin
  7    update DRIVER
  8    set totalTripMade = nvl(totalTripMade,0) + (select count(t#) from
trip where l#=:new.l#)
  9    where l#=:new.l#;
 10  end;
 11  /

Trigger created.

SQL> show error
No errors.
SQL>
SQL> --
SQL> select * from driver;

    E#         L# STATUS     TOTALTRIPMADE
------ ---------- ---------- -------------
     1      10001 AVAILABLE
     3      10002 AVAILABLE
     5      10003 ON LEAVE
     7      20002 BUSY
     9      30005 BUSY
    11      20005 ON LEAVE
```

```
    13      10008 ON LEAVE
   777        666 AVAILABLE

8 rows selected.

SQL>
SQL> insert into trip values(109, 10003, 'QRT834', sysdate);

1 row created.

SQL> insert into trip values(110, 20005, 'KKK007', sysdate);

1 row created.

SQL> insert into trip values(111, 20005, 'SST005', sysdate);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from driver;

    E#         L# STATUS     TOTALTRIPMADE
------ ---------- ---------- -------------
     1      10001 AVAILABLE
     3      10002 AVAILABLE
     5      10003 ON LEAVE              19
     7      20002 BUSY
     9      30005 BUSY
    11      20005 ON LEAVE              31
    13      10008 ON LEAVE
   777        666 AVAILABLE

8 rows selected.

SQL> insert into trip values(111, 20005, 'SYF777', sysdate);
insert into trip values(111, 20005, 'SYF777', sysdate)
*
ERROR at line 1:
ORA-00001: unique constraint (CSCI235.TRIP_PKEY) violated


SQL> commit;

Commit complete.

SQL> select * from driver;

    E#         L# STATUS     TOTALTRIPMADE
------ ---------- ---------- -------------
```

```
     1      10001 AVAILABLE
     3      10002 AVAILABLE
     5      10003 ON LEAVE              19
     7      20002 BUSY
     9      30005 BUSY
    11      20005 ON LEAVE              31
    13      10008 ON LEAVE
   777        666 AVAILABLE

8 rows selected.

SQL>
SQL> --
SQL> spool off
```

Task 3 (3.0 marks)
Stored PL/SQL function

Implement a stored PL/SQL function LONGTRIP(DLNUM) that finds the length (the total number of legs) of the longest trip performed by a driver identified by a driving license number (L# attribute in table DRIVER and parameter DLNUM parameter in the function). Remember to include or consider the drivers that performed no trips; that is to say, do not ignore drivers that do not perform any trip. If a driver has not performed any trip, then output the longest trip a 0.

Use a stored function LONGTRIP in SELECT statement to list the names of all drivers together with the length of the longest trip performed by each driver.

Deliverables

Hand in your solution3.lst in pdf format. Your report MUST have no errors and the report MUST list all SQL statements processed. All SQL statements of the script must be executed with SET ECHO ON, SET FEEEDBACK ON and SET SERVEROUTPUT ON options of SQL*Plus. It is a good idea to set the options at the beginning of the script. The script must be implemented with SQL*Plus. Oracle database server used for the testing and the final execution of the script is up to you. **The printouts that do not satisfy the requirements listed above will score NO MARKS!**

```
SQL> set echo on
SQL> set feedback on
SQL> set line 132
SQL> set pagesize 100
SQL> set serveroutput on
SQL> --
SQL> CREATE OR REPLACE FUNCTION LONGTRIP( DLNUM NUMBER) RETURN NUMBER IS
  2  aLongTrip NUMBER;
  3  BEGIN
  4    SELECT NVL(MAX(COUNT(*)),0)
  5    INTO aLongTrip
  6    FROM TRIPLEG right outer JOIN TRIP
  7    ON TRIPLEG.T# = TRIP.T#
  8    WHERE TRIP.L# = DLNUM
  9    GROUP BY TRIP.T#;
 10    RETURN(aLongTrip);
 11  END LONGTRIP;
 12  /

Function created.

SQL>
SQL> SHOW ERRORS;
No errors.
SQL>
```

```
SQL> SELECT L#, LONGTRIP(L#)
  2  FROM DRIVER;

        L# LONGTRIP(L#)
---------- ------------
       666            0
     10001            5
     10002            5
     10003            8
     10008            0
     20002            5
     20005            4
     30005            5

8 rows selected.

SQL>
SQL> --
SQL> Spool off
```

Submissions

This assignment is due by 9:00 pm (2100 hours) Sunday, 10 May 2020, Singapore time.

Submit the files **solution1.pdf**, **solution2.pdf, solution3.pdf** through Moodle in the following way:

1) Zip all the files (Solution1.pdf, solution2.pdf and solution3.pdf into one zipped folder. Name your zipped file as YourName-A3)
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 (SP220) 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