Islamic University of Technology

RDBMS

CSE 4508

Lab Report 8

Name : Abdullah

Student ID : 200041126

Section : 1

Lab Group : 1B

Date of Performance : 01/11/23

Date of Submission : 02/11/23

Task 1

In Task 1, I've set up an automatic system to make it easier for us to add user information into our database. This system takes care of generating a user ID and inserting it right after we put in a user's name.

To create the user IDs in a neat order, I've used a function that keeps track of the numbers one after the other, making sure they're all unique.

The trigger I've made ensures that the user ID is created and inserted before we add any other details about the user to our database.

Then I inserted data from the helper file and generated the same output.

Table creation and PL/SQL procedure:

```
-- TASK 1 PREREQUISITE
drop table UserInfo;
create table UserInfo(
    userId int,
    name varchar(10)
);

INSERT INTO UserInfo (name) VALUES ('monir');
INSERT INTO UserInfo (name) VALUES ('shabnur');
INSERT INTO UserInfo (name) VALUES ('nargis');
INSERT INTO UserInfo (name) VALUES ('jalil');

select * from UserInfo;

--Ids should be 1 to 4
```

```
-- TASK 1
DROP SEQUENCE UserInfoIdSequence;
CREATE SEQUENCE UserInfoIdSequence
INCREMENT BY 1
START WITH 1
MINVALUE 1
MAXVALUE 999999;

CREATE OR REPLACE TRIGGER InsertUserID
BEFORE INSERT ON UserInfo
FOR EACH ROW
WHEN (NEW.userId IS NULL)
BEGIN
   SELECT UserInfoIdSequence.NEXTVAL INTO :NEW.userId FROM DUAL;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error Message: ' || SQLERRM);
END InsertUserID;
/

SET SERVEROUTPUT ON;
```

Result:

```
SQL> INSERT INTO UserInfo (name) VALUES ('monir');

1 row created.

SQL> INSERT INTO UserInfo (name) VALUES ('shabnur');

1 row created.

SQL> INSERT INTO UserInfo (name) VALUES ('nargis');

1 row created.

SQL> INSERT INTO UserInfo (name) VALUES ('jalil');

1 row created.

SQL>
SQL> select * from UserInfo;

    USERID NAME
---------- ----------
         1 monir
         2 shabnur
         3 nargis
         4 jalil

SQL>
```

Task 2

In this task, I have constructed a table named "FrndList" using a procedure called "makeFriends." Additionally, I've developed another procedure designed to identify and filter out invalid user IDs. The valid user IDs are subsequently inserted into a designated result table.

The "makeFriends" procedure was primarily designed for the straightforward task of inserting data into the "FrndList" table.

Furthermore, I have incorporated error-handling capabilities within the procedures to provide clear and informative error messages in the event of incorrect input.

Then I inserted data from helper file and generated output.

Below is the implementation of the procedure :

```
-- TASK 2 PREREQUISITE
drop table FrndList;
create table FrndList(
    userId1 int,
    userId2 int
);

EXECUTE makeFriends(1,2);
EXECUTE makeFriends(8,2);
EXECUTE makeFriends(1,13);
EXECUTE makeFriends(3,2);
EXECUTE makeFriends(2,2);
EXECUTE makeFriends(1,3);
EXECUTE makeFriends(17,17);
EXECUTE makeFriends(4,2);

select * from FrndList;

--should show 8 rows
```

```
-- TASK 2
CREATE OR REPLACE PROCEDURE makeFriends
(fid1 IN INTEGER, fid2 IN INTEGER)
IS
    InvalidUserIDExeception EXCEPTION;
    PRAGMA EXCEPTION_INIT(InvalidUserIDExeception, -1718);
BEGIN
    INSERT INTO FrndList (userId1, userId2) VALUES (fid1, fid2);
    INSERT INTO FrndList (userId1, userId2) VALUES (fid2, fid1);
EXCEPTION
     WHEN InvalidUserIDExeception THEN
        DBMS_OUTPUT.PUT_LINE('Error: ORA-01718 - Invalid user IDs.');
END makeFriends;
/
```

```
CREATE OR REPLACE TRIGGER VerifyUserID
BEFORE INSERT ON FrndList
FOR EACH ROW
BEGIN
  DECLARE
    inputUserID1Exists NUMBER;
    inputUserID2Exists NUMBER;

    InvalidUserIDExeception EXCEPTION;
    PRAGMA EXCEPTION_INIT(InvalidUserIDExeception, -1718);
  BEGIN
    SELECT COUNT(*) INTO inputUserID1Exists
    FROM UserInfo
    WHERE userId = :NEW.userId1;

    SELECT COUNT(*) INTO inputUserID2Exists
    FROM UserInfo
    WHERE userId = :NEW.userId2;

    IF inputUserID1Exists = 0 OR inputUserID2Exists = 0 OR :NEW.userId1 = :NEW.userId2 THEN
      RAISE InvalidUserIDExeception;
    END IF;
  END;
END;
/

SET SERVEROUTPUT ON;
```

Result:

```
SQL>
SQL> EXECUTE makeFriends(1,2);

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(8,2);
Error: ORA-01718 - Invalid user IDs.

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(1,13);
Error: ORA-01718 - Invalid user IDs.

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(3,2);

PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE makeFriends(2,2);
Error: ORA-01718 - Invalid user IDs.

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(1,3);

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(17,17);
Error: ORA-01718 - Invalid user IDs.

PL/SQL procedure successfully completed.

SQL> EXECUTE makeFriends(4,2);

PL/SQL procedure successfully completed.

SQL>
SQL> select * from FrndList;

   USERID1    USERID2
---------- ----------
         1          2
         2          1
         3          2
         2          3
         1          3
         3          1
         4          2
         2          4

8 rows selected.
```

Task 3

In this task, I've created a table called "Message" and set up a procedure called "VerifyUserIDPair." This procedure checks if two people who want to send messages to each other are actually friends.

If someone tries to use the procedure with the wrong input, it will generate an error message to let them know there's a problem.

Then I inserted data from helper file and generated output.

Below is the implementation of the procedure :

```
-- TASK 3
CREATE OR REPLACE TRIGGER VerifyUserIDPair
BEFORE INSERT ON Message
FOR EACH ROW
BEGIN
   DECLARE
      inputUserIDPairExists NUMBER;
   BEGIN
      SELECT COUNT(*) INTO inputUserIDPairExists
      FROM FrndList
      WHERE (userId1 = :NEW.senderId AND userId2 = :NEW.recieverId)
      OR (userId2 = :NEW.senderId AND userId1 = :NEW.recieverId);

      IF inputUserIDPairExists = 0 OR :NEW.senderId = :NEW.recieverId THEN
         RAISE_APPLICATION_ERROR (-00123, 'Error: ORA-00123 - User IDs not friend.');
      END IF;
   END;
END;
/

SET SERVEROUTPUT ON;
```

```
-- TASK 3 PREREQUISITE
drop table Message;
create table Message(
senderId int,
recieverId int,
text varchar(56)
);

INSERT INTO Message (senderId,recieverId,text) VALUES (13,2,'Should raise error 2');
INSERT INTO Message (senderId,recieverId,text) VALUES (4,4,'Should raise error 3');
INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'Hi');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Hello');
INSERT INTO Message (senderId,recieverId,text) VALUES (4,1,'Should raise error 4');
INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'How are you ?');
INSERT INTO Message (senderId,recieverId,text) VALUES (4,2,'Give me the solution of todays task.');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Fine.');
INSERT INTO Message (senderId,recieverId,text) VALUES (2,1,'Jalil is asking for task again. Should I give it to him?');
INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'Do you have a girlfriend?');
INSERT INTO Message (senderId,recieverId,text) VALUES (3,4,'Should raise error 5');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,2,'No.');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,4,'Should raise error 6');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Yes, her name is Shabnur.');

select * from Message;

--should show only 9 messages
```

Result:

```
SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (13,2,'Should raise error 2');
INSERT INTO Message (senderId,recieverId,text) VALUES (13,2,'Should raise error 2')
            *
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -123 is out of
range
ORA-06512: at "MYUSER.VERIFYUSERIDPAIR", line 11
ORA-04088: error during execution of trigger 'MYUSER.VERIFYUSERIDPAIR'


SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (4,4,'Should raise error 3');
INSERT INTO Message (senderId,recieverId,text) VALUES (4,4,'Should raise error 3')
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -123 is out of
range
ORA-06512: at "MYUSER.VERIFYUSERIDPAIR", line 11
ORA-04088: error during execution of trigger 'MYUSER.VERIFYUSERIDPAIR'


SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'Hi');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Hello');

1 row created.
```

```
SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (4,1,'Should raise error 4');
INSERT INTO Message (senderId,recieverId,text) VALUES (4,1,'Should raise error 4')           *
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -123 is out of
range
ORA-06512: at "MYUSER.VERIFYUSERIDPAIR", line 11
ORA-04088: error during execution of trigger 'MYUSER.VERIFYUSERIDPAIR'


SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'How are you ?');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (4,2,'Give me the solution of todays task.');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Fine.');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (2,1,'Jalil is asking for task again. Should I give it to him?');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (3,1,'Do you have a girlfriend?');

1 row created.
```

```
SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (3,4,'Should raise error 5');
INSERT INTO Message (senderId,recieverId,text) VALUES (3,4,'Should raise error 5')            *
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -123 is out of
range
ORA-06512: at "MYUSER.VERIFYUSERIDPAIR", line 11
ORA-04088: error during execution of trigger 'MYUSER.VERIFYUSERIDPAIR'


SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (1,2,'No.');

1 row created.

SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (1,4,'Should raise error 6');
INSERT INTO Message (senderId,recieverId,text) VALUES (1,4,'Should raise error 6')            *
ERROR at line 1:
ORA-21000: error number argument to raise_application_error of -123 is out of
range
ORA-06512: at "MYUSER.VERIFYUSERIDPAIR", line 11
ORA-04088: error during execution of trigger 'MYUSER.VERIFYUSERIDPAIR'


SQL> INSERT INTO Message (senderId,recieverId,text) VALUES (1,3,'Yes, her name is Shabnur.');

1 row created.

SQL>
SQL> select * from Message;

  SENDERID RECIEVERID TEXT
---------- ---------- ------------------------------------------------------------
         3          1 Hi
         1          3 Hello
         3          1 How are you ?
         4          2 Give me the solution of todays task.
         1          3 Fine.
         2          1 Jalil is asking for task again. Should I give it to him?
         3          1 Do you have a girlfriend?
         1          2 No.
         1          3 Yes, her name is Shabnur.

9 rows selected.
```

Task 4

In this particular task, I've developed a procedure responsible for deleting messages from the message table. Before removing a message, the procedure performs a check to ensure that the provided user IDs are indeed friends.

Also I created a trigger for a correct deletion.

Then I inserted data from helper file and generated output.

In the created procedure, I can generate error message for wrong input.

Below is the implementation of the procedure :

```
-- TASK 4 PREREQUISITE
Execute unfriend(1,13);
Execute unfriend(1,3);

select * from Message;

--after executing unfriend(1,3) Message should only contain conversation between (4,2),(2,1),(1,2)
```

```
-- TASK 4
CREATE OR REPLACE PROCEDURE unfriend
(fid1 IN INTEGER, fid2 IN INTEGER)
IS
    inputUserIDPairExists NUMBER;

    NoRecordExeception EXCEPTION;
    PRAGMA EXCEPTION_INIT(NoRecordExeception, -00009);
BEGIN
    SELECT COUNT(*) INTO inputUserIDPairExists
    FROM FrndList
    WHERE (userId1 = fid1 AND userId2 = fid2)
    OR (userId2 = fid1 AND userId1 = fid2);

    IF inputUserIDPairExists = 0 OR fid1 = fid2 THEN
        RAISE NoRecordExeception;
    END IF;

    DELETE FROM FrndList
    WHERE (userId1 = fid1 AND userId2 = fid2)
    OR (userId2 = fid1 AND userId1 = fid2);
EXCEPTION
      WHEN NoRecordExeception THEN
          DBMS_OUTPUT.PUT_LINE('Error: ORA-00009 - User IDs not friend.');
END unfriend;
/

CREATE OR REPLACE TRIGGER DeleteMessages
BEFORE DELETE ON FrndList
FOR EACH ROW
BEGIN
    DELETE FROM Message
    WHERE (senderId = :OLD.userId1 AND recieverId = :OLD.userId2)
    OR (senderId = :OLD.userId2 AND recieverId = :OLD.userId1);
END;
/
```

Result:

```
SQL> Execute unfriend(1,13);
Error: ORA-00009 - User IDs not friend.

PL/SQL procedure successfully completed.

SQL> Execute unfriend(1,3);

PL/SQL procedure successfully completed.

SQL>
SQL> select * from Message;

  SENDERID RECIEVERID TEXT
---------- ---------- ------------------------------------------------------------
         4          2 Give me the solution of todays task.
         2          1 Jalil is asking for task again. Should I give it to him?
         1          2 No.
```

Task 5

To accomplish the task, I have created 'DeletePreviousPosts' trigger that is fired before an 'INSERT' operation on 'Post' table removing all record from 'PostNotification' table associated with 'userId' value in the 'INSERT' statement and 'UpdatePostNotification' trigger that is fired after an 'INSERT' operation on 'Post' inserting new record with given 'postId' for each of the 'frndId' saved in 'FrndList' table associated with 'userId' value in the 'INSERT' statement.

Below is the implementation of the procedure :

```
-- TASK 5 PREREQUISITE
drop table PostNotification;
drop table Post;

create table Post(
postId int PRIMARY key,
userId int,
text varchar(56)
);

create table PostNotification(
userId int,
frndId int,
postId int,
CONSTRAINT FK_postId FOREIGN KEY (postId) REFERENCES Post(postId)
);

Insert into Post (postId,userId,text) values (1,4,'I am lonely.');

select * from PostNotification;

-- should show one row with post id as 1

Insert into Post (postId,userId,text) values (2,4,'I need friends.');

select * from PostNotification;

-- should show one row with post id as 2
```

```
-- TASK 5
CREATE OR REPLACE TRIGGER DeletePreviousPosts
BEFORE INSERT ON Post
FOR EACH ROW
BEGIN
    DELETE FROM PostNotification
    WHERE (userId = :NEW.userId);
END;
/

CREATE OR REPLACE TRIGGER UpdatePostNotification
AFTER INSERT ON Post
FOR EACH ROW
BEGIN
    FOR frndRcrd IN (SELECT userId2 FROM FrndList WHERE userId1 = :NEW.userId)
    LOOP
        INSERT INTO PostNotification (userId, frndId, postId) VALUES (:NEW.userId, frndRcrd.userId2, :NEW.postId);
    END LOOP;
END;
/

SET SERVEROUTPUT ON;
```

Result:

```
SQL> Insert into Post (postId,userId,text) values (1,4,'I am lonely.');

1 row created.

SQL>
SQL> select * from PostNotification;

    USERID     FRNDID     POSTID
---------- ---------- ----------
         4          2          1

SQL> Insert into Post (postId,userId,text) values (2,4,'I need friends.');

1 row created.

SQL>
SQL> select * from PostNotification;

    USERID     FRNDID     POSTID
---------- ---------- ----------
         4          2          2

SQL>
```