Intro to Transactions

Tutorial Questions

1. What is a Transaction? Transaction is a logical unit of work

2. What does Atomicity mean? Atomicity means that a transaction is indivisible

3. What does a 'Commit' statement do? Commit statement is used to save all changes made during a transaction

4. What does 'Rollback' statement do? Rollback statement is used to undo all the changes made during a transaction

5. What is meant by Consistency and 'a consistent database state'? Consistency means that a transaction brings the database from one valid state to another. A consistent database state is a state where all integrity constraints are met, and database accurately reflects the real-word scenario its model

6. What two events cause a Commit to occur?

   When a Commit is executed by the user or when the system executes DDL

7. What two events cause a Rollback to occur?

   When a rollback is executed by the user, errors that cause the trasaction to fails

8. Consider this Stored Procedure.

```
1   DECLARE
2     TYPE my_refcur_type IS REF CURSOR;
3     rv_refcur MY_REFCUR_TYPE;
4     emp_details employee%ROWTYPE;
5   BEGIN
6     OPEN rv_refcur FOR SELECT * FROM employee;
7     LOOP FETCH rv_refcur INTO emp_details;
8       EXIT WHEN rv_refcur%NOTFOUND ;
9       DBMS_OUTPUT.PUT_LINE(emp_details.empname);
10    END LOOP;
11    CLOSE rv_refcur ;
12  END;
```

   a) In regard to line 2, what is **`my_refcur_type?`** user-defined cursor data types
   b) In regard to line 3, what is **`rv_refcur?`** is a cursor variable of the type my_refcur_type
   c) In regard to line 4, what is **`emp_details?`** variable with a type of the employee columns in the table
   d) Describe what occurs as lines 5-12 are executed Line 6: Opens the cursor rv_refcur and associates it with the result set of the SELECT * FROM employee query.
   Lines 7-10: Fetches each row of the result set into emp_details, outputs the employee name (empname) to the console, and repeats this until there are no more rows
   .Line 11: Closes the cursor rv_refcur.

9. Imagine that the following change is made:
   - Lines 2 & 3 are replaced with:
     `rv_refcur SYS_REFCURSOR;`

   a) What difference would the change make? the code used a predefined cursor of Oracle
   b) What is SYS_REFCURSOR? built-in cursor of Oracle

10. In PL/SQL, what is a package? A package in PL/SQL is a schema object that groups logically related PL/SQL types, items, and subprograms

11. What is the difference between a package specification and a package body?

12. What is the difference between a package specification and a package body?

    The package specification declares the elements (variables, types, procedures, functions) that are available for use outside the package. The package body contains the actual implementation of these elements.

13. Consider the code below.

```
CREATE OR REPLACE PACKAGE
               pckg_get_empdetails AS
TYPE rv_refcur IS REF CURSOR;
  PROCEDURE GetAllNames(
      pRefCur OUT rv_RefCur);
  END pckg_get_empdetails;
```

a) Is this a package spec or a package body? package specification

b) What is the name of the package? pckg_get_empdetails

c) How many procedures are defined in the code? 1

d) How many parameters in GetAllNames? 1

e) For each parameters in GetAllNames, specify the parameter name and the data type. pRefCur rv_RefCur

f) What does the keyword OUT mean? The parameter is an output parameter

g) Is the code for the procedure GetAllNames defined in the package specification?
NO

14. Consider the code below.

```
CREATE OR REPLACE PACKAGE BODY
               pckg_get_empdetails AS
  PROCEDURE GetAllNames(pRefCur OUT rv_RefCur) AS
  BEGIN
    OPEN pRefCur FOR SELECT empname FROM employee;
  END GetAllNames;
END pckg_get_empdetails;
```

a) Is this a package spec or a package body? package body

b) What is the name of the package? pckg_get_empdetails

c) Is the code for the procedure GetAllNames defined in the package specification? The declaration is in the package specification, but the actual code (implementation) is in the package body

d) What is the purpose of the code within the GetAllNames procedure?
The procedure GetAllNames opens a cursor pRefCur for selecting all employee names (empname) from the employee table

15. Consider the **anonymous block** below.

```
DECLARE
aRefCur SYS_REFCURSOR;
vEmpName employee.empname%TYPE;
BEGIN
    pckg_get_empdetails.GetAllNames(aRefCur);
    LOOP
      EXIT WHEN aRefCur%NOTFOUND;
      FETCH aRefCur INTO vEmpName;
      DBMS_OUTPUT.PUT_LINE(vEmpName);
    END LOOP;
END;
```

a) What is the datatype of aRefCur? SYS_REFCURSOR

b) What is the datatype of vEmpName? It is the data type of the empname column in the employee table

c) What is the name of the package called on line 5 pckg_get_empdetails

d) What is the name of the procedure called on line 5 GetAllNames

e) What happens to aRefCur after line 5 is executed? aRefCur is assigned a cursor that retrieves all employee names from the employee table

f) Describe what occurs as lines 6-10 are executed. A loop begins to fetch each employee name into vEmpName. If there are no more rows, the loop exits; otherwise, it prints each employee name

16. Consider the **VB code** below.

```
1.  Sub ShowAllEmployeesPack()
2.      Try
3.          Dim connOracle As Oracle.DataAccess.Client.OracleConnection
4.          Dim commOracle As New Oracle.DataAccess.Client.OracleCommand
5.          Dim paramOracle As Oracle.DataAccess.Client.OracleParameter
6.          connOracle = NewConnection()
7.          commOracle.Connection = connOracle
8.          commOracle.CommandType = CommandType.StoredProcedure
9.          commOracle.CommandText = "pckg_get_empdetails.GetAllNames"
10.
11.         paramOracle = New Oracle.DataAccess.Client.OracleParameter
12.         paramOracle.ParameterName = "pReturnValue"
13.         paramOracle.OracleDbType = Oracle.DataAccess.Client.OracleDbType.RefCursor
14.         paramOracle.Direction = ParameterDirection.Output
15.         commOracle.Parameters.Add(paramOracle)
16.         connOracle.Open()
17.
18.         Dim readerOracle As Oracle.DataAccess.Client.OracleDataReader
19.         readerOracle = commOracle.ExecuteReader()
20.         If readerOracle.HasRows = True Then
21.             Output_TextBox.Text = ""
22.             Do While readerOracle.Read()
23.                MsgBox("Name: " & readerOracle("empname")  )
24.             Loop
25.         Else
26.            MessageBox.Show("No rows found")
27.         End If
28.         connOracle.Close()
29.
30.     Catch ex As Exception
31.         MessageBox.Show(ex.Message)
32.     End Try
```

a) Describe the purpose of line 8 & 9 Set the command type to stored procedure and specify the name of the stored procedure to be executed

b) Describe the purpose of line 13 & 14 Set the parameter type to RefCursor and direction to Output to retrieve a result set from the stored procedure.

c) Describe the purpose of line 18 & 19 Create an OracleDataReader to read the result set returned by the stored procedure.

d) Describe the purpose of line 20 to 28

Check if the reader has rows. If yes, it loops through each row, displaying the employee name; if no rows are found, it shows a message stating "No rows found." Finally, it closes the database connection.

Lab Tasks:

1. If you have never created a VB application before, work through the VB_HELLO_WORLD documents available on Blackboard under VB Tutorials / Help / Demos
2. If you have never created a VB application that connects to Oracle before, work through the VB_HELLO_ORACLE documents available on Blackboard under VB Tutorials / Help / Demos
3. Execute the following code in SQL Developer

```
CREATE OR REPLACE FUNCTION SF_GETMSG RETURN VARCHAR2 AS
BEGIN
  RETURN 'Hello, wish you were here';
END;
```

4. Create a VB application that calls SF_GETMSG.

   You may want to use the code shown in the Button 2 example above.
   Test your by Clicking Button 2.
   If successful, your VB should display `Hello, wish you were here`

5. Execute the following code in SQL Developer

```
DROP TABLE CUST CASCADE CONSTRAINTS;
/
CREATE TABLE CUST (
CUSTID NUMBER PRIMARY KEY
, CUSTNAME VARCHAR2(100)
);
/
INSERT INTO CUST VALUES (100, 'Jon Jones');
INSERT INTO CUST VALUES (115, 'Evan Evans');
INSERT INTO CUST VALUES (123, 'Sue Smith');
```

6. Modify you VB application so that Buttons 1,3 & 4 from the above tutorial are implemented. Test of each of the buttons in turn to see if they work.

   Note: Button 1 deletes rows from the CUST table so outcomes for each of the other buttons may change. You may need to execute the INSERT statements again while testing.

Continue with assignment work