Intro to Transactions

Tutorial Questions

1. What is a Transaction?

⇨ **Transaction is a logical unit of word that comprises of several operations.**

2. What does Atomicity mean?

⇨ **It is one of ACID properties, it refers to the state that a transaction should be indivisible. Meaning that the transaction is either performed entirely or is not performed at all.**

3. What does a 'Commit' statement do?

⇨ **Commit permanently preserves changes to the database**

4. What does 'Rollback' statement do?

⇨ **Rollback reverses the effect of any uncommitted changes**

5. What is meant by Consistency and 'a consistent database state'?

⇨ **Consistency is the ability of a database to transform from one consistent state to another consistent state using various integrity constraints. A consistent database state, therefore, can be defined as a database that is entirely completed or entirely aborted.**

6. What two events cause a Commit to occur?

⇨ **Explicit Commit happens when the application issues an explicit SQL Commit command to the server. Implicit Commit happens as the application program terminates normally and the server commit the transaction.**

7. What two events cause a Rollback to occur?

⇨ **Explicit Rollback happens as the application issues an SQL Rollback command to the server. Implicit rollback happens as the application program terminates abnormally so the server rolls back the transaction.**

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?**

⇨ **IT is a weak ref cursor declare by the user. As the cursor does not return any value it is ,therefore, considered a weak ref cursor.**

b) In regard to line 3, what is **rv_refcur?**

⇨ **It is a Cursor variable of my_refcur_type**

c) In regard to line 4, what is **emp_details?**

⇨ **It is a record variable of the table Employee**

d) Describe what occurs as lines 5-12 are executed

⇨ **The program will stored all the data from employee into rv_refcur, after that emp_details will then be fetched with all the data from rv_refcur and the program will print out the name of the employee in the Employee table until there is nothing left.**

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?

⇨ **Declaring a weak ref cursor includes two parts. Firstly, we have to create the weak ref cursor type, and then we have to create a cursor variable. SYS_REFCURSOR saves us all the hard work as we only have to create a cursor variable and nothing else. Therefore, this change would be more convenient for us.**

b) What is SYS_REFCURSOR?

⇨ **SYS_REFCURSOR is basically a weak ref cursor and is predefined in the Oracle database software. Using this, we can minimize our work of creating a weak ref cursor.**

10. In PL/SQL, what is a package?

⇨ **It is a way of returning a ref cursor to an external application. It can be seen as an object that groups variable, constraints, types in SQL.**

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

⇨ **Package Specification contains the information about the package body, the package body contains the logic of SQL program.**

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

⇨ **Package Specification contains the information about the package body, the package body contains the logic of SQL program.**

13. Consider the code below.

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

a) Is this a package spec or a package body?
⇨ **It is a package specification as it contains the information about the package**

b) What is the name of the package?
⇨ **Pckg_get_empdetails**

c) How many procedures are defined in the code?
⇨ **One which is GetAllNames**

d) How many parameters in GetAllNames?
⇨ **One which is pRefCur**

e) For each parameters in GetAllNames, specify the parameter name and the data type.
⇨ **pRefCur – REF CURSOR**

f) What does the keyword OUT mean?
⇨ **It means the parameter is sending out data**

g) Is the code for the procedure GetAllNames defined in the package specification?
⇨ **No the specification does not contain logical part of the program.**

14. Consider the code below.

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

a) Is this a package spec or a package body?
⇨ **It contains the logic, so it is the 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?
⇨ **No, it is defined in the body**

d) What is the purpose of the code within the GetAllNames procedure?
=> **It selects one row from empname in employee table and write it into pRefCur.**

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

```
1   DECLARE
2   aRefCur SYS_REFCURSOR;
3   vEmpName employee.empname%TYPE;
4   BEGIN
5       pckg_get_empdetails.GetAllNames(aRefCur);
```

```
6       LOOP
7         EXIT WHEN aRefCur%NOTFOUND;
8         FETCH aRefCur INTO vEmpName;
9         DBMS_OUTPUT.PUT_LINE(vEmpName);
10      END LOOP;
11  END;
```

a) What is the datatype of aRefCur?

⇨ **It is a SYS_REFCURSOR which is basically a weak ref cursor**

b) What is the datatype of vEmpName?

⇨ **It is the same type as employee.empname which may be VARCHAR2**

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?

⇨ **It is passed to the procedure GetAllNames**

f) Describe what occurs as lines 6-10 are executed.

=> **Every row in aRefCur will be fetched into vEmpName and the program will print the name out until there is nothing left in aRefCur.**

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

⇨ **commOracle is a object that is responsible for facilitating running command on VB, in order to do that, we first set the CommandType property of the object to the suitable command. After that we change the CommandText property of the object to the procedure we want to call.**

b) Describe the purpose of line 13 & 14

⇨ **Line 13 allows us to obtain REF CURSOR data type as an OracleRefCursor. Line 14 is used to determine if the parameter is an input or output. In our case, the parameter is an output.**

c) Describe the purpose of line 18 & 19

⇨ **We create an object called readerOracle from the Oracle's library to enable data reading process. After reading the data, we will execute the data using ExecuteReader() method.**

d) Describe the purpose of line 20 to 28

⇨ **Line 20 check if there is we successfully fetched the table**

⇨ **Line 21 to Line 24 will be executed if there are rows fetched from the table, it will display the name of the employee and loop the process .**

⇨ **Line 25 to Line 26 will be executed I there is no row fetched from the table**

⇨ **Close the oracle connection after executing the commOracle's command.**

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