ITE 5220 Oracle Database Programming using PL/SQL

Lab Exercise 8[Chapter 7(Dynamic SQL)]

4 POINTS

Agenda:

To do this lab you will have to use your laptops.

You have to capture the output and write your findings about the output.

Practice: Using Native Dynamic SQL

In this practice, you create a package that uses Native Dynamic SQL to create or drop a table, and to populate, modify, and delete rows from the table. In addition, you create a package that compiles the PL/SQL code in your schema, either all the PL/SQL code or only code that has an `INVALID` status in the `USER_OBJECTS` table....

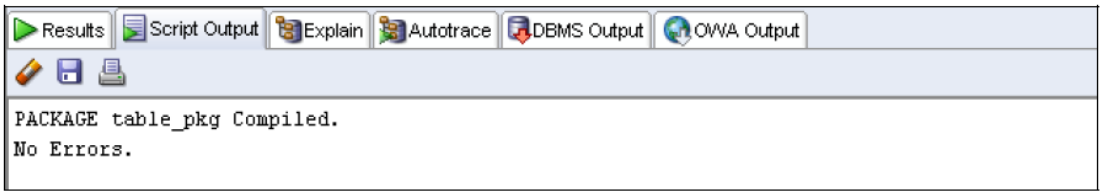
1) Create a package called `TABLE_PKG` that uses Native Dynamic SQL to create or drop a table, and to populate, modify, and delete rows from the table. The subprograms should manage optional default parameters with `NULL` values.

   a) Create a package specification with the following procedures:

```
PROCEDURE make(p_table_name VARCHAR2, p_col_specs VARCHAR2)
PROCEDURE add_row(p_table_name VARCHAR2, p_col_values
  VARCHAR2, p_cols VARCHAR2 := NULL)
PROCEDURE upd_row(p_table_name VARCHAR2, p_set_values
  VARCHAR2, p_conditions VARCHAR2 := NULL)
PROCEDURE del_row(p_table_name VARCHAR2,
    p_conditions VARCHAR2 := NULL);
PROCEDURE remove(p_table_name VARCHAR2)
```

Open the `sol_07_01_a.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to create the package specification. The result is shown below. To compile the package's specification, right-click the package's name in the Object Navigation tree, and then select Compile.

```
CREATE OR REPLACE PACKAGE table_pkg IS
  PROCEDURE make(p_table_name VARCHAR2, p_col_specs
      VARCHAR2);
  PROCEDURE add_row(p_table_name VARCHAR2, p_col_values
      VARCHAR2, p_cols VARCHAR2 := NULL);
  PROCEDURE upd_row(p_table_name VARCHAR2, p_set_values
      VARCHAR2, p_conditions VARCHAR2 := NULL);
  PROCEDURE del_row(p_table_name VARCHAR2, p_conditions
      VARCHAR2 := NULL);
  PROCEDURE remove(p_table_name VARCHAR2);
END table_pkg;
/
SHOW ERRORS
```



b) Create the package body that accepts the parameters and dynamically constructs the appropriate SQL statements that are executed using Native Dynamic SQL, except for the remove procedure. This procedure should be written using the DBMS_SQL package.

Open the `sol_07_01_b.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to create the package specification. The result is shown below. To compile the package's specification, right-click the package's name in the Object Navigation tree, and then select Compile.

```
CREATE OR REPLACE PACKAGE BODY table_pkg IS
  PROCEDURE execute(p_stmt VARCHAR2) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE(p_stmt);
    EXECUTE IMMEDIATE p_stmt;
  END;

  PROCEDURE make(p_table_name VARCHAR2, p_col_specs VARCHAR2)
  IS
    v_stmt VARCHAR2(200) := 'CREATE TABLE '|| p_table_name ||
                            ' (' || p_col_specs || ')';
  BEGIN
    execute(v_stmt);
  END;

  PROCEDURE add_row(p_table_name VARCHAR2, p_col_values
                    VARCHAR2, p_cols VARCHAR2 := NULL) IS
    v_stmt VARCHAR2(200) := 'INSERT INTO '|| p_table_name;
  BEGIN
    IF p_cols IS NOT NULL THEN
       v_stmt := v_stmt || ' (' || p_cols || ')';
    END IF;
    v_stmt := v_stmt || ' VALUES (' || p_col_values || ')';
    execute(v_stmt);
  END;

  PROCEDURE upd_row(p_table_name VARCHAR2, p_set_values
                   VARCHAR2, p_conditions VARCHAR2 := NULL) IS
```

```
    v_stmt VARCHAR2(200) := 'UPDATE '|| p_table_name || ' SET '
|| p_set_values;
  BEGIN
    IF p_conditions IS NOT NULL THEN
       v_stmt := v_stmt || ' WHERE ' || p_conditions;
    END IF;
    execute(v_stmt);
  END;

  PROCEDURE del_row(p_table_name VARCHAR2, p_conditions
                    VARCHAR2 := NULL) IS
    v_stmt VARCHAR2(200) := 'DELETE FROM '|| p_table_name;
  BEGIN
    IF p_conditions IS NOT NULL THEN
       v_stmt := v_stmt || ' WHERE ' || p_conditions;
    END IF;
    execute(v_stmt);
  END;

  PROCEDURE remove(p_table_name VARCHAR2) IS
    cur_id INTEGER;
    v_stmt VARCHAR2(100) := 'DROP TABLE '||p_table_name;
  BEGIN
    cur_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_OUTPUT.PUT_LINE(v_stmt);
    DBMS_SQL.PARSE(cur_id, v_stmt, DBMS_SQL.NATIVE);
    -- Parse executes DDL statements,no EXECUTE is required.
    DBMS_SQL.CLOSE_CURSOR(cur_id);
  END;

END table_pkg;
/
SHOW ERRORS
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
PACKAGE BODY table_pkg Compiled.
No Errors.
```

```
Messages - Log
TABLE_PKG Compiled
```

c) Execute the MAKE package procedure to create a table as follows:

```
make('my_contacts', 'id number(4), name
varchar2(40)');
```

Open the `sol_07_01_c.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to create the package specification. The code and the results are shown below. To compile the package's specification, right-click the package's name in the Object Navigation tree, and then select Compile.

```
EXECUTE table_pkg.make('my_contacts', 'id number(4), name
varchar2(40)')
```

```
Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

anonymous block completed
```

d) Describe the MY_CONTACTS table structure.

The code and the results are shown below.

```
MyDBConnection
1.34393167 seconds
Enter SQL Statement:
DESCRIBE my_contacts

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

DESCRIBE my_contacts
Name                          Null     Type
----------------------------- -------- ----------------------------------------------
ID                                     NUMBER(4)
NAME                                   VARCHAR2(40)

2 rows selected
```

e) Execute the `ADD_ROW` package procedure to add the following rows:

```
add_row('my_contacts','1,''Lauran Serhal''','id, name');
add_row('my_contacts','2,''Nancy''','id, name');
add_row('my_contacts','3,''Sunitha Patel''','id,name');
add_row('my_contacts','4,''Valli Pataballa''','id,name');
```

Open the `sol_07_01_e.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to execute the script. The result is shown below. To compile the package's specification, right-click the package's name in the Object Navigation tree, and then select Compile.

0.52349108 seconds

Enter SQL Statement:

```
BEGIN
  table_pkg.add_row('my_contacts','1,''Lauran Serhal''','id, name');
  table_pkg.add_row('my_contacts','2,''Nancy''','id, name');
  table_pkg.add_row('my_contacts','3,''Sunitha Patel''','id,name');
  table_pkg.add_row('my_contacts','4,''Valli Pataballa''','id,name');
END;
/
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
anonymous block completed
INSERT INTO my_contacts (id, name) VALUES (1,'Lauran Serhal')
INSERT INTO my_contacts (id, name) VALUES (2,'Nancy')
INSERT INTO my_contacts (id,name) VALUES (3,'Sunitha Patel')
INSERT INTO my_contacts (id,name) VALUES (4,'Valli Pataballa')
```

f) Query the MY_CONTACTS table contents to verify the additions.

The code and result are shown below.

Enter SQL Statement:

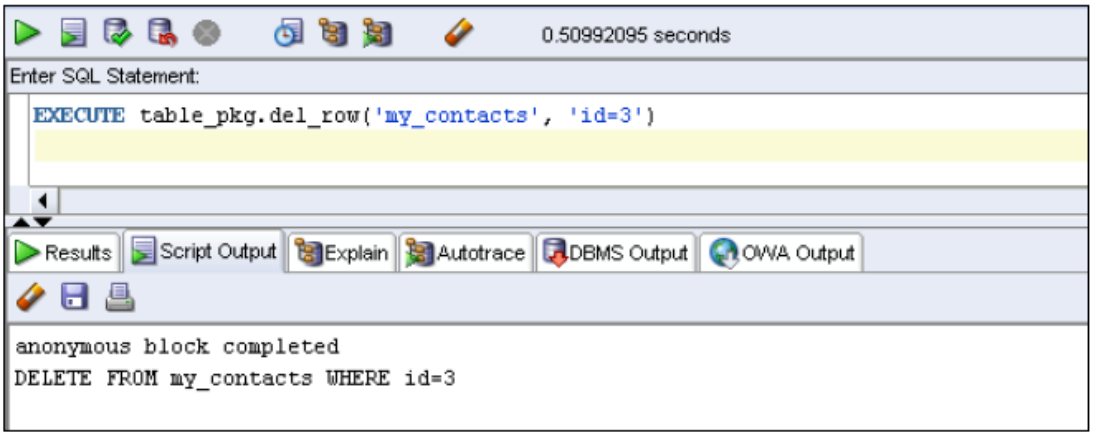
```
SELECT *
FROM my_contacts;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
ID                     NAME
---------------------- ----------------------------------------
1                      Lauran Serhal
2                      Nancy
3                      Sunitha Patel
4                      Valli Pataballa

4 rows selected
```

g) Execute the `DEL_ROW` package procedure to delete a contact with ID value `3`.

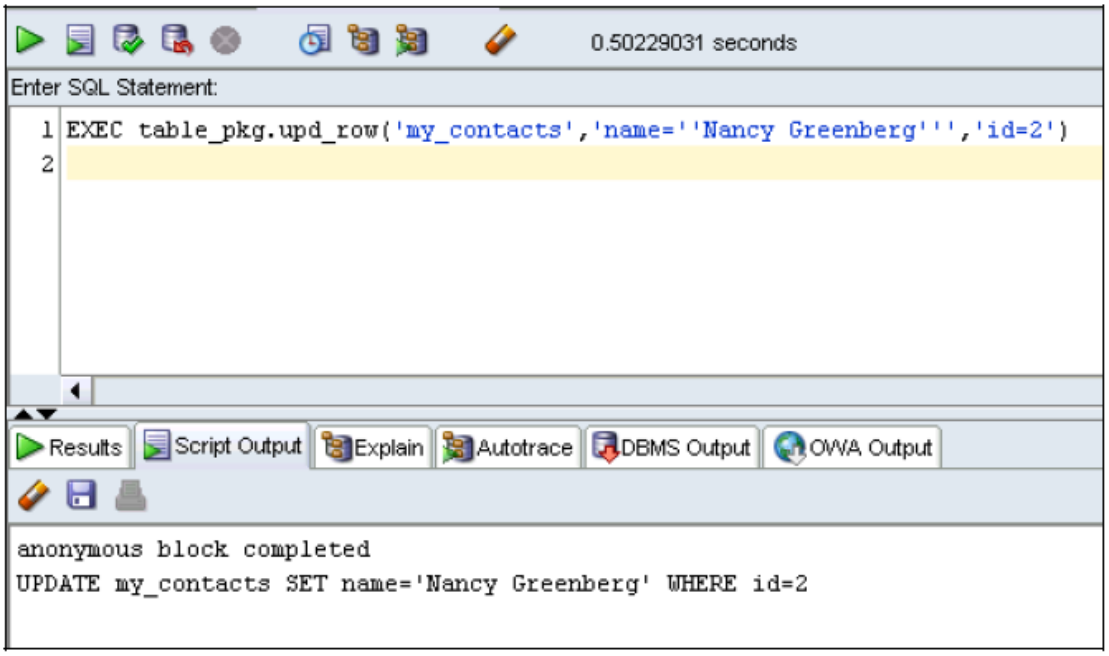
The code and result are shown below.



h) Execute the `UPD_ROW` procedure with the following row data:

```
upd_row('my_contacts','name=''Nancy Greenberg''','id=2');
```

The code and result are shown below.



i) Query the MY_CONTACTS table contents to verify the changes.

The code and result are shown below.

0.50275838 seconds

Enter SQL Statement:

```
1 SELECT *
2 FROM my_contacts;
3
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
ID                     NAME
---------------------- ----------------------------------------
1                      Lauran Serhal
2                      Nancy Greenberg
4                      Valli Pataballa

3 rows selected
```

j) Drop the table by using the remove procedure and describe the MY_CONTACTS table.

The code and result are shown below.

1.22356808 seconds

Enter SQL Statement:

```
EXECUTE table_pkg.remove('my_contacts')
DESCRIBE my_contacts
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

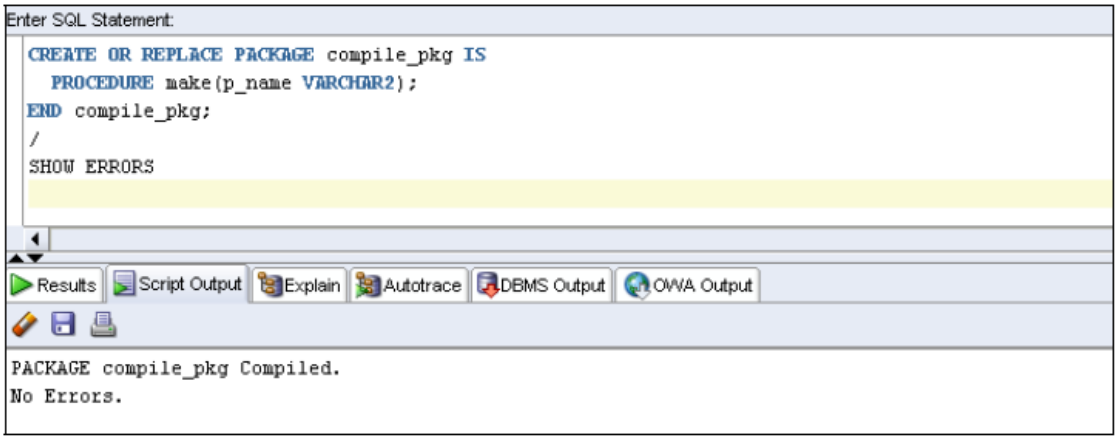
```
anonymous block completed
DROP TABLE my_contacts

DESCRIBE my_contacts
Name                           Null     Type
------------------------------ -------- -------------------------------------


0 rows selected
```

2) Create a COMPILE_PKG package that compiles the PL/SQL code in your schema.

a) In the specification, create a package procedure called MAKE that accepts the name of a PL/SQL program unit to be compiled.

Open the sol_07_02_a.sql file in the D:\labs\PLPU\solns folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to create the package specification. The code and the results are shown below. To compile the package's specification, right-click the package's name in the Object Navigation tree, and then select Compile.

```
CREATE OR REPLACE PACKAGE compile_pkg IS
  PROCEDURE make(p_name VARCHAR2);
END compile_pkg;
/
SHOW ERRORS
```



b) In the package body, include the following:

i) The EXECUTE procedure used in the TABLE_PKG procedure in step 1 of this practice.

ii) A private function named GET_TYPE to determine the PL/SQL object type from the data dictionary.

- The function returns the type name (use PACKAGE for a package with a body) if the object exists; otherwise, it should return a NULL.

- In the `WHERE` clause condition, add the following to the condition to ensure that only one row is returned if the name represents a `PACKAGE`, which may also have a `PACKAGE BODY`. In this case, you can only compile the complete package, but not the specification or body as separate components:

```
rownum = 1
```

iii) Create the `MAKE` procedure by using the following information:

- The `MAKE` procedure accepts one argument, `name`, which represents the object name.
- The `MAKE` procedure should call the `GET_TYPE` function. If the object exists, `MAKE` dynamically compiles it with the `ALTER` statement.

Open the `sol_07_02_b.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to create the package body. The code and the results are shown below. To compile the package's body, right-click the package's name or body in the Object Navigation tree, and then select Compile.

```oracle_plsql
CREATE OR REPLACE PACKAGE BODY compile_pkg IS

  PROCEDURE execute(p_stmt VARCHAR2) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE(p_stmt);
    EXECUTE IMMEDIATE p_stmt;
  END;

  FUNCTION get_type(p_name VARCHAR2) RETURN VARCHAR2 IS
    v_proc_type VARCHAR2(30) := NULL;
  BEGIN
    /*
     * The ROWNUM = 1 is added to the condition
     * to ensure only one row is returned if the
     * name represents a PACKAGE, which may also
     * have a PACKAGE BODY. In this case, we can
     * only compile the complete package, but not
     * the specification or body as separate
     * components.
     */
    SELECT object_type INTO v_proc_type
    FROM user_objects
    WHERE object_name = UPPER(p_name)
    AND ROWNUM = 1;
    RETURN v_proc_type;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN NULL;
  END;
```

```
  PROCEDURE make(p_name VARCHAR2) IS
    v_stmt       VARCHAR2(100);
    v_proc_type  VARCHAR2(30) := get_type(p_name);
  BEGIN
    IF v_proc_type IS NOT NULL THEN
      v_stmt := 'ALTER '|| v_proc_type ||' '|| p_name ||'
COMPILE';
      execute(v_stmt);
    ELSE
      RAISE_APPLICATION_ERROR(-20001,
         'Subprogram '''|| p_name ||''' does not exist');
    END IF;
  END make;
END compile_pkg;
/
SHOW ERRORS
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
PACKAGE BODY compile_pkg Compiled.
No Errors.
```

Messages - Log

```
COMPILE_PKG Body Compiled
```

c) Use the `COMPILE_PKG.MAKE` procedure to compile the following:

i) The `EMPLOYEE_REPORT` procedure

ii) The `EMP_PKG` package

iii) A nonexistent object called `EMP_DATA`

Open the `sol_07_02_c.sql` file in the `D:\labs\PLPU\solns` folder, or copy and paste the following code in the SQL Worksheet area. Click the Run Script (F5) icon on the SQL Worksheet toolbar to execute the package's procedure. The code and the results are shown below.

```
EXECUTE compile_pkg.make('employee_report')
EXECUTE compile_pkg.make('emp_pkg')
EXECUTE compile_pkg.make('emp_data')
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

```
anonymous block completed
ALTER PROCEDURE employee_report COMPILE

anonymous block completed
ALTER PACKAGE emp_pkg COMPILE


Error starting at line 3 in command:
EXECUTE compile_pkg.make('emp_data')
Error report:
ORA-20001: Subprogram 'emp_data' does not exist
ORA-06512: at "ORA61.COMPILE_PKG", line 39
ORA-06512: at line 1
```