Exercises for Chapter 17: Native Dynamic SQL

Try It Yourself

The projects in this section are meant to have you utilize all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

The Labs below provide you with exercises and suggested answers with discussion related to how those answers resulted. The most important thing to realize is whether your answer works. You should figure out the implications of the answers here and what the effects are from any different answers you may come up with.

1. Create a stored procedure based on the script ch20_1c.sql (version 3), created in first lab of this chapter. The procedure should accept two parameters to hold a table name and an ID, and return six parameters with first name, last name, street, city, state, and zip information.

   Answer: Your procedure should look similar to the procedure shown below. All changes are highlighted in bold.

```
CREATE OR REPLACE PROCEDURE get_name_address (table_name_in   IN VARCHAR2
                                             ,id_in            IN NUMBER
                                             ,first_name_out OUT VARCHAR2
                                             ,last_name_out  OUT VARCHAR2
                                             ,street_out     OUT VARCHAR2
                                             ,city_out       OUT VARCHAR2
                                             ,state_out      OUT VARCHAR2
                                             ,zip_out        OUT VARCHAR2)
AS
   sql_stmt VARCHAR2(200);
BEGIN
   sql_stmt := 'SELECT a.first_name, a.last_name, a.street_address'||
               '      ,b.city, b.state, b.zip'                     ||
               '  FROM '||table_name_in||' a, zipcode b'           ||
               ' WHERE a.zip = b.zip'                              ||
               '   AND '||table_name_in||'_id = :1';
   EXECUTE IMMEDIATE sql_stmt
   INTO first_name_out, last_name_out, street_out, city_out, state_out,
        zip_out
   USING id_in;
```

```
END get_name_address;
```

The procedure above contains two IN parameters whose values are used by the dynamic SQL statement, and six OUT parameters that hold date returned by the SELECT statement. Once created, the procedure can be tested with the following PL/SQL block:

```
SET SERVEROUTPUT ON
DECLARE
   v_table_name VARCHAR2(20) := '&sv_table_name';
   v_id NUMBER := &sv_id;
   v_first_name VARCHAR2(25);
   v_last_name VARCHAR2(25);
   v_street VARCHAR2(50);
   v_city VARCHAR2(25);
   v_state VARCHAR2(2);
   v_zip VARCHAR2(5);
BEGIN
   get_name_address (v_table_name, v_id, v_first_name, v_last_name,
                     v_street, v_city, v_state, v_zip);

   DBMS_OUTPUT.PUT_LINE ('First Name: '||v_first_name);
   DBMS_OUTPUT.PUT_LINE ('Last Name:  '||v_last_name);
   DBMS_OUTPUT.PUT_LINE ('Street:     '||v_street);
   DBMS_OUTPUT.PUT_LINE ('City:       '||v_city);
   DBMS_OUTPUT.PUT_LINE ('State:      '||v_state);
   DBMS_OUTPUT.PUT_LINE ('Zip Code:   '||v_zip);

END;
```

When run, this script produces the following output (the first run is against the STUDENT table, and the second run is against the INSTRUCTOR table):

```
Enter value for sv_table_name: student
old   2:    v_table_name VARCHAR2(20) := '&sv_table_name';
new   2:    v_table_name VARCHAR2(20) := 'student';
Enter value for sv_id: 105
old   3:    v_id NUMBER := &sv_id;
new   3:    v_id NUMBER := 105;
First Name: Angel
Last Name:  Moskowitz
Street:     320 John St.
City:       Ft. Lee
State:      NJ
Zip Code:   07024

PL/SQL procedure successfully completed.


Enter value for sv_table_name: instructor
old   2:    v_table_name VARCHAR2(20) := '&sv_table_name';
```

```
new   2:    v_table_name VARCHAR2(20) := 'instructor';
Enter value for sv_id: 105
old   3:    v_id NUMBER := &sv_id;
new   3:    v_id NUMBER := 105;
First Name: Anita
Last Name:  Morris
Street:     34 Maiden Lane
City:       New York
State:      NY
Zip Code:   10015

PL/SQL procedure successfully completed.
```

---

2. Modify procedure created in the previous exercise. Instead of using six parameters to hold name and address information, the procedure should return a user-defined record that contains six fields that hold name and address information. *Note: you may want to create a package where you define record type. This record may be used later, for example, when the procedure is invoked in a PL/SQL block.*

Answer: Your package should look similar to the package shown below. All changes to the procedure are highlighted in bold.

```
CREATE OR REPLACE PACKAGE dynamic_sql_pkg AS

   -- Create user-defined record type
   TYPE name_addr_rec_type IS RECORD
      (first_name VARCHAR2(25),
       last_name VARCHAR2(25),
       street VARCHAR2(50),
       city VARCHAR2(25),
       state VARCHAR2(2),
       zip VARCHAR2(5));

   PROCEDURE get_name_address (table_name_in  IN VARCHAR2
                              ,id_in          IN NUMBER
                              ,name_addr_rec OUT name_addr_rec_type);
END dynamic_sql_pkg;
/

CREATE OR REPLACE PACKAGE BODY dynamic_sql_pkg AS

PROCEDURE get_name_address (table_name_in  IN VARCHAR2
                           ,id_in          IN NUMBER
                           ,name_addr_rec OUT name_addr_rec_type)
IS
   sql_stmt VARCHAR2(200);
BEGIN
   sql_stmt := 'SELECT a.first_name, a.last_name, a.street_address'||
               '      ,b.city, b.state, b.zip'                    ||
               '  FROM '||table_name_in||' a, zipcode b'          ||
               ' WHERE a.zip = b.zip'                             ||
               '   AND '||table_name_in||'_id = :1';
   EXECUTE IMMEDIATE sql_stmt
   INTO name_addr_rec
```

```
   USING id_in;
END get_name_address;

END dynamic_sql_pkg;
/
```

In the package specification created above, you declare a user-defined record type. This record type is used by the procedure for its OUT parameter, `name_addr_rec`. Once the package is created, its procedure can be tested with the following PL/SQL block (changes are shown in bold):

```
SET SERVEROUTPUT ON
DECLARE
   v_table_name VARCHAR2(20) := '&sv_table_name';
   v_id NUMBER := &sv_id;
   name_addr_rec DYNAMIC_SQL_PKG.NAME_ADDR_REC_TYPE;
BEGIN
   dynamic_sql_pkg.get_name_address (v_table_name, v_id, name_addr_rec);

   DBMS_OUTPUT.PUT_LINE ('First Name: '||name_addr_rec.first_name);
   DBMS_OUTPUT.PUT_LINE ('Last Name:  '||name_addr_rec.last_name);
   DBMS_OUTPUT.PUT_LINE ('Street:     '||name_addr_rec.street);
   DBMS_OUTPUT.PUT_LINE ('City:       '||name_addr_rec.city);
   DBMS_OUTPUT.PUT_LINE ('State:      '||name_addr_rec.state);
   DBMS_OUTPUT.PUT_LINE ('Zip Code:   '||name_addr_rec.zip);

END;
```

Notice that instead of declaring six variables, you declare one variable of the user-defined record type, `name_addr_rec_type`. Because this record type has been defined in the package DYNAMIC_SQL_PKG, the name of the record type is prefixed by the name of the package. Similarly, the name of package has been added to the procedure call statement.

When run, this script produces the output shown below (the first output is against the STUDENT table, and the second output is against the INSTRUCTOR table):

```
Enter value for sv_table_name: student
old   2:    v_table_name VARCHAR2(20) := '&sv_table_name';
new   2:    v_table_name VARCHAR2(20) := 'student';
Enter value for sv_id: 105
old   3:    v_id NUMBER := &sv_id;
new   3:    v_id NUMBER := 105;
First Name: Angel
Last Name:  Moskowitz
Street:     320 John St.
City:       Ft. Lee
State:      NJ
Zip Code:   07024

PL/SQL procedure successfully completed.
```

```
Enter value for sv_table_name: instructor
old   2:    v_table_name VARCHAR2(20) := '&sv_table_name';
new   2:    v_table_name VARCHAR2(20) := 'instructor';
Enter value for sv_id: 105
old   3:    v_id NUMBER := &sv_id;
new   3:    v_id NUMBER := 105;
First Name: Anita
Last Name:  Morris
Street:     34 Maiden Lane
City:       New York
State:      NY
Zip Code:   10015

PL/SQL procedure successfully completed.
```