Exercises for Chapter 1: PL/SQL Concepts

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.

By the Way

Note that there is no Exercise section for Lab 1.2 PL/SQL Development Environment.

Lab 1.1 PL/SQL Architecture

Answer the following questions:

PL/SQL Architecture

a) Why it is more efficient to combine SQL statements into PL/SQL blocks?

Answer: It is more efficient to use SQL statements within PL/SQL blocks because network traffic can be decreased significantly, and an application becomes more efficient as well.
When an SQL statement is issued on the client computer, the request is made to the database on the server, and the result set is sent back to the client. As a result, a single SQL statement causes two trips on the network. If multiple SELECT statements are issued, the network traffic can increase significantly very quickly. For example, four SELECT statements cause eight network trips. If these statements are part of the PL/SQL block, there are still only two network trips made, as in the case of a single SELECT statement.

b) What are the differences between named and anonymous PL/SQL blocks?

Answer: Named PL/SQL blocks can be stored in the database and referenced later by their names. Since anonymous PL/SQL blocks do not have names, they cannot be stored in the database and referenced later.

PL/SQL Block Structure

For the next two questions, consider the following code:

```
DECLARE
   v_name  VARCHAR2(50);
   v_total NUMBER;
```

```
BEGIN
   SELECT i.first_name||' '||i.last_name, COUNT(*)
     INTO v_name, v_total
     FROM instructor i, section s
    WHERE i.instructor_id = s.instructor_id
      AND i.instructor_id = 102
   GROUP BY i.first_name||' '||i.last_name;

   DBMS_OUTPUT.PUT_LINE ('Instructor '||v_name||' teaches '||v_total||' courses');

EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such instructor');
END;
```

a) Based on the code example provided, describe the structure of a PL/SQL block.

Answer: PL/SQL blocks contain three sections: declaration section, executable section, and exception-handling section. The executable section is the only mandatory section of the PL/SQL block.
The declaration section holds definitions of PL/SQL identifiers such as variables, constants, and cursors. The declaration section starts with the keyword DECLARE. The declaration section

```
DECLARE
   v_name  VARCHAR2(50);
   v_total NUMBER;
```

contains definitions of two variables, v_name and v_total.
The executable section holds executable statements. It starts with the keyword BEGIN and ends with the keyword END. The executable section shown in bold letters

```
BEGIN
   SELECT i.first_name||' '||i.last_name, COUNT(*)
     INTO v_name, v_total
     FROM instructor i, section s
    WHERE i.instructor_id = s.instructor_id
      AND i.instructor_id = 102
   GROUP BY i.first_name||' '||i.last_name;

   DBMS_OUTPUT.PUT_LINE
      ('Instructor '||v_name||' teaches '||v_total||' courses');

EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such instructor');
END;
```

contains a SELECT INTO statement that assigns values to the variables v_name and v_total, and a DBMS_OUTPUT.PUT_LINE statement that displays their values on the screen.
The exception-handling section of the PL/SQL block contains statements that are executed only if runtime errors occur in the PL/SQL block. The following exception-handling section

```
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such instructor');
```

contains the `DBMS_OUTPUT.PUT_LINE` statement that is executed when runtime error `NO_DATA_FOUND` occurs.

b) What happens when runtime error `NO_DATA_FOUND` occurs in the PL/SQL block just shown?

Answer: When a runtime error occurs in the PL/SQL block, control is passed to the -exception-handling section of the block. The exception `NO_DATA_FOUND` is evaluated then with the help of the `WHEN` clause.
When the `SELECT INTO` statement

```
SELECT i.first_name||' '||i.last_name, COUNT(*)
 INTO v_name, v_total
 FROM instructor i, section s
WHERE i.instructor_id = s.instructor_id
  AND i.instructor_id = 102
GROUP BY i.first_name||' '||i.last_name;
```

does not return any rows, control of execution is passed to the exception-handling section of the block. Next, the `DBMS_OUTPUT.PUT_LINE` statement associated with the exception `NO_DATA_FOUND` is executed. As a result, the message "There is no such instructor" is displayed on the screen.

How PL/SQL Gets Executed

a) What happens when an anonymous PL/SQL block is executed?

Answer: When an anonymous PL/SQL block is executed, the code is sent to the PL/SQL engine on the server, where it is compiled.

b) What steps are included in the compilation process of a PL/SQL block?

Answer: The compilation process includes syntax checking, binding, and p-code generation.
Syntax checking involves checking PL/SQL code for compilation errors. Once syntax errors have been corrected, a storage address is assigned to the variables that are used to hold data for Oracle. This process is called binding. Next, p-code is generated for the PL/SQL block. P-code is a list of instructions to the PL/SQL engine. For named blocks, p-code is stored in the database, and it is used the next time the program is executed.

c) What is a syntax error?

Answer: A syntax error occurs when a statement does not correspond to the syntax rules of the programming language. An undefined variable or a misplaced keyword are examples of syntax error.

d) How does a syntax error differ from a runtime error?

Answer: A syntax error can be detected by the PL/SQL compiler. A runtime error occurs while the program is running and cannot be detected by the PL/SQL compiler.
A misspelled keyword is an example of the syntax error. For example, the script

```
BEIN
```

```
   DBMS_OUTPUT.PUT_LINE ('This is a test');
END;
```

contains a syntax error. You should try to find this error.
A `SELECT INTO` statement returning no rows is an example of a runtime error. This error can be handled with the help of the exception-handling section of the PL/SQL block.

Lab 1.3 PL/SQL: The Basics

Answer the following questions.

DBMS_OUTPUT.PUT_LINE Statement

a) What the `DBMS_OUTPUT.PUT_LINE` statement is used for?

   Answer: The `DBMS_OUTPUT.PUT_LINE` statement is used to display information on the screen.

b) Where does the `DBMS_OUTPUT.PUT_LINE` statement write its information prior to displaying it on the screen?

   Answer: First, the information is written into the buffer for storage. Once a program has been completed, the information from the buffer is displayed on the screen. The size of the buffer can be set between 2,000 and 1,000,000 bytes.

For the next two questions, create the following PL/SQL script:

For Example *ch01_3a.sql*

```
DECLARE
   v_num    NUMBER := 10;
   v_result NUMBER;
BEGIN
   v_result := POWER(v_num, 2);
END;
```

c) Modify the script above so that the value of the `v_result` variable is displayed on the screen.

   Answer: The script should be modified similar the following. Newly added `DBMS_OUTPUT.PUT_LINE` statement is shown in bold.

   For Example *ch01_3b.sql*

```
DECLARE
   v_num    NUMBER := 10;
   v_result NUMBER;
BEGIN
   v_result := POWER(v_num, 2);
   DBMS_OUTPUT.PUT_LINE ('The value of v_result is: '||v_result);
END;
```

Watch Out!

When executing the script in SQL Developer, you need to enable the Dbms Output window. To enable the DBMS_OUTPUT in SQL*Plus, you enter one of the following statement before the PL/SQL block:

```
SET SERVEROUTPUT ON;
```

Otherwise, you will not be able to see the results of the DBMS_OUTPUT.PUT_LINE statement on the screen!

d) What output is generated by the script?

Answer: The following output is shown in the Dbms Output window:

```
The value of v_result is: 100
```

Substitution Variable Feature

a) What are substitution variables?

Answer: Substitution variables are special type of variables that enable PL/SQL to accept input from a user at a run-time. Substitution variables cannot be used to output values because no memory is allocated for them. Substitution variables are replaced with the values provided by the user before the PL/SQL block is sent to the database.

b) How do you recognize substitution variable in a PL/SQL script?

Answer: Substitution variables are usually prefixed by the ampersand (&) character or double ampersand (&&) character.

c) Why is it considered a good practice to enclose substitution variables with single quotes for string datatypes?

Answer: A program cannot depend wholly on a user to provide text information in single quotes. Enclosing a substitution variable with single quotes allows a program to be less error-prone.

d) Modify script ch01_3b.sql created in the previous section so that it prompts a user to provide input value for the variable `v_num`.

Answer: The script should be modified similar the following. Newly added substitution variable is highlighted in bold.

For Example *ch01_3c.sql*

```
DECLARE
   v_num    NUMBER := &sv_num;
   v_result NUMBER;
BEGIN
   v_result := POWER(v_num, 2);
   DBMS_OUTPUT.PUT_LINE ('The value of v_result is: '||v_result);
END;
```

In this version of the script, the assignment statement in the declaration portion has been modified. Instead of initializing the variable `v_num` to 10, it is now initialized to a value provided at the run time. When this version of the script is executed, and the value of 5 is provided at the run time, it produces the output is as shown:

```
The value of v_result is: 25
```

By the Way?

When a substitution variable is used in a PL/SQL script, a user is prompted to enter the value for the variable regardless of the Dbms Output setting. The user prompt is provided by SQL Developer and SQL*Plus tools and does not depend on whether the Dbms Output setting has been enabled for a particular script execution.

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.

1) In order to calculate the area of a circle, the circle's radius must be squared and then multiplied by π. Write a program that calculates the area of a circle. The value for the radius should be provided with the help of a substitution variable. Use 3.14 for the value of π. Once the area of the circle is calculated, display it on the screen.

   Answer: The script should look similar to the following:

   For Example *ch01_4a.sql*

```
DECLARE
   v_radius NUMBER := &sv_radius;
   v_area   NUMBER;
BEGIN
   v_area := POWER(v_radius, 2) * 3.14;
   DBMS_OUTPUT.PUT_LINE ('The area of the circle is: '||v_area);
END;
```

   The declaration section of this script contains definitions of two variables, `v_radius` and `v_area`, to store the values for the radius of the circle and its area, respectively. Note that the value for variable `v_radius` is provided at the run via substitution variable `&sv_radius`.
   Next, the executable section computes the area of a circle and assigns its value to the variable `v_area`. This is accomplished with the help of the built-in function `POWER` and the value stored in the variable `v_radius`. Finally, it displays newly calculated value of the `v_area` on the screen.
   For example, if value of 5 is provided at the run time, then the script produces the following output:

```
The area of the circle is: 78.5
```

2) Create a script that displays the name and time of the day based on the system date. The sample output is provided below for your reference: `'Today is Monday at 10:49 AM'`

   Answer: The script should look similar to the following:

   For Example *ch01_5a.sql*

```
DECLARE
   v_date DATE := SYSDATE;
BEGIN
```

```
   DBMS_OUTPUT.PUT_LINE ('Today is '||to_char(v_date, 'fmDay')||
                         ' at '     ||to_char(v_date, 'hh:mi AM'));
END;
```

The declaration section of this script contains definition and initialization of the variable, `v_date`. The executable section contains a single `DBMS_OUTPUT.PUT_LINE` statement that construct the output message displayed on the screen. Note the usage of the `TO_CHAR` function. In the first instance, it is used to extract the name of the day, and in the second instance it is used to extract the time of the day. When run, the script produces the following output:

```
Today is Monday at 03:49 PM
```