Database Programming with PL/SQL

10-3: Advanced Package Concepts

Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

FORWARD DECLARATION	Defines subprograms in logical or alphabetical order, defines mutually recursive subprograms, and groups and logically organizes subprograms in a package body.
STANDARD	A package that defines the PL/SQL environment and globally declares types, exceptions, and subprograms that are available automatically to PL/SQL programs.
OVERLOADING	Enables you to develop two or more packaged subprograms with the same name.
INITIALIZATION BLOCK	An un-named block at the end of the package body that automatically executes once and is used to initialize public and private package variables.
BODILESS PACKAGE	A package that has a specification but no executable code containing only public variables.

Try It / Solve It

1. Create a package called overload. The package should contain three procedures all called what_am_i. The first procedure should accept a VARCHAR2 as an IN parameter, the second a NUMBER, and the third a DATE. Each procedure should display a simple message to show which datatype was passed to it. For example, the first procedure could display “Here I am a Varchar2.” Save your work for later. When you are done, describe the package.

```
CREATE OR REPLACE PACKAGE OVERLOAD AS
PROCEDURE what_am_i(p_parametro IN VARCHAR2);
```

```
PROCEDURE what_am_i(p_parametro IN NUMBER);
PROCEDURE what_am_i(p_parametro IN DATE);
END;

CREATE OR REPLACE PACKAGE BODY OVERLOAD AS
PROCEDURE what_am_i(p_parametro IN VARCHAR2) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Varchar2.');
END;
PROCEDURE what_am_i(p_parametro IN NUMBER) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Number.');
END;
PROCEDURE what_am_i(p_parametro IN DATE) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Date.');
END;
END;

DESCRIBE OVERLOAD;
```

2. Test the overload package by calling it and passing in a character string, a number, and then a date. You should see the different messages returned.

```
BEGIN
OVERLOAD.WHAT_AM_I('hola');
OVERLOAD.WHAT_AM_I(29);
```

```
OVERLOAD.WHAT_AM_I('29-Apr-2019');
END;
```

3. Now modify the overload package to have a fourth procedure in it, again called what_am_i, accepting two parameters: p_in NUMBER and p_in2 VARCHAR2. Test the new procedure to verify that it works.

```
CREATE OR REPLACE PACKAGE OVERLOAD AS
PROCEDURE what_am_i(p_parametro IN VARCHAR2);
PROCEDURE what_am_i(p_parametro IN NUMBER);
PROCEDURE what_am_i(p_parametro IN DATE);
PROCEDURE what_am_i(p_in IN NUMBER, p_in2 IN VARCHAR2);
END;

CREATE OR REPLACE PACKAGE BODY OVERLOAD AS
PROCEDURE what_am_i(p_parametro IN VARCHAR2) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Varchar2.');
END;
PROCEDURE what_am_i(p_parametro IN NUMBER) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Number.');
END;
PROCEDURE what_am_i(p_parametro IN DATE) AS
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Date.');
END;

PROCEDURE what_am_i(p_in IN NUMBER, p_in2 IN VARCHAR2) AS
```

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Here I am a Number and Varchar2.');
END;
END;

BEGIN
OVERLOAD.WHAT_AM_I('hola');
OVERLOAD.WHAT_AM_I(29);
OVERLOAD.WHAT_AM_I('29-Apr-2019');
OVERLOAD.WHAT_AM_I(29,'hola');
END;
```

4. Modify the overload package specification again to add a fifth procedure called what_am_i with the same two IN parameters as the fourth procedure. Try to recreate the specification. What happens and why?

```
 Error at line 6: PLS-00305: previous use of 'WHAT_AM_I' (at line 5) conflicts
with this use
```

Ya existe un procedimiento con el mismo tipo de dato en los parámetros.

5. Suppose you write a packaged function that returns a BOOLEAN value of TRUE or FALSE. Does the BOOLEAN limit the possible places from which you can call the packaged function?

   No se podria utilizar en sentencias SQL, pero si se podría usar en procedures, functions, y bloques anonimos.

6. Create a package init_pkg as follows:

```
CREATE OR REPLACE PACKAGE init_pkg IS
g_max_sal  employees.salary%TYPE;
  PROCEDURE get_emp_sal
    (p_emp_id IN employees.employee_id%TYPE);
END init_pkg;

CREATE OR REPLACE PACKAGE BODY init_pkg IS
PROCEDURE get_emp_sal
     (p_emp_id IN employees.employee_id%TYPE) IS
v_salary  employees.salary%TYPE;
  BEGIN
    SELECT salary INTO v_salary
      FROM employees
      WHERE employee_id = p_emp_id;
    IF v_salary > (g_max_sal / 2) THEN
      DBMS_OUTPUT.PUT_LINE('This employee earns MORE than half of '
           || g_max_sal);
    ELSE
      DBMS_OUTPUT.PUT_LINE('This employee earns LESS than half of '
           || g_max_sal);
    END IF;
  END get_emp_sal;
  BEGIN
    SELECT MAX(salary) INTO g_max_sal
      FROM employees;
END init_pkg;
```

Now execute the following anonymous block. Explain the output.

```
BEGIN
```

```
  init_pkg.get_emp_sal(101); -- line 1
init_pkg.g_max_sal := 80000; -- line 2
init_pkg.get_emp_sal(101); -- line 3
  DBMS_OUTPUT.PUT_LINE('g_max_sal is ' || init_pkg.g_max_sal); -- line 4
END;
```

Without doing anything else, execute the following anonymous block. Explain the results.

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('g_max_sal is ' || init_pkg.g_max_sal);
END;
```

El bloque anónimo llama a la variable global que esta en el package y esta contiene el valor máximo que puede tener como salario un empleado.

7. Create a bodiless package called time_conversion_cons that contains constants that can be used to convert:

- days into hours, minutes, and seconds,
- hours into days, minutes, and seconds,
- minutes into days, hours, and seconds, and •
  seconds into days, hours, and minutes.

The first three lines of code for the package are:

```
CREATE OR REPLACE PACKAGE time_conversion_cons AS
c_day_to_hr     CONSTANT NUMBER  := 24;   c_day_to_min
CONSTANT NUMBER := 1440;
```

Create an anonymous block that uses the package constants to display the following conversions. Your output should be four lines formatted as, "x days is x hours or x minutes or x seconds," using the appropriate units on each line. Use good coding practices.

From	To	To	To
2.5 days	Hours	Minutes	Seconds
1.8 hours	Days	Minutes	Seconds
13 minutes	Days	Hours	Seconds
720 seconds	Days	Hours	Minutes