Database Programming with PL/SQL

9-1: Creating Functions

Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

STORED FUNCION	A named PL/SQL block that can accept optional IN parameters and must return a single output.

Try It / Solve It

1. Name the characteristics of a stored function.

   Una función siempre debe retornar un solo valor.

   Las funciones pueden ser utilizadas en sentencias SQL y en bloques PL/SQL

   Al igual que los procedimientos, las funciones aceptan parámetros.

2. Create a function called full_name. Pass two parameters to the function, an employee's last name and first name. The function should return the full name in the format, last name, comma, space, first name (for example: Smith, Joe). Save your code.

```
CREATE OR REPLACE FUNCTION full_name(p_first_name employees.first_name%type,
p_last_name employees.last_name%type) RETURN  VARCHAR2  IS
v_full VARCHAR2(50);
BEGIN
v_full:= p_last_name||' , '||p_first_name;
return v_full;
END;
```

A. Test your function from an anonymous block which uses a local variable to store and display the returned value.

```
DECLARE
v_first_name employees.first_name%type:= 'Jorge';
v_last_name employees.last_name%type:= 'Meza';
v_name VARCHAR2(20);
BEGIN
V_NAME:= full_name(v_first_name,v_last_name);
dbms_output.put_line(V_NAME);
END;
```

B. Modify your anonymous block from the previous step to remove the local variable declaration and call the function directly from within the DBMS_OUTPUT.PUT_LINE call. Test the block again.

```
DECLARE
v_first_name employees.first_name%type:= 'Jorge';
v_last_name employees.last_name%type:= 'Meza';
BEGIN
dbms_output.put_line(full_name(v_first_name,v_last_name));
END;
```

C. Now call the function from within a SELECT statement, not a PL/SQL block. Your SELECT statement should display the first_name, last_name, and full name (using the function) of all employees in department 50. Your output should look like this:

FIRST_NAME	LAST_NAME	Full Name
Kevin	Mourgos	Mourgos, Kevin
Trenna	Rajs	Rajs, Trenna
Curtis	Davies	Davies, Curtis
Randall	Matos	Matos, Randall
Peter	Vargas	Vargas, Peter

```
SELECT first_name,last_name, full_name(first_name,last_name) AS "Full Name" FROM
EMPLOYEES WHERE DEPARTMENT_ID =50;
```

3. Create a function called divide that accepts two numbers as input and returns the result of dividing the first number by the second number, rounded to two decimal places. Save your code.

```
CREATE OR REPLACE FUNCTION divide(p_n1 NUMBER,
p_n2 NUMBER) RETURN  NUMBER  IS
v_res NUMBER;
BEGIN
v_res:= p_n1/p_n2;
return v_res;
END;
```

A. Test your function twice from an anonymous block using input values (50, 2) and (25, 3).

```
BEGIN
dbms_output.put_line(divide(50,2));
END;
```

```
BEGIN
dbms_output.put_line(divide(25,3));
END;
```

B. Test your function a third time using input values (16, 0). What happens?

```
BEGIN
dbms_output.put_line(divide(16,0));
END;
ORA-01476: divisor is equal to zero
```

Marca error porque se esta dividiendo entre cero.

C. Modify the function code to trap the ZERO_DIVIDE exception. The exception handler should return a value of zero from the function if ZERO_DIVIDE is raised.

```
CREATE OR REPLACE FUNCTION divide(p_n1 NUMBER,
p_n2 NUMBER) RETURN  NUMBER  IS
v_res NUMBER;
BEGIN
v_res:= p_n1/p_n2;
return v_res;
EXCEPTION
when zero_divide then
return 0;
```

END;

D. Test your function again using input values (16,0) as before. Now what happens?

BEGIN

dbms_output.put_line(divide(16,0));

END;

La función retorna cero cada vez que se divide entre cero.

4. List four major differences between a procedure and a function.

Un procedimiento se invoca en un bloque anónimo, en cambio una función puede ser parte de una expresión.

En una función se tiene que indicar el tipo de dato del valor que se va retornar.

Una función solo retorna un solo valor, en cambio un procedimiento puede retornar varios valores.

Las funciones se pueden usar en sentencias SQL y bloques PL/SQL y los procedimientos solo se usan en PL/SQL.

5. Look at the following two subprograms. The first is a procedure; the second is a function. Answer the following questions.

```
CREATE OR REPLACE PROCEDURE get_country_name_proc
 (p_country_id IN countries.country_id%TYPE,
p_country_name OUT countries.country_name%TYPE)
IS
BEGIN
```

```
  SELECT country_name INTO p_country_name
    FROM countries
    WHERE country_id = p_country_id;
END;

CREATE OR REPLACE FUNCTION get_country_name_func
    (p_country_id IN  countries.country_id%TYPE)
  RETURN VARCHAR2
IS
  v_country_name          countries.country_name%TYPE;
BEGIN
  SELECT country_name INTO v_country_name
    FROM countries
    WHERE country_id = p_country_id;
  RETURN v_country_name;
END;
```

A. For a given country id, will both of these subprograms return the same results?

Si, los dos subprogramas retornaran lo mismo.

B. What is the advantage of creating this subprogram as a function rather than as a procedure?

Puede ser invocado en una sentencia SQL sin necesidad de tener que hacer un bloque para poder llamarlo.

C. Which of the following procedure and function calls are valid and which are not? Explain why the invalid ones will fail.

```
DECLARE
```

```
    v_country_id    countries.country_id%TYPE       :=    2;
   v_country_name          countries.country_name%TYPE;
   BEGIN
  get_country_name_proc(v_country_id, v_country_name);        -- Call 1

  v_country_name := get_country_name_func(v_country_id);      -- Call 2

    v_country_name := get_country_name_proc(v_country_id);        -- Call 3
```

Solo se manda un parámetro.

```
END;


      SELECT get_country_name_proc(country_id)                        -- Call 4
  FROM countries;
```

Un procedimiento no se puede invocar en sentencias SQL

```
      SELECT get_country_name_func(country_id)                    -- Call 5
        FROM countries;
```

6. List the ways you can invoke (i.e., call) a function.

- **Se puede asignar la función a una variable.**
- **Se puede mandar llamar en un DBMS_OUTPUT.PUT_LINE**
- **Se puede invocar en sentencias SQL**

7. Create a function which accepts a character string as input and returns the same character string but with the order of the letters reversed. For example, "Smith" would be returned as "htimS." Save your code. Hint: you will need to declare a local variable to store the reversed string, and build its contents by reading the input one character at a time (using SUBSTR) in a loop structure, starting from the last character. Each execution of the loop reads the preceding character and concatenates it to the reversed string.

```
CREATE OR REPLACE FUNCTION reverse_string(p_nombre employees.first_name%type)
RETURN VARCHAR2 IS
 reverse varchar2(100);
 BEGIN
 FOR i in reverse 1..length(p_nombre) LOOP
 reverse := reverse||''||substr(p_nombre, i, 1);
 END LOOP;
```

```
RETURN reverse;
END;
```

8. Test your function using the following SQL statements:

```
SELECT last_name, reverse_string(last_name)
  FROM employees;
```


```
SELECT country_name, reverse_string(country_name)
  FROM countries;
```