Database Programming with PL/SQL

9-2: Using Functions in SQL Statements

Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

User-Defined Function	A function created by the PL/SQL programmer that can be used anywhere there is a value or function.

Try It / Solve It

The questions in this Practice use partial copies of the employees and departments tables. Create these copies by executing the following SQL statements:

```
CREATE TABLE f_emps
  AS SELECT employee_id, last_name, salary, department_id
  FROM employees;

CREATE TABLE f_depts
  AS SELECT department_id, department_name
  FROM departments;
```

1. Create and execute a function sal_increase using the following two code samples. The first creates a function which returns an employee's new salary if a percentage increase is granted. The second calls this function in a SELECT statement, using an increase of 5 percent.

```
CREATE OR REPLACE FUNCTION sal_increase
(p_salary          f_emps.salary%TYPE,
p_percent_incr     NUMBER)
```

```
  RETURN NUMBER
IS

BEGIN

  RETURN (p_salary + (p_salary * p_percent_incr / 100));

END;
```

SELECT last_name, salary, sal_increase(salary, 5)

FROM f_emps;

Now, suppose you want to see the same information in your SELECT statement, but only for those employees for whom the increased salary would be greater than 10000. Write and test two SELECT statements to do this. In the first, do NOT use your function. In the second, use your function. Use an increase of 5 percent. Which do you think is better, and why?

SELECT last_name, salary, salary + (salary * 5 / 100) AS "aumento" FROM f_emps WHERE (salary + (salary * 5 / 100)) > 10000;

SELECT last_name, salary, sal_increase(salary, 5) FROM f_emps;

Yo pienso que es mejor tener una función que realice esa operación, ya que es mas sencillo porque se puede utilizar las veces que se necesiten y además es más rápido.

2. Name five places within a SQL statement where a function can be used. The first one has been done for you (think of four more).

- The column-list of a SELECT statement
- **En un SELECT en la parte del WHERE**
- **En una SELECT en la parte del HAVING**

- **En un SELECT, en la parte de ORDER BY**

- **En una sentencia UPDATE al momento de actualizar el salario de un empleado.**

3. Modify your anonymous block from question 1 (the block with the calls to the sal_increase function) to ORDER the results by the increased salary in descending order (i.e., highest increased salary first).

```
SELECT last_name, salary, sal_increase(salary, 5)
FROM f_emps
ORDER BY (sal_increase(salary, 5)) desc;
```

4. Examine the following SELECT statement which lists the total salaries in each department for those departments whose total salary is greater than 20000.

```
SELECT department_id, SUM(salary)
  FROM f_emps
  GROUP BY department_id
  HAVING SUM(salary) > 20000;
```

Modify the statement so that it also lists the total salary in each department if a 5 percent increase is granted, and lists those departments whose increased total salary would be greater than 20000. Your modified statement should call the sal_increase function twice, once in the column_list and once in the HAVING clause. Test the modified statement.

```
SELECT department_id, SUM(sal_increase(salary, 5)))
 FROM f_emps
 GROUP BY department_id
 HAVING SUM(sal_increase(salary, 5))) > 20000;
```

5. The following function accepts a department id as an input parameter and checks whether the department exists in the f_depts table. Run this code to create the check_dept function.

```
CREATE OR REPLACE FUNCTION check_dept
   (p_dept_id       f_depts.department_id%TYPE)
RETURN BOOLEAN IS
 v_dept_id        f_depts.department_id%TYPE;
BEGIN
 SELECT department_id INTO v_dept_id
   FROM f_depts
   WHERE department_id = p_dept_id;
 RETURN TRUE;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
   RETURN FALSE;
END;
```

Examine the above function and explain why it could not be used within a SQL statement. Could this function be used within a PL/SQL statement? Why or why not?

Porque la función retorna un tipo de dato booleano y el booleano no existe en SQL.

6. Write a procedure called insert_emp which inserts a new employee into f_emps. Pass the employee id, last name, salary, and department id to the procedure as IN parameters. The procedure should call your check_dept function to verify that the passed department id exists in the f_depts table. If it exists, insert the employee. If it does not exist, use DBMS_OUTPUT.PUT_LINE to display a suitable error message. Save your code.

```
CREATE OR REPLACE PROCEDURE insert_emp(p_emp_id f_emps.employee_id%type,
 p_apellido IN f_emps.last_name%type,
p_salario IN f_emps.salary%type,
p_department_id IN f_emps.department_id%type
```

```
) AS
v_check boolean;
BEGIN
v_check:= check_dept(p_department_id);
IF v_check = TRUE THEN
    INSERT INTO f_emps(employee_id,last_name,salary,department_id)
values(p_emp_id,p_apellido,p_salario,p_department_id);
ELSE
    DBMS_OUTPUT.PUT_LINE('No existe ese departamento ');
end if;
END;
```

7. Test your insert_emp procedure from an anonymous block using the following IN parameter values: employee_id = 800, last_name = Jokinen, salary = 5000, and department_id = 750. What happened and why?

```
BEGIN
insert_emp(800,'Jokinen',5000,750);
END;
```

```
No existe ese departamento
```

No inserta al empleado, porque no existe ningún departamento con id 750.

8. Modify your insert_emp procedure so that if the department does not exist, the procedure first inserts a new department with the non-existent department id and a department name of 'Temporary', and then inserts the employee. Test your procedure again with the same IN values used in the previous question.

```
CREATE OR REPLACE PROCEDURE insert_emp(p_emp_id f_emps.employee_id%type,
p_apellido IN f_emps.last_name%type,
p_salario IN f_emps.salary%type,
p_department_id IN f_emps.department_id%type
) AS
v_check boolean;
BEGIN
v_check:= check_dept(p_department_id);
IF v_check = FALSE THEN
INSERT INTO f_depts values(p_department_id,'Temporal');
end if;

INSERT INTO f_emps(employee_id,last_name,salary,department_id)
values(p_emp_id,p_apellido,p_salario,p_department_id);
END;
```

9. Execute two SELECT statements to confirm department id 750 and employee id 800 were added to the F_DEPTS and F_EMPS tables, respectively.

10. Create the function get_sal using the following code:

```
CREATE OR REPLACE FUNCTION get_sal
   (p_emp_id          f_emps.employee_id%TYPE)
  RETURN NUMBER
IS
  v_salary       f_emps.salary%TYPE;
BEGIN
  SELECT salary INTO v_salary
    FROM f_emps
    WHERE employee_id = p_emp_id;
  RETURN v_salary;
END;
```

Use the get_sal function in the following SQL statement (which attempts to move all highersalaried employees to department 50). What happens and why?

```
UPDATE f_emps
  SET department_id = 50
  WHERE get_sal(employee_id) > 10000;
```

```
 ORA-04091: table MX_A104_SQL_S39.F_EMPS is mutating, trigger/function may not see
it
```

Estamos tratando de usarla en una declaración SQL que también realiza DML en la misma tabla.

11. Examine the following function (which doubles the salary of a chosen employee) and the SQL statement which uses it. What will happen when the SQL statement is executed? Why? Create the upd_sal function, then run the SELECT statement to confirm your prediction.

```
CREATE OR REPLACE FUNCTION upd_sal
    (p_emp_id           f_emps.employee_id%TYPE)
  RETURN NUMBER
IS
  v_salary       f_emps.salary%TYPE;
BEGIN
  SELECT salary INTO v_salary
    FROM f_emps
    WHERE employee_id = p_emp_id;
v_salary := v_salary * 2;
  UPDATE f_emps
    SET salary = v_salary
    WHERE employee_id = p_emp_id;
  RETURN v_salary;
END;

SELECT employee_id, last_name, salary, upd_sal(employee_id)
  FROM f_emps
  WHERE employee_id = 100;
```

ORA-14551: cannot perform a DML operation inside a query