LENGUAJE DE PROGRAMACIÓN ORACLE

Bloques Anónimos, Procedimientos, funciones y disparadores.

Contenido

BLOQUES ANÓNIMOS

1. Programa que intercambie los salarios de los empleados con id 120 y 122.

```
DECLARE
V_salary_120 employees.salary%type;
Begin
SELECT salary INTO V_salary_120
FROM employees WHERE employee_id = 120;
UPDATE employees SET salary = ( SELECT salary FROM employees
WHERE employee_id = 122)
WHERE employee_id = 120;
UPDATE employees SET salary = V_salary_120 WHERE employee_id = 122;
COMMIT;
END;
```

2. Incrementar el salario del empleado 115 basado en las siguientes condiciones: Si la experiencia es de más de 10 años aumentar 20%, si la experiencia es mayor que 5 años aumentar 10%, en otro caso 5%.

```
DECLARE
v_exp NUMBER(2);
v_aumento NUMBER(5,2);
BEGIN
SELECT FLOOR((SYSDATE - hire_date) / 365) INTO v_exp
FROM employees
WHERE employee_id = 115;

v_aumento := 1.05;

CASE
WHEN v_exp > 10 THEN
v_aumento := 1.20;
WHEN v_exp > 5 THEN
```

```
v_aumento := 1.10;
END CASE;

UPDATE employees SET salary = salary * v_aumento
WHERE employee_id = 115;
END;
```

3. Cambiar el porcentaje de comisión para el empleado con id 150. Si el salario es mayor que 10000 cambiarlo a 0.4%. Si el salario es menor a 10000 pero la experiencia es mayor a 10 años, entonces 0.35%. Si el salario es menor a 3000 entonces a 0.25%, en otro caso 0.15%.

```
DECLARE
v_salary employee.salary%type;
v_exp NUMBER(2);
v_porcentaje NUMBER(5,2);
BEGIN
SELECT salary, FLOOR((SYSDATE - hire_date) / 365) INTO v_salary, v_exp
FROM employees
WHERE employee_id = 150;

IF v_salary > 10000 THEN
v_porcentaje := 0.4;
ELSIF v_exp > 10 THEN
v_porcentaje := 0.35;
ELSIF v_salary < 3000 THEN
v_porcentaje := 0.25;
ELSE
v_porcentaje := 0.15;
END IF;

UPDATE employees SET commission_pct = v_porcentaje
WHERE employee_id = 150;
END;
```

4. Encontrar el nombre del empleado y del departamento para el manager que está a cargo del empleado id 103

```
DECLARE
v_name employees.first_name%type;
v_deptname departments.department_name%type;
BEGIN
SELECT first_name, department_name INTO v_name, v_deptname
FROM employees JOIN departments USING(department_id)
WHERE employee_id = (SELECT manager_id FROM employees
WHERE employee_id = 103);

dbms_output.put_line(v_name);
dbms_output.put_line(v_deptname);
```

```
END;
```

5. Mostrar los ids de los empleados que faltan.

```
DECLARE
v_min NUMBER(3);
v_max NUMBER(3);
v_c NUMBER(1);
BEGIN
SELECT MIN(employee_id), MAX(employee_id) INTO v_min, v_max
FROM employees;

FOR i in v_min + 1 .. v_max - 1
LOOP
SELECT COUNT(*) INTO v_c
FROM employees
WHERE employee_id = i;

IF v_c = 0 THEN
dbms_output.put_line(i);
END IF;
END LOOP;
END;
```

6. Mostrar el año en que se unió el máximo número de empleados y mostrar cuantos empleados por mes se unieron dicho año.

```
DECLARE
v_year NUMBER(4);
v_c NUMBER(2);
BEGIN
SELECT TO_CHAR(hire_date, 'YYYY') INTO v_year
FROM employees
GROUP BY TO_CHAR(hire_date, 'YYYY')
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
FROM employees
GROUP BY TO_CHAR(hire_date, 'YYYY'));

dbms_output.put_line('Year: ' || v_year);

FOR month IN 1 .. 12
LOOP
SELECT COUNT(*) INTO v_c
FROM employees
WHERE TO_CHAR(hire_date, 'MM') = month AND TO_CHAR(hire_date, 'YYYY') = v_year;

dbms_output.put_line('Month: ' || TO_CHAR(month) || '# Emp: ' || TO_CHAR(v_c));
END LOOP;
```

```
END;
```

7. Cambiar el salario del empleado 130 por el salario del empleado de nombre 'Joe'. Si no se encuentra Joe, se debe tomar el promedio del salario de todos los empleados. Si hay más de un empleado con el nombre Joe se deberá tomar el menor de todos los salarios de los empleados de nombre Joe

```
DECLARE
v_salary employee.salary%type;
BEGIN
SELECT salary INTO v_salary
FROM employees
WHERE first_name = 'Joe';

UPDATE employees SET salary = v_salary
WHERE employee_id = 130;

EXCEPTION
WHEN no_data_found THEN
UPDATE employees SET salary = (SELECT AVG(salary) FROM employees)
WHERE employee_id = 130;
END;
```

8. Mostrar el nombre del puesto y el nombre del empleado que ingresó primero a ese puesto.

```
DECLARE
CURSOR jobscur IS SELECT job_id, job_title FROM jobs;
v_name employees.first_name%type;
BEGIN
FOR jobrec IN jobscur
LOOP
SELECT first_name into v_name
FROM employees
WHERE hire_date = (SELECT MIN(hire_date)
FROM employees WHERE job_id = jobrec.job_id)
AND job_id = jobrec.job_id;

dbms_output.put_line(jobrec.job_title || '-' || v_name);
END LOOP;
END;
```

9. Mostrar del 5° al 10° empleado de la tabla de employees

```
DECLARE
CURSOR empcur IS SELECT employee_id, first_name FROM employees;
BEGIN
FOR emprec IN empcur
LOOP
```

```
IF empcur%rowcount > 4 THEN
dbms_output.put_line(emprec.employee_id || ' ' || emprec.first_name);
EXIT WHEN empcur%rowcount >= 10;
END IF;
END LOOP;
END;
```

10. Actualizar el salario de un empleado basado en su departamento y porcentaje de comisión. Si el departamento es 40 aumentar un 10%, si el departamento es 70 aumentar 15%. Si el porcentaje de comisión es mayor que 0.3% aumentar 5%, en otro caso aumentar 10%

```
DECLARE
CURSOR empcur IS SELECT employee_id, department_id, commission_pct
FROM employees;
v_aumento NUMBER(2);
BEGIN
FOR emprec IN empcur
LOOP
IF emprec.department_id = 40 THEN
v_aumento := 10;
ELSIF emprec.department_id = 70 THEN
v_aumento := 15;
ELSIF emprec.commission_pct > 0.3 THEN
v_aumento := 5;
ELSE
v_aumento := 10;
END IF;

UPDATE employees SET salary = salary + (salary * v_aumento / 100)
WHERE employee_id = emprec.employee_id;
END LOOP;
END;
```

FUNCTION

12. Crear una función que reciba el id de empleado y regrese el número de puestos que ha tenido dicho empleado.

```
CREATE OR REPLACE FUNCTION get_no_of_jobs(empid NUMBER)
RETURN NUMBER IS
v_count NUMBER(2);
BEGIN
SELECT COUNT(*) INTO v_count
FROM job_history
WHERE employee_id = empid;

RETURN v_count;
END;
```

13. Crear un procedimiento que reciba el ID de departamento y cambie su manager por el empleado en dicho departamento con el salario más alto

```
CREATE OR REPLACE PROCEDURE change_dept_manager(deptid NUMBER)
IS
v_empid employees.employee_id%type;
BEGIN
SELECT employee_id INTO v_empid
FROM employees
WHERE salary = (SELECT MAX(salary)
FROM employees
WHERE department_id = deptid)
AND department_id = deptid;

UPDATE departments SET manager_id = v_empid
WHERE department_id = deptid;
END;
```

14. Crear una función que reciba el manager id y regrese el nombre de los empleados a cargo de dicho manager, los nombres deben ser retornados como una cadena separada por comas.

```
CREATE OR REPLACE FUNCTION get_employees_for_manager(manager NUMBER)
RETURN VARCHAR2 IS
v_employees VARCHAR2(1000) := '';
CURSOR empcur IS
SELECT first_name FROM employees
WHERE manager_id = manager;
BEGIN
FOR emprec IN empcur
LOOP
v_employees := v_employees || ',' || emprec.first_name;
END LOOP;

RETURN LTRIM(v_employees, ',');
END;
```

TRIGGER

15. Asegurar que no se puedan hacer cambios en la tabla de empleados antes de las 6am y después de las 10pm

```
CREATE OR REPLACE TRIGGER trg_employees_time_check
BEFORE UPDATE OR INSERT OR DELETE
ON employees
FOR EACH ROW
BEGIN
IF TO_CHAR(SYSDATE, 'hh24') < 6 OR TO_CHAR(SYSDATE, 'hh24') > 22 THEN
raise_application_error(-20111, 'No se permiten modificaciones');
END IF;
```

```
END;
```

16. Crear un trigger para asegurar que el salario de un empleado no disminuya

```
CREATE OR REPLACE TRIGGER trg_employees_salary_check
BEFORE UPDATE
ON employees
FOR EACH ROW
BEGIN
IF :old.salary > :new.salary THEN
raise_application_error(-20111, 'No es posible cambiar el salario a
una cantidad menor');
END IF;
END;
```

17. Cada que se cambie un puesto para un empleado, escribir los siguientes detalles en la tabla job_history. Id del empleado, antiguo job_id, fecha de contratación para la nueva fecha de inicio, y sysdate para la fecha de fin. Si la fila ya existe, entonces la fecha de inicio debe ser la fecha de fin de esa fila + 1.

```
CREATE OR REPLACE TRIGGER trg_log_job_change
AFTER UPDATE OF job_id
ON employees
FOR EACH ROW
v_enddate DATE;
v_startdate DATE;
BEGIN
SELECT MAX(end_date) INTO v_enddate
FROM job_history
WHERE employee_id = :old.employee_id;

IF v_enddate IS NULL THEN
v_startdate := :old.hire_date
ELSE
v_startdate := v_enddate + 1;
END IF;

INSERT INTO job_history VALUES (:old.employee_id, v_startdate, sysdate,
:old.job_id, :old.department_id);
END;
```

Otros Ejemplos Anónimos

```
DECLARE
 -- Este ejemplo es Hola Mundo
 mensaje VARCHAR2(20) := 'Hola mundo';
BEGIN
 /*
```

```
 Estas son
 líneas ejecutables

 dbms_output.put_line(mensaje);
END;

DECLARE
 num1 INTEGER;
 num2 REAL;
 num3 DOUBLE PRECISION;
 nombre VARCHAR2(50);
 CP CHAR(5);
BEGIN
 NULL;
END;


 DECLARE
 num1 INTEGER;
 ventas NUMBER(10, 2);
 pi CONSTANT DOUBLE PRECISION := 3.14159;
 nombre VARCHAR(2);
 direccion VARCHAR2(100);
 var FLOAT DEFAULT 1.5;
BEGIN
 -- Este programa declara variables
 NULL;

 /*
  Y demuestra el uso de comentarios

END;



DECLARE
 a INTEGER := 10;
 b INTEGER := 20;
 c INTEGER;
 f REAL;
BEGIN
 c := a + b;
 dbms_output.put_line('Valor de c: ' || c);
 f := 70.0 / 3.0;
 dbms_output.put_line('Valor de f: ' || f);
END;

DECLARE
```

```
 -- Variables globales
 num1 NUMBER := 95;
 num2 NUMBER := 85;
BEGIN
 dbms_output.put_line('Variable global num1: ' || num1);
 dbms_output.put_line('Variable global num2: ' || num2);

 DECLARE
  num1 NUMBER := 195;
  num2 NUMBER := 185;
 BEGIN
  dbms_output.put_line('Variable local num1: ' || num1);
  dbms_output.put_line('Variable local num2: ' || num2);
 END;
END;

DECLARE
 nombre    employees.first_name%type;
 apellido  employees.last_name%type;
 sueldo    employees.salary%type;
BEGIN
 SELECT first_name, last_name, salary
  INTO nombre, apellido, sueldo
 FROM employees
 WHERE employee_id = 110;
 dbms_output.put_line('Nombre: ' || nombre);
 dbms_output.put_line('Apellido: ' || apellido);
 dbms_output.put_line('Sueldo: ' || sueldo);
END;

BEGIN
 dbms_output.put_line('10 + 5: ' || (10 + 5));
 dbms_output.put_line('10 - 5: ' || (10 - 5));
 dbms_output.put_line('10 * 5: ' || (10 * 5));
 dbms_output.put_line('10 / 5: ' || (10 / 5));
 dbms_output.put_line('10 ** 5: ' || (10 ** 5));
END;

DECLARE
 a NUMBER(2) := 21;
 b NUMBER(2) := 10;
BEGIN
 IF(a = b) THEN
  dbms_output.put_line('A es igual a B');
 ELSE
  dbms_output.put_line('A no es igual a B');
 END IF;
```

```
 IF(a < b) THEN
  dbms_output.put_line('A es menor que B');
 ELSE
  dbms_output.put_line('A no es menor que B');
 END IF;

 IF(a > b) THEN
  dbms_output.put_line('A es mayor que B');
 ELSE
  dbms_output.put_line('A no es mayor que B');
 END IF;

 a := 5;
 b := 20;

 IF(a <= b) THEN
  dbms_output.put_line('A es menor o igual que B');
 ELSE
  dbms_output.put_line('A no es menor o igual que B');
 END IF;

 IF(a >= b) THEN
  dbms_output.put_line('A es mayor o igual que B');
 ELSE
  dbms_output.put_line('A no es mayor o igual que B');
 END IF;

 IF(a <> b) THEN
  dbms_output.put_line('A es diferente de B');
 ELSE
  dbms_output.put_line('A no es diferente de B');
 END IF;
END;

DECLARE
 a BOOLEAN := TRUE;
 b BOOLEAN := FALSE;
BEGIN
 IF(a AND b) THEN
  dbms_output.put_line('1a condición: TRUE');
 END IF;

 IF(a OR b) THEN
  dbms_output.put_line('2a condición: TRUE');
 END IF;

 IF(NOT a) THEN
  dbms_output.put_line('3a condición: a no es TRUE');
```

```
 ELSE
  dbms_output.put_line('3a condición: a es TRUE');
 END IF;

 IF(NOT b) THEN
  dbms_output.put_line('4a condición: b no es TRUE');
 ELSE
  dbms_output.put_line('4a condición: b es TRUE');
 END IF;
END;

DECLARE
 a NUMBER(2) := 10;
BEGIN
  a := 30;
  -- revisar condición
  IF( a < 20) THEN
    -- La condición fue verdadera
    dbms_output.put_line('A es mayor que 20');
  END IF;

  dbms_output.put_line('Valor de a: ' || a);
END;

DECLARE
 a NUMBER(3) := 20;
BEGIN
 IF(a < 50) THEN
  dbms_output.put_line('A es menor que 50');
 ELSE
  dbms_output.put_line('A no es menor que 50');
 END IF;

 dbms_output.put_line('Valor de a: ' || a);
END;

DECLARE
 A number(3) := 30;
BEGIN
 IF(a = 10) THEN
  dbms_output.put_line('A es igual a 10');
 ELSIF(a = 20) THEN
  dbms_output.put_line('A es igual a 20');
 ELSIF(a = 30) THEN
  dbms_output.put_line('A es igual a 30');
 ELSE
  dbms_output.put_line('El valor de a no coincidio');
 END IF;
```

```
 dbms_output.put_line('Valor de A: ' || a);
END;

DECLARE
 calif CHAR(2) := '10';
BEGIN
 CASE
  WHEN calif = '10' THEN dbms_output.put_line('Excelente');
  WHEN calif = '9' THEN dbms_output.put_line('Muy bien');
  WHEN calif = '8' THEN dbms_output.put_line('Bien hecho');
  WHEN calif = '7' THEN dbms_output.put_line('Pasaste');
  WHEN calif = '5' THEN dbms_output.put_line('Reprobado');
 END CASE;
END;

DECLARE
 x NUMBER := 10;
BEGIN
 LOOP
  dbms_output.put_line(x);
  x := x + 10;

  EXIT WHEN x > 50;
 END LOOP;

 dbms_output.put_line('Al terminar el ciclo x: ' || x);
END;

DECLARE
 a NUMBER(2) := 10;
BEGIN
 WHILE a < 20 LOOP
  dbms_output.put_line('Valor de a: ' || a);
  a := a + 1;
 END LOOP;
END;

DECLARE
 a NUMBER(2);
BEGIN
 FOR a IN REVERSE 10 .. 20 LOOP
  dbms_output.put_line('Valor de a: ' || a);
 END LOOP;
END;

CREATE OR REPLACE PROCEDURE saludos
AS
```

```
BEGIN
 dbms_output.put_line('Saludos desde procedimiento');
END;

--EXECUTE saludos;

BEGIN
 saludos;
END;

DROP PROCEDURE saludos;

DECLARE
 a NUMBER;
 b NUMBER;
 c NUMBER;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
 IF x < y THEN
  z := x;
 ELSE
  z := y;
 END IF;
END;
BEGIN
 a := 23;
 b := 45;
 findMin(a, b, c);
 dbms_output.put_line('El mínimo de 45 y 23 es: ' || c);
END;

DECLARE
 a NUMBER;
PROCEDURE cuadrado(x IN OUT NUMBER) IS
BEGIN
 x := x ** 2;
END;
BEGIN
 a := 9;
 cuadrado(a);
 dbms_output.put_line('Cuadrado de 9: ' || a);
END;

CREATE OR REPLACE FUNCTION totalEmpleados
RETURN NUMBER IS
 total NUMBER(3) := 0;
BEGIN
 SELECT COUNT(*) INTO total
```

```
  FROM employees;

  RETURN total;
END;

DECLARE
  c NUMBER(3);
BEGIN
  c := totalEmpleados();
  dbms_output.put_line('Total: ' || c);
END;

-- DROP function nombre;

DECLARE
  a NUMBER;
  b NUMBER;
  c NUMBER;
FUNCTION findMax(x IN NUMBER, y IN NUMBER)
RETURN NUMBER
IS
  z NUMBER;
BEGIN
  IF x > y THEN
    z := x;
  ELSE
    z := y;
  END IF;

  RETURN z;
END;
BEGIN
  a := 55;
  b := 31;

  c := findMax(a, b);
  dbms_output.put_line('El mayor de a y b es: ' || c);
END;

DECLARE
  num NUMBER;
  factorial NUMBER;
FUNCTION fact(x NUMBER)
RETURN NUMBER
IS
  f NUMBER;
BEGIN
  IF x = 0 THEN
```

```
  f := 1;
 ELSE
  f := x * fact(x - 1);
 END IF;

 RETURN f;
END;
BEGIN
 num := 6;
 factorial := fact(num);
 dbms_output.put_line('Factorial de 6: ' || factorial);
END;
```

Referencias Bibliográficas

https://github.com/enrique7mc/sql-fundamentals/blob/master/PLSQL%20excercises.sql
https://github.com/enrique7mc/sql-fundamentals/blob/master/SQL%20excercises.sql
https://github.com/enrique7mc/sql-fundamentals/blob/master/PLSQL%20examples.sql