Prácticas ORACLE PL/SQL

Práctica de PAQUETES

1. **Crear un paquete denominado REGIONES que tenga los siguientes componentes:**
   - **PROCEDIMIENTOS:**
     - - ALTA_REGION, con parámetro de código y nombre Región. Debe devolver un error si la región ya existe. Inserta una nueva región en la tabla. Debe llamar a la función EXISTE_REGION para controlarlo.
     - - BAJA_REGION, con parámetro de código de región y que debe borrar una región. Debe generar un error si la región no existe, Debe llamar a la función EXISTE_REGION para controlarlo
     - - MOD_REGION: se le pasa un código y el nuevo nombre de la región Debe modificar el nombre de una región ya existente. Debe generar un error si la región no existe, Debe llamar a la función EXISTE_REGION para controlarlo
   - **FUNCIONES**
     - CON_REGION. Se le pasa un código de región y devuelve el nombre
     - EXISTE_REGION. Devuelve verdadero si la región existe. Se usa en los procedimientos y por tanto es PRIVADA, no debe aparecer en la especificación del paquete
2. **Crear un paquete denominado NOMINA que tenga sobrecargado la función CALCULAR_NOMINA de la siguiente forma**:
   - CALCULAR_NOMINA(NUMBER): se calcula el salario del empleado restando un 15% de IRPF.
   - CALCULAR_NOMINA(NUMBER,NUMBER): el segundo parámetro es el porcentaje a aplicar. Se calcula el salario del empleado restando ese porcentaje al salario
   - CALCULAR_NOMINA(NUMBER,NUMBER,CHAR): el segundo parámetro es el porcentaje a aplicar, el tercero vale 'V' . Se calcula el salario del empleado aumentando la comisión que le pertenece y restando ese porcentaje al salario siempre y cuando el empleado tenga comisión.

Soluciones

1. **Crear un paquete denominado REGIONES que tenga los siguientes componentes:**

   - **PROCEDIMIENTOS:**
     - - ALTA_REGION, con parámetro de código y nombre Región. Debe devolver un error si la región ya existe. Inserta una nueva región en la tabla. Debe llamar a la función EXISTE_REGION para controlarlo.
     - - BAJA_REGION, con parámetro de código de región y que debe borrar una región. Debe generar un error si la región no existe, Debe llamar a la función EXISTE_REGION para controlarlo
     - - MOD_REGION: se le pasa un código y el nuevo nombre de la región Debe modificar el nombre de una región ya existente. Debe generar un error si la región no existe, Debe llamar a la función EXISTE_REGION para controlarlo
   - **FUNCIONES**
     - CON_REGION. Se le pasa un código de región y devuelve el nombre
     - EXISTE_REGION. Devuelve verdadero si la región existe. Se usa en los procedimientos y por tanto es PRIVADA, no debe aparecer en la especificación del paquete

```
--PACKAGE HEAD
CREATE OR REPLACE PACKAGE regiones IS
PROCEDURE alta_region ( codigo NUMBER,nombre VARCHAR2 );
PROCEDURE baja_region ( id NUMBER );
PROCEDURE mod_region ( id NUMBER,nombre VARCHAR2 );
FUNCTION con_regiom ( codigo NUMBER ) RETURN VARCHAR2;
END regiones;
/
--BODY / PROCEDURE ALTA REGIÓN
CREATE OR REPLACE PACKAGE BODY regiones IS
--FUNCIÓN EXISTE_REGION
FUNCTION existe_region ( id NUMBER,nombre VARCHAR2 ) RETURN
BOOLEAN IS
CURSOR c1 IS
SELECT
region_name,
region_id
FROM
regions;
```

```
y VARCHAR2(20);
BEGIN
FOR i IN c1 LOOP
IF
i.region_name = nombre AND i.region_id = id
THEN
RETURN true;
END IF;
END LOOP;
RETURN false;
END;
PROCEDURE alta_region ( codigo NUMBER,nombre VARCHAR2 ) IS
devuelto BOOLEAN;
BEGIN
devuelto := existe_region(codigo,nombre);
IF
devuelto = false
THEN
INSERT INTO regions ( region_id,region_name ) VALUES ( codigo,nombre );
dbms_output.put_line('Inserción correcta');
ELSE
dbms_output.put_line('La región ya existe.');
END IF;
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('La ID YA EXISTE (duplicada)');
END;
--PROCEDURE BAJA_REGION
PROCEDURE baja_region ( id NUMBER ) IS
devuelto BOOLEAN;
otro VARCHAR2(20);
BEGIN
SELECT
region_name
INTO
otro
FROM
regions
WHERE
```

```
region_id = id;
devuelto := existe_region(id,otro);
IF
devuelto = true
THEN
DELETE FROM regions WHERE
region_id = id;
dbms_output.put_line('Región con ID ' || id || ' borrada');
END IF;
EXCEPTION
WHEN no_data_found THEN
dbms_output.put_line('La región no existe!');
END;
--PROCEDURE MOD_REGION
PROCEDURE mod_region ( id NUMBER,nombre VARCHAR2 ) IS
devuelto BOOLEAN;
BEGIN
devuelto := existe_region(id,nombre);
IF
devuelto = true
THEN
UPDATE regions
SET
region_name = nombre
WHERE
region_id = id;
dbms_output.put_line('La región ha sido actualizada.');
ELSE
dbms_output.put_line('La región no existe.');
END IF;
END;
--FUNCIÓN CON_REGIOM
FUNCTION con_regiom ( codigo NUMBER ) RETURN VARCHAR2 IS
nombre_devolver VARCHAR2(20);
BEGIN
SELECT
region_name
INTO
nombre_devolver
FROM
```

```
regions
WHERE
region_id = codigo;
RETURN nombre_devolver;
END;
END regiones;
/
EXECUTE mod_region(7,'pikachutotal');
EXECUTE regiones.baja_region(10);
EXECUTE regiones.alta_region(10,'Prueba');
SELECT
*
FROM
regions;
DELETE FROM regions WHERE
region_id > 4;
ROLLBACK;
```

2. **Crear un paquete denominado NOMINA que tenga sobrecargado la función CALCULAR_NOMINA de la siguiente forma**:
   - CALCULAR_NOMINA(NUMBER): se calcula el salario del empleado restando un 15% de IRPF.
   - CALCULAR_NOMINA(NUMBER,NUMBER): el segundo parámetro es el porcentaje a aplicar. Se calcula el salario del empleado restando ese porcentaje al salario
   - CALCULAR_NOMINA(NUMBER,NUMBER,CHAR): el segundo parámetro es el porcentaje a aplicar, el tercero vale 'V' . Se calcula el salario del empleado aumentando la comisión que le pertenece y restando ese porcentaje al salario siempre y cuando el empleado tenga comisión.

```
CREATE OR REPLACE PACKAGE NOMINA IS
function calcular_nomina(id number) RETURN NUMBER;
function calcular_nomina(id number,porcentaje varchar) RETURN NUMBER;
function calcular_nomina(id number,porcentaje number, tercero varchar2:='V') RETURN NUMBER;
END NOMINA;
/
CREATE OR REPLACE PACKAGE BODY NOMINA IS
--PRIMERA FUNCION
•
FUNCTION calcular_nomina ( id NUMBER ) RETURN NUMBER IS
```

```
salario_final NUMBER;
salario NUMBER;
BEGIN
SELECT salary INTO salario from employees where employee_id=id;
salario_final := salario - ( salario * 0.15 );
RETURN salario_final;
END;
--SEGUNDA FUNCION
FUNCTION calcular_nomina ( id NUMBER,porcentaje varchar ) RETURN NUMBER IS
salario_final NUMBER;
salario NUMBER;
BEGIN
SELECT salary INTO salario from employees where employee_id=id;
salario_final := salario - ( salario * (to_number(porcentaje)/100 ));
RETURN salario_final;
END;
--TERCERA FUNCION
FUNCTION calcular_nomina (
id NUMBER,
porcentaje NUMBER,
tercero VARCHAR2 := 'V'
) RETURN NUMBER IS
salario_final NUMBER;
comision NUMBER;
salario NUMBER;
BEGIN
SELECT salary into salario from employees where employee_id=id;
SELECT commission_pct into comision from employees where employee_id=id;
salario_final := salario - ( salario * (porcentaje/100 )) + (salario*comision);
RETURN salario_final;
END;
END NOMINA;
/
declare
x number;
BEGIN
x:=NOMINA.CALCULAR_NOMINA(150,'8');
DBMS_OUTPUT.PUT_LINE(x);
```

```
END;
/
desc nomina;
```