Prácticas ORACLE PL/SQL

Práctica de TRIGGERS

1. Crear un TRIGGER BEFORE DELETE sobre la tabla EMPLOYEES que impida borrar un registro si su JOB_ID es algo relacionado con CLERK
2. Crear una tabla denominada AUDITORIA con las siguientes columnas:

```
CREATE TABLE AUDITORIA (
USUARIO VARCHAR(50),
FECHA DATE,
SALARIO_ANTIGUO NUMBER,
SALARIO_NUEVO NUMBER);
```

3. Crear un TRIGGER BEFORE INSERT de tipo STATEMENT, de forma que cada vez que se haga un INSERT en la tabla REGIONS guarde una fila en la tabla AUDITORIA con el usuario y la fecha en la que se ha hecho el INSERT
4. Realizar otro trigger BEFORE UPDATE de la columna SALARY de tipo EACH ROW.
    - Si la modificación supone rebajar el salario el TRIGGER debe disparar un RAISE_APPLICATION_FAILURE “no se puede bajar un salario”.
    - Si el salario es mayor debemos dejar el salario antiguo y el salario nuevo en la tabla AUDITORIA.
5. Crear un TRIGGER BEFORE INSERT en la tabla DEPARTMENTS que al insertar un departamento compruebe que el código no esté repetido y luego que si el LOCATION_ID es NULL le ponga 1700 y si el MANAGER_ID es NULL le ponga 200
    -

Soluciones

1. Crear un TRIGGER BEFORE DELETE sobre la tabla EMPLOYEES que impida borrar un registro si su JOB_ID es algo relacionado con CLERK

```
CREATE OR REPLACE TRIGGER t1 BEFORE
DELETE ON employees FOR EACH ROW
BEGIN
IF
:old.job_id LIKE ( '%CLERK' )
THEN
raise_application_error(-20320,'NADA');
END IF;
END;
/
select * from employees;
delete from employees where job_id LIKE ('%CLERK');
```

2. Crear una tabla denominada AUDITORIA con las siguientes columnas:

```
CREATE TABLE AUDITORIA (
USUARIO VARCHAR(50),
FECHA DATE,
SALARIO_ANTIGUO NUMBER,
SALARIO_NUEVO NUMBER);
```

3. Crear un TRIGGER BEFORE INSERT de tipo STATEMENT, de forma que cada vez que se haga un INSERT en la tabla REGIONS guarde una fila en la tabla AUDITORIA con el usuario y la fecha en la que se ha hecho el INSERT

```
CREATE TRIGGER T2 BEFORE INSERT ON REGIONS
BEGIN
INSERT INTO AUDITORIA (usuario, fecha)
VALUES (user,sysdate);
END;
/
INSERT INTO REGIONS VALUES (20,'Prueba');
SELECT USER FROM DUAL;
select * from auditoria;
```

4. Realizar otro trigger BEFORE UPDATE de la columna SALARY de tipo EACH ROW.
   - Si la modificación supone rebajar el salario el TRIGGER debe disparar un RAISE_APPLICATION_FAILURE “no se puede bajar un salario”.
   - Si el salario es mayor debemos dejar el salario antiguo y el salario nuevo en la tabla AUDITORIA.

```
create or replace TRIGGER TRIGGER_salario
BEFORE UPDATE ON EMPLOYEES
FOR EACH ROW
BEGIN
   IF :NEW.SALARY< :OLD.SALARY THEN
      RAISE_APPLICATION_ERROR(-20000,'NO SE PUEDE BAJAR EL
SALARIO');
   END IF;
   IF :NEW.SALARY> :OLD.SALARY THEN
      INSERT INTO AUDITORIA VALUES
(USER,SYSDATE,:OLD.SALARY,:NEW.SALARY);
   END IF;
END;
```

5. Crear un TRIGGER BEFORE INSERT en la tabla DEPARTMENTS que al insertar un departamento compruebe que el código no esté repetido y luego que si el LOCATION_ID es NULL le ponga 1700 y si el MANAGER_ID es NULL le ponga 200

```
CREATE OR REPLACE TRIGGER TRIGGER1
BEFORE INSERT ON DEPARTMENTS
FOR EACH ROW
declare
deptno  number;
BEGIN
 select department_id into deptno from departments where
department_id=:new.department_id;
 RAISE_APPLICATION_ERROR(-20000,'Departamento ya existe');

 EXCEPTION
 WHEN NO_DATA_FOUND THEN
   if :new.location_id is null then
      :new.location_id:=1700;
   end if;
   if :new.manager_id is null then
      :new.manager_id:=200;
   end if;
END;
```