```
-- 1. REALIZA UN TRIGGER QUE LOGRE ASEGURARSE QUE LOS EQUIPOS NO PUEDEN CAMBIAR
DE CONFERENCIA
CREATE OR REPLACE TRIGGER no_cambia_conferencia
BEFORE UPDATE OF CONFERENCIA
ON equipos
FOR EACH ROW

BEGIN

   IF :OLD.CONFERENCIA <> :NEW.CONFERENCIA THEN
      raise_application_error(-20600, 'LOS EQUIPOS NO PUEDEN CAMBIAR DE CONFERENCIA');
   END IF;

END;


SELECT * FROM EQUIPOS;

UPDATE EQUIPOS
SET CONFERENCIA= 'Pp'
WHERE NOMBRE = 'Celtics';

-- 2. EN UN MISMO TRIGGER, IMPEDIR QUE UN EQUIPO JUEGUE CONTRA SI MISMO Y QUE UN
PARTIDO NO EMPATEN EN PUNTOS
CREATE OR REPLACE TRIGGER NO_SI_MISMO_NO_EMPATE
BEFORE INSERT
   OR UPDATE OF PUNTOS_LOCAL
   OR UPDATE OF PUNTOS_VISITANTE
   OR UPDATE OF EQUIPO_LOCAL
   OR UPDATE OF EQUIPO_VISITANTE
ON PARTIDOS
FOR EACH ROW

BEGIN

   IF :NEW.PUNTOS_LOCAL = :NEW.PUNTOS_VISITANTE THEN
      raise_application_error(-20600, 'LOS EQUIPOS NO PUEDEN EMPATAR');
   END IF;

   IF :NEW.EQUIPO_LOCALL = :NEW.EQUIPO_VISITANTE THEN
      raise_application_error(-20601, 'LOS EQUIPOS NO PUEDEN JUGAR CONTRA SI MISMOS');
   END IF;

END;

--3. Trigger para rellenar automáticamente el promedio de puntos por partido
--   de un jugador cuando se inserta en estadísticas, suponiendo que los puntos del partido se
reparten
```

```
--   equitativamente entre todos los jugadores.
CREATE OR REPLACE TRIGGER TR_RELLENA_PROMEDIO
BEFORE INSERT
ON ESTADISTICAS
FOR EACH ROW
DECLARE
   V_PUNTOS_TOTAL_EQ_JUGADOR NUMBER;
   V_PARTIDOS_JUGADOS NUMBER;
   V_JUG_EQ NUMBER;
   V_MEDIA_JUG NUMBER;

BEGIN
    -- CALCULAMOS LOS PUNTOS TOTALES DEL EQUIPO DEL JUGADOR POR TEMPORADA Y
    -- CALCULAMOS LOS NUMEROS DE PARTIDOS JUGADOS
    SELECT SUM(PUNTOS),SUM(NUMPARTIDOS) INTO
V_PUNTOS_TOTAL_EQ_JUGADOR,V_PARTIDOS_JUGADOS
    FROM (
      SELECT SUM(PA.PUNTOS_VISITANTE) AS PUNTOS,NVL(COUNT(PA.CODIGO),0) AS
NUMPARTIDOS
      FROM PARTIDOS PA, EQUIPOS E, JUGADORES J
      WHERE E.NOMBRE = PA.EQUIPO_VISITANTE
      AND E.NOMBRE = J.NOMBRE_EQUIPO
      AND J.CODIGO = :NEW.JUGADOR
      AND PA.TEMPORADA = :NEW.TEMPORADA
      UNION ALL
      SELECT SUM(PA.PUNTOS_LOCAL),NVL(COUNT(PA.CODIGO),0)
      FROM PARTIDOS PA, EQUIPOS E, JUGADORES J
      WHERE E.NOMBRE = PA.EQUIPO_LOCAL
      AND E.NOMBRE = J.NOMBRE_EQUIPO
      AND J.CODIGO = :NEW.JUGADOR
      AND PA.TEMPORADA = :NEW.TEMPORADA
      );

      -- SI NO HAY PUNTOS, PUES DIRECTAMENTE PONGO UN CERO
      IF (V_PUNTOS_TOTAL_EQ_JUGADOR = 0) THEN
        :NEW.PUNTOS_POR_PARTIDO := 0;
      ELSE
        -- NECESITO SABER CUANTOS JUGADORES TIENE EL EQUIPO
        SELECT COUNT(CODIGO) INTO V_JUG_EQ
        FROM JUGADORES
        WHERE NOMBRE_EQUIPO = (SELECT NOMBRE_EQUIPO FROM JUGADORES WHERE
CODIGO = :NEW.JUGADOR);

        -- CALCULAMOS LA MEDIA DE PUNTOS POR JUGADORES
        V_MEDIA_JUG :=
((V_PUNTOS_TOTAL_EQ_JUGADOR/V_PARTIDOS_JUGADOS)/V_JUG_EQ);

        :NEW.PUNTOS_POR_PARTIDO := V_MEDIA_JUG;
```

```
      END IF;
END;


CREATE OR REPLACE TRIGGER TR_RELLENA_PROMEDIO_CON_FUNCION
BEFORE INSERT
ON ESTADISTICAS
FOR EACH ROW
BEGIN
  :NEW.PUNTOS_POR_PARTIDO := F_PROMEDIO_JUGADOR(:new.jugador, :new.temporada);
END;



-- 6. Crear un trigger en pedidos de tal forma que, al insertar el pedido, debemos asegurarnos
-- qué el cliente tiene la misma ciudad que el empleado que le da el pedido. (2 puntos).

CREATE OR REPLACE TRIGGER TR_MISMA_CIUDAD
BEFORE
INSERT
ON PEDIDOS
FOR EACH ROW
DECLARE
   V_CIU_EMP OFICINAS.CIUDAD%TYPE;
   V_CIU_CLI CLIENTES.CIUDAD%TYPE;
BEGIN

   -- CIUDAD CLIENTE
   SELECT C.CIUDAD, O.CIUDAD INTO V_CIU_CLI, V_CIU_EMP
   FROM CLIENTES C, EMPLEADOS E, OFICINAS O
   WHERE C.CODIGOEMPLEADOREPVENTAS = E.CODIGOEMPLEADO
   AND E.CODIGOOFICINA = O.CODIGOOFICINA
   AND C.CODIGOCLIENTE = :NEW.CODIGOCLIENTE;

   IF V_CIU_CLI <> V_CIU_EMP THEN
      raise_application_error (-20600,'UN CLIENTE NO PUEDE TENER DISTINTA CIUDAD QUE SU
EMPLEADO QUE LO ATIENDE');
   END IF;

END;



-- OBLIGAR QUE LOS EMPLEADOS, NO COBREN MENOS DE 500 EUROS.
-- NO_COBRA_MENOS_500

CREATE OR REPLACE TRIGGER NO_COBRA_MENOS_500
BEFORE
```

```
UPDATE OF SAL OR INSERT
ON EMP
FOR EACH ROW --VA DE LA MANO CON LAS VARIABLES :NEW Y :OLD
BEGIN

   IF :NEW.SAL <= 500 THEN
      raise_application_error(-20600,'NO PUEDE HABER UN SUELDO MENOR A 500');
   END IF;


END;

INSERT INTO EMP
VALUES (68,'RAUL','STUDENT',NULL, NULL, 300, 0, 10);

UPDATE EMP
   SET COMM =200
WHERE EMPNO = 7934;




CREATE OR REPLACE TRIGGER NO_CAMBIO_LOC_DEPT
BEFORE
UPDATE ON DEPT
FOR EACH ROW
--DECLARE
BEGIN

   IF :NEW.LOC <> :OLD.LOC THEN
      raise_application_error(-20600,'NO SE PUEDE CAMBIAR LA LOCALIZACIÓN DE UN
DEPARTAMENTO');
   END IF;
END;


INSERT INTO DEPT VALUES (50,'COLEGIO','SEVILLA');

UPDATE DEPT
   SET DNAME = 'COLE', LOC ='MADRID'
WHERE DEPTNO = 50;
```

Ejercicio :
Sobre la tabla Pedidos. Crear un trigger que no permita a un cliente realizar
más de un pedido en un mismo día.

```
CREATE OR REPLACE TRIGGER UN_PEDIDO_DIA
```

```
BEFORE
INSERT OR UPDATE OF FECHAPEDIDO OR UPDATE OF CODIGOCLIENTE
ON PEDIDOS
FOR EACH ROW
DECLARE
   V_NUM_PEDIDO INTEGER:= 0;
BEGIN
   SELECT COUNT(*) INTO V_NUM_PEDIDO
   FROM PEDIDOS
   WHERE TRUNC(FECHAPEDIDO) = TRUNC(SYSDATE)
   AND CODIGOCLIENTE = :NEW.CODIGOCLIENTE;

   IF V_NUM_PEDIDO >= 1 THEN
      raise_application_error (-20600,'UN CLIENTE NO PUEDE TENER MÁS DE UN PEDIDO EN UN
DIA');
    END IF;
END;

INSERT INTO PEDIDOS(CODIGOPEDIDO, CODIGOCLIENTE, FECHAPEDIDO, FECHAESPERADA,
ESTADO)
VALUES (128,1, SYSDATE,SYSDATE+3,'RAUL');
_____

CREATE OR REPLACE TRIGGER NO_CAMBIO_LOC_DEPT
BEFORE
UPDATE ON DEPT
FOR EACH ROW
--DECLARE
BEGIN

   IF :NEW.LOC <> :OLD.LOC THEN
      raise_application_error(-20600,'NO SE PUEDE CAMBIAR LA LOCALIZACIÓN DE UN
DEPARTAMENTO');
   END IF;
END;


INSERT INTO DEPT VALUES (50,'COLEGIO','SEVILLA');

UPDATE DEPT
   SET DNAME = 'COLE', LOC ='MADRID'
WHERE DEPTNO = 50;

-- OBLIGAR QUE LOS EMPLEADOS, NO COBREN MENOS DE 500 EUROS.
-- NO_COBRA_MENOS_500

CREATE OR REPLACE TRIGGER NO_COBRA_MENOS_500
BEFORE
```

```
UPDATE OF SAL OR INSERT
ON EMP
FOR EACH ROW --VA DE LA MANO CON LAS VARIABLES :NEW Y :OLD
BEGIN

   IF :NEW.SAL <= 500 THEN
      raise_application_error(-20600,'NO PUEDE HABER UN SUELDO MENOR A 500');
   END IF;


END;

INSERT INTO EMP
VALUES (68,'RAUL','STUDENT',NULL, NULL, 300, 0, 10);

UPDATE EMP
   SET COMM =200
WHERE EMPNO = 7934;


-- Ejercicio 1
-- Haz un trigger llamado NoToquesManolo que impida al usuario MANOLO que cambie el
sueldo
-- de los empleados que trabajan en DALLAS.
CREATE OR REPLACE TRIGGER NoToquesManolo
BEFORE
UPDATE OF SAL OR INSERT OR UPDATE
ON EMP
FOR EACH ROW --VA DE LA MANO CON LAS VARIABLES :NEW Y :OLD
DECLARE
   V_DEPTNO DEPT.DEPTNO%TYPE;

BEGIN

   SELECT DEPTNO INTO V_DEPTNO
   FROM DEPT
   WHERE UPPER(LOC) = 'DALLAS';

   IF UPDATING THEN
      IF UPPER(USER) = 'MANOLO' AND :OLD.DEPTNO = V_DEPTNO THEN
         raise_application_error(-20600,'MANOLO QUE TE VEO');
      END IF;
   END IF;

   IF (DELETING AND :OLD.DEPTNO = V_DEPTNO) OR (INSERTING AND
:NEW.DEPTNO = V_DEPTNO) THEN
      raise_application_error(-20600,'MANOLO QUE TE VEO');
   END IF;
```

```
END;

------------------------
CREATE TABLE DEPT_COPIA AS SELECT * FROM DEPT;
CREATE TABLE DEPT_COPIA2 AS SELECT * FROM DEPT;

-- CREAR UN TRIGGER QUE MANTEGAN SINCRONIZADA LA TABLA EMP_COPIA CON
LA TABLA EMP_COPIA2;

CREATE OR REPLACE TRIGGER SINCRONIZAR_EMP_COPIAS
AFTER
UPDATE OR INSERT OR UPDATE
ON DEPT_COPIA
FOR EACH ROW --VA DE LA MANO CON LAS VARIABLES :NEW Y :OLD
BEGIN

   IF DELETING THEN
      DELETE FROM DEPT_COPIA2
      WHERE DEPTNO = :OLD.DEPTNO;
   END IF;

   IF INSERTING THEN
      INSERT INTO DEPT_COPIA2
      VALUES
         (:NEW.DEPTNO, :NEW.DNAME, :NEW.LOC);
    END IF;

    IF UPDATING THEN
      UPDATE DEPT_COPIA2
         SET DEPTNO = :NEW.DEPTNO,
            DNAME = :NEW.DNAME,
            LOC = :NEW.LOC
      WHERE DEPTNO = :OLD.DEPTNO;

    END IF;

END;


CREATE OR REPLACE PACKAGE GEOMETRIA IS
   PROCEDURE ROMBO (N INTEGER);
END GEOMETRIA;

CREATE OR REPLACE PACKAGE BODY GEOMETRIA IS
   PROCEDURE ROMBO(N INTEGER)
   IS
   ESTRELLA VARCHAR2(100):='*';
```

```
    ASTERISCO VARCHAR2(10):='*';
    ESPACIO VARCHAR2(100);
    CONTADOR INTEGER :=0;
    HUECO VARCHAR2(10):='-';
    BEGIN
        for k in 1..N-1 loop
            espacio:=espacio||hueco;
        end loop;

        DBMS_OUTPUT.PUT_LINE(ESPACIO||ESTRELLA||ESPACIO);

        WHILE LENGTH(ESTRELLA)<(2*N-1) AND LENGTH(ESPACIO)>0 LOOP
            ESTRELLA:=ASTERISCO||ESTRELLA||ASTERISCO;
            ESPACIO:=SUBSTR(ESPACIO,1,LENGTH(ESPACIO)-1);
            DBMS_OUTPUT.PUT_LINE(ESPACIO||ESTRELLA||ESPACIO);
        END LOOP;

        FOR K IN REVERSE 1..N-1 LOOP
            ESPACIO:=HUECO||ESPACIO;
            ESTRELLA:=SUBSTR(ESTRELLA,2,LENGTH(ESTRELLA)-2);
            DBMS_OUTPUT.PUT_LINE(ESPACIO||ESTRELLA||ESPACIO);
        END LOOP;
    END ROMBO;
END GEOMETRIA;
/

BEGIN
GEOMETRIA.ROMBO(5);
END;

-- Ejercicio 2
-- Haz un trigger llamado DeptSiempreLLeno que impida que un departamento se quede sin
empleados.
CREATE OR REPLACE TRIGGER deptSiempreLleno
BEFORE
DELETE OR UPDATE OF DEPTNO
ON EMP
FOR EACH ROW
DECLARE
    PRAGMA autonomous_transaction;
    V_NUM_EMP_DEPTNO emp.deptno%type;
BEGIN

     -- CONTAR LOS EMPLEADOS DE ESE DEPARTAMENTO
     -- Y SI ME QUEDA UNO O MENOS ANTES DE BORRAR, PUES PARA LA
EJECUCCIÓN
     select count(EMPNO) into V_NUM_EMP_DEPTNO
     from emp
```

```
     where deptno = :OLD.deptno;

   IF V_NUM_EMP_DEPTNO <= 1 THEN
      raise_application_error(-20621, ' No se puede dejar el departamento vacio ');
   END IF;
END;

SELECT *
FROM EMP;

INSERT INTO EMP (EMPNO, ENAME,JOB,DEPTNO) VALUES (33,'RAUL','SERENO',40);

DELETE FROM EMP
WHERE DEPTNO = 40;

-- Haz un trigger que le suba un 10% el sueldo a los empleados cuando cambia la localidad
donde trabajan.
CREATE OR REPLACE TRIGGER SUBE10
AFTER
UPDATE OF LOC
ON DEPT
FOR EACH ROW
DECLARE
BEGIN
   IF :OLD.LOC <> :NEW.LOC THEN
      UPDATE EMP
         SET  SAL = SAL *1.10
      WHERE DEPTNO = :NEW.DEPTNO;
   END IF;
END;

-- Ejercicio 3
-- Haz un trigger que controle si los sueldos están en los siguientes rangos:
-- CLERK: 800 – 1100
-- ANALYST: 1200 – 1600
-- MANAGER:1800 – 2000

CREATE OR REPLACE TRIGGER RANGO_SUELDO
BEFORE
INSERT OR UPDATE OF SAL OR UPDATE OF JOB
ON EMP
FOR EACH ROW
BEGIN
   IF UPPER(:NEW.JOB) = 'CLERK' AND (:NEW.SAL > 1100 OR :NEW.SAL <800) THEN
      raise_application_error (-20600,:NEW.SAL||' NO ES UN SUELDO VALIDO PARA
'||:NEW.JOB);
   END IF;
```

```
    IF UPPER(:NEW.JOB) = 'ANALYST' AND (:NEW.SAL > 1600 OR :NEW.SAL <1200)
THEN
       raise_application_error (-20600,:NEW.SAL||' NO ES UN SUELDO VALIDO PARA
'||:NEW.JOB);
   END IF;

   IF UPPER(:NEW.JOB) = 'MANAGER' AND (:NEW.SAL > 2000 OR :NEW.SAL <1800)
THEN
       raise_application_error (-20600,:NEW.SAL||' NO ES UN SUELDO VALIDO PARA
'||:NEW.JOB);
   END IF;
END;

SELECT *
FROM EMP;

UPDATE EMP
SET JOB = 'CLERK'
   ,SAL = 2000
WHERE EMPNO = 33;

-- TRIGGER EN PEDIDOS, QUE SI NO TIENE COMENTARIOS, PONGA SIN COMENTARIOS, AL
ACTUAR SOBRE LA TABLA
SELECT *
FROM PEDIDOS
ORDER BY 1 DESC;

INSERT INTO PEDIDOS(CODIGOPEDIDO, CODIGOCLIENTE, FECHAPEDIDO, FECHAESPERADA,
ESTADO,COMENTARIOS)
VALUES (180,1, SYSDATE+2,SYSDATE+3,'ESTADO',NULL);


CREATE OR REPLACE TRIGGER COMENTARIO_OBLIGADO
BEFORE
INSERT OR UPDATE
ON PEDIDOS
FOR EACH ROW
BEGIN
   IF :NEW.COMENTARIOS IS NULL THEN
      :NEW.COMENTARIOS := 'SIN COMENTARIOS';
   END IF;
END;
UPDATE PEDIDOS
SET ESTADO = 'RAULIN'
WHERE CODIGOPEDIDO = 129;
```