PRACTICA 5. TRIGGERS

2. Divida la tabla en dos, una tabla llamada INFO_PERSONAL con los atributos personales del ciclista (ID, NOMBRE, NACIONALIDAD) y otra tabla llamada INFO_PROFESIONAL con los atributos de su equipo (ID, EQUIPO, PAIS, CATEGORIA). Use el atributo ID como enlace en ambas tablas, definiéndolo como clave primaria en las dos.

```
CREATE TABLE INFO_PERSONAL AS (SELECT id,name, nationality FROM TOUR);
ALTER TABLE INFO_PERSONAL ADD PRIMARY KEY(id));

CREATE TABLE INFO_PROFESIONAL AS (SELECT id,team, category FROM TOUR);
ALTER TABLE INFO_PROFESIONAL ADD PRIMARY KEY(id));

--OPCION B
CREATE TABLE INFO_PERSONAL(
  id NUMBER(3,0) primary key,
  nombre VARCHAR2(50)
  nacionalidad VARCHAR2(3));
```

3. Cree una vista TOUR_VIEW que permita a las aplicaciones poder seguir viendo un solo objeto, como estaba la tabla original. Es decir, cree una vista para que la consulta SELECT * FROM TOUR_VIEW produzca el mismo resultado que producía antes SELECT * FROM TOUR. De este modo se garantiza la independencia de los datos.

```
CREATE VIEW TOUR_VIEW AS
(SELECT per.id, pro.equipo, pro.pais, per.nombre, per.nacionalidad, pro.categoria
FROM INFO_PERSONAL per JOIN INFO_PERSONAL pro USING (id));
```

4. Sin embargo la independencia se debería poder asegurar también en inserción. Cree un disparador llamado TR_INSERTA_CICLISTA que recibe la petición de insertar una tupla nueva en TOUR_VIEW y produce dos inserciones, una por cada tabla actual. Observe que el disparador ha de ser de INSTEAD OF.

```
CREATE OR REPLACE TRIGGER TR_INSERTA_CICLISTA INSTEAD OF
INSERT ON TOUR_VIEW FOR EACH ROW
BEGIN
INSERT INTO INFO_PERSONAL VALUES (:NEW.id, :NEW.nombre, :NEW.nacionalidad)
INSERT INTO INFO_PROFESIONAL VALUES (:NEW.id, :NEW.equipo, :NEW.pais, :NEW.pais, :NEW.categoria);
END
```

5. Cree una tabla llamada LOG_INSERCION que almacena las accciones de INSERT sobre la tabla TOUR. La idea es que las aplicaciones que aún hacían inserciones a través de la tabla antigua sean detectadas. La tabla tendrá dos campos, USUARIO y FECHA, guardando así el usuario ORACLE que pide la inserción y el instante en que se hace. Es decir, cuando una aplicación pida hacer un INSERT en TOUR, se almacena en LOG_INSERCION desde qué usuario ORACLE se hace la petición y su

fecha completa, para que el administrador detecte estas aplicaciones y pueda avisar a los programadores de que cambien la tabla por la nueva vista.

```
CREATE TABLE LOG_INSERCION(USUARIO VARCHAR2(19), FECHA DATE);
CREATE OR REPLACE TRIGGER TR_LOG_INSERCION
AFTER INSERT ON TOUR
BEGIN
   INSERT INTO LOG_INSERCION VALUES(USER, CURRENT_DATE);
END;
```

6. Volvamos a la tabla TOUR. Cree una vista llamada TOUR_SPAIN con los ciclistas españoles Y una vista llamada TOUR_ITALY con los ciclistas italianos (Nacionalidad).

```
CREATE VIEW TOUR_ITALY AS
SELECT * FROM TOUR WHERE UPPER(NACIONALITY) = 'ITA'

CREATE VIEW TOUR_SPAIN AS
SELECT * FROM TOUR WHERE UPPER(NACIONALITY) = 'ESP'
```

7. Pruebe a hacer una inserción de un nuevo ciclista español a través de la vista TOUR_SPAIN.

```
INSERT INTO TOUR_SPAIN  VALUES (1, 'TEAM CHERRY', 'Spain', 'CARLOS
JIMENEZ','ESP', 'WorldTour')
```

8. ¿Y si nuevo ciclista no es español, puede insertar a través de ella? Pruébelo ¿Puede verlo?

```
INSERT INTO TOUR_SPAIN  VALUES (1, 'TEAM CHERRY', 'Spain', 'CARLOS
JIMENEZ','ITA', 'WorldTour')
```

9. ¿Cómo la modificaría para que no se pueda insertar información que no puede luego verse? Revise en el manual la clausula CHECK OPTION en la creación de vistas.

```
CREATE VIEW TOUR_SPAIN AS SELECT* FROM TOUR
WHERE UPPER(NATIONALITY)='ESP'
WITH CHECK OPTION

CREATE VIEW TOUR_ITALY AS
SELECT * FROM TOUR WHERE UPPER(NACIONALITY) = 'ITA'
WITH CHECK OPTION
```

10.¿Esta modificación es para cualquier operación de actualización o puede decirse que funcione en INSERT y no en UPDATE? ¿Hay alguna manera de hacer esta especie de filtro según la operación? No lo haga, pero indique cómo se haría.

```
Si
```

11.Modifique la vista TOUR_SPAIN para que sea de sólo lectura y pruebe a insertar una fila a través de ella.

```
CREATE OR REPLACE VIEW "UBD3522"."TOUR_SPAIN" ("ID", "TEAM",
"COUNTRY", "NAME", "NATIONALITY", "CATEGORY") AS
select"ID","TEAM","COUNTRY","NAME","NATIONALITY","CATEGORY"from TOUR
where UPPER(NATIONALITY) = 'ESP';
```

12.Cree una vista llamada V_TOUR_FRANCE con los ciclistas franceses. Ceda permiso de borrado sobre la vista V_TOUR_FRANCE a otro esquema (pídale el usuario a un compañero).

```
CREATE VIEW V_TOUR_FRANCE
AS
SELECT * FROM TOUR WHERE UPPER(NACIONALITY) = 'ITA'
GRANT DELETE ON V_TOUR_FRANCE
TO UBD3522
```

13.Borre un piloto francés desde ese esquema. Borre un piloto no francés.

```
DELETE FROM TOUR_VIEW WHERE nationality = 'Fra'
DELETE FROM TOUR_VIEW WHERE nationality != 'Fra'
```

14. Cree una vista CICLISTAS_POR_EQUIPOS donde para cada equipo dice el nombre del equipo (NOMBRE) y su número de ciclistas(NUM_CICLISTAS). ¿Puede insertar sobre ella? ¿Porqué?

```
CREATE VIEW CICLISTAS_POR_EQUIPOS
AS
SELECT TEAM nombre, COUNT(*) num_ciclistas
FROM TOUR
GROUP BY TEAM
```

15. Cree una tabla llamada TB_CICLISTAS_POR_EQUIPOS con la misma información que la vista anterior. Cree un disparador llamado TR_CICLISTAS_POR_EQUIPOS que cada vez que se inserta un nuevo ciclista en la tabla TOUR, suma en la tabla TB_CICLISTAS_POR_EQUIPOS una unidad al atributo NUM_CICLISTAS de la tupla correspondiente al equipo.

```
CREATE VIEW TB_CICLISTAS_POR_EQUIPOS AS (SELECT team NOMBRE, COUNT(*) NUM_CICLISTAS FROM TOUR GROUP BY team);

CREATE TRIGGER TR_CICLISTAS_POR_EQUIPOS
AFTER INSERT ON TOUR FOR EACH ROW
BEGIN
    UPDATE TB_CICLISTAS_POR_EQUIPOS SET NUM_CICLISTAS = :OLD.NUM_CICLISTAS + 1
        WHERE NOMBRE = :NEW.TOUR.
END TR_CICLISTAS_POR_EQUIPOS;
```