Tabla de prueba:

HR:

```
CREATE TABLE PRUEBAS (
TABLA VARCHAR2(15),
USUARIO VARCHAR2(15),
OPERACION VARCHAR2(10),
FECHA DATE);
```

Ejemplo 1:

```
CREATE OR REPLACE TRIGGER T1 AFTER INSERT OR DELETE OR UPDATE ON DEPARTMENTS
BEGIN

  DBMS_OUTPUT.PUT_LINE('TABLA DEPARTMENTS CAMBIADA');

END;
```

```
UPDATE DEPARTMENTS
SET SUMASALARIAL = 0 WHERE SUMASALARIAL IS NULL
```

Ejemplo 2:

```
CREATE OR REPLACE TRIGGER T2 AFTER INSERT OR DELETE OR UPDATE ON DEPARTMENTS
BEGIN

  IF INSERTING THEN
  DBMS_OUTPUT.PUT_LINE('INSERTADA FILA EN TABLA DEPARTMENTS.');
  ELSE
    IF UPDATING THEN
    DBMS_OUTPUT.PUT_LINE('FILA ACTUALIZADA EN TABLA DEPARTMENTS.');
    ELSE
    DBMS_OUTPUT.PUT_LINE('FILA BORRADA EN TABLA DEPARTMENTS.');
    END IF;
  END IF;

END;
```

Crear T3 que inserte una fila de la tabla PRUEBA que imprima las operaciones de la tabla REGIONS.

```
CREATE OR REPLACE TRIGGER T3 AFTER INSERT OR UPDATE OR DELETE ON REGIONS
BEGIN
  IF INSERTING THEN
  INSERT INTO PRUEBAS VALUES ('REGIONS', USER, 'INSERT', SYSDATE);
  DBMS_OUTPUT.PUT_LINE('INSERTADA FILA EN TABLA REGIONS. TABLA PRUEBAS
ACTUALIZADA.');
  ELSE
    IF UPDATING THEN
    INSERT INTO PRUEBAS VALUES ('REGIONS', USER, 'UPDATE', SYSDATE);
    DBMS_OUTPUT.PUT_LINE('FILA ACTUALIZADA EN TABLA REGIONS. TABLA PRUEBAS
ACTUALIZADA.');
    ELSE
    INSERT INTO PRUEBAS VALUES ('REGIONS', USER, 'DELETE', SYSDATE);
    DBMS_OUTPUT.PUT_LINE('FILA BORRADA EN TABLA REGIONS. TABLA PRUEBAS
ACTUALIZADA.');
    END IF;
  END IF;
END;
INSERT INTO REGIONS VALUES (6, 'Oceania');
```

Versión 2:

```
CREATE OR REPLACE TRIGGER T3_1 AFTER INSERT OR DELETE OR UPDATE
ON REGIONS
DECLARE
OPERACION VARCHAR2(8);

BEGIN

 IF INSERTING THEN
  OPERACION:='INSERT';
 ELSE
   IF UPDATING THEN
   OPERACION:='UPDATE';
   ELSE
    OPERACION:='DELETE';
   END IF;
 END IF;

 INSERT INTO PRUEBAS
 VALUES ('REGIONS', USER, OPERACION, SYSDATE);

END;
```

Versión 3 (disparador por fila, no por tabla):

```
CREATE OR REPLACE TRIGGER T4 AFTER INSERT OR DELETE OR UPDATE
ON REGIONS FOR EACH ROW
DECLARE
OPERACION VARCHAR2(8);

BEGIN

 IF INSERTING THEN
  OPERACION:='INSERT';
 ELSE
   IF UPDATING THEN
   OPERACION:='UPDATE';
   ELSE
    OPERACION:='DELETE';
   END IF;
 END IF;

 INSERT INTO PRUEBAS
 VALUES ('REGIONS', USER, OPERACION, SYSDATE);

END;
```

Escribe un disparador a nivel de fila que salte si se modifica EMPLOYEES y que emita un mensaje en pantalla.

```
CREATE OR REPLACE TRIGGER T5 AFTER UPDATE ON EMPLOYEES
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('SE HA TOCADO EL SALARIO DEL EMPLEADO
'||:NEW.EMPLOYEE_ID);
    DBMS_OUTPUT.PUT_LINE('ANTES COBRABA '||:OLD.SALARY||' Y AHORA COBRA
'||:NEW.SALARY);

END;
```

```
UPDATE EMPLOYEES
SET SALARY = SALARY+1000
WHERE EMPLOYEE_ID=100;
```

Siempre que se inserte una región, el nombre debe ir en mayúsculas.

```
CREATE OR REPLACE TRIGGER T6 BEFORE INSERT ON REGIONS
FOR EACH ROW
BEGIN

 :NEW.REGION_NAME:=UPPER(:NEW.REGION_NAME);
 END;
```

El nombre de un departamento no se puede modificar. (sin alerta al usuario)

```
CREATE OR REPLACE TRIGGER T7 BEFORE UPDATE ON DEPARTMENTS
FOR EACH ROW
BEGIN

  :NEW.DEPARTMENT_NAME:= :OLD.DEPARTMENT_NAME;

END;
```

```
UPDATE DEPARTMENTS
SET DEPARTMENT_NAME='HOLA';
```

Versión 2 (con alerta al usuario):

```
CREATE OR REPLACE TRIGGER T8 BEFORE UPDATE ON DEPARTMENTS
FOR EACH ROW
BEGIN

  IF :NEW.DEPARTMENT_NAME!= :OLD.DEPARTMENT_NAME THEN
  RAISE_APPLICATION_ERROR(-20001, 'NO SE PUEDE CAMBIAR EL NOMBRE DE
DEPARTAMENTO.');
  END IF;

END;
```

UPDATE → :NEW y :OLD
INSERT → :NEW
DELETE → :OLD

```
ALTER TABLE PRUEBAS ADD DESCRIPCION VARCHAR2(50);
```

Escribe un disparador que almacene en PRUEBAS los datos de COUNTRIES que se borren.

```
CREATE OR REPLACE TRIGGER T9 AFTER DELETE ON COUNTRIES
FOR EACH ROW
BEGIN

  INSERT INTO PRUEBAS VALUES ('COUNTRIES', USER, 'DELETE', SYSDATE,
(:OLD.COUNTRY_ID||' '||:OLD.COUNTRY_NAME||' '||:OLD.REGION_ID));

END;
```

```
DELETE FROM COUNTRIES WHERE COUNTRY_NAME='Zambia'
```

- En la tabla capítulo, la duración debe estar entre 20 y 120 minutos, lanzando un mensaje de error en caso de no cumplir el requisito.

```
CREATE OR REPLACE TRIGGER T1 BEFORE INSERT OR UPDATE ON CAPITULO
FOR EACH ROW
BEGIN
  IF :NEW.DURACION < 20 OR :NEW.DURACION > 120 THEN
  RAISE_APPLICATION_ERROR(-20002, 'LA DURACION DEBE ESTAR ENTRE 20 Y 120.');
  END IF;
END;
```

Solo en los trigger before insert puedo hacer una select en la tabla que protagoniza el trigger.

- Escribir un disparador que impida escribir títulos de capitulos repetidos de la misma serie.

```
CREATE OR REPLACE TRIGGER T2 BEFORE INSERT ON CAPITULO
FOR EACH ROW
DECLARE
CONTADOR NUMBER(1);
BEGIN
  SELECT COUNT(*) INTO CONTADOR
  FROM CAPITULO
  WHERE TITULO_ORIGINAL=:NEW.TITULO_ORIGINAL
  AND SERIE_ID=:NEW.SERIE_ID;

  IF CONTADOR > 0 THEN
  RAISE_APPLICATION_ERROR(-20003, 'NO PUEDE HABER 2 TITULOS REPETIDOS DE LA
MISMA SERIE.');
  END IF;
END;
```

```
INSERT INTO CAPITULO
VALUES ('BRGN',2, 2, 'Dyden i midten', 50);
```

- En la tabla reparto, debemos impedir que se repita el nombre de un personaje en la misma serie.

```
CREATE OR REPLACE TRIGGER T3 BEFORE INSERT ON REPARTO
FOR EACH ROW
DECLARE
CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM REPARTO
WHERE SERIE_ID=:NEW.SERIE_ID
AND PERSONAJE_NOMBRE=:NEW.PERSONAJE_NOMBRE;
```

```
IF CONTADOR > 0 THEN
RAISE_APPLICATION_ERROR(-20004, 'NO SE PUEDE REPETIR EL MISMO PERSONAJE EN LA
MISMA SERIE.');
END IF;
END;

INSERT INTO REPARTO
VALUES ('BRGN','Birgitte Nyborg Christensen', 'A001', 'Descripcion');
```

- Mantener la columna capítulos de la tabla serie actualizada.

```
CREATE OR REPLACE TRIGGER T4 AFTER INSERT OR DELETE ON CAPITULO
FOR EACH ROW
BEGIN

IF INSERTING THEN
  UPDATE SERIE
  SET CAPITULOS = CAPITULOS+1 WHERE SERIE_ID=:NEW.SERIE_ID;
  DBMS_OUTPUT.PUT_LINE('FILA CAPITULOS DE TABLA SERIE ACTUALIZADA.');
ELSE
  UPDATE SERIE
  SET CAPITULOS = CAPITULOS-1 WHERE SERIE_ID=:OLD.SERIE_ID;
  DBMS_OUTPUT.PUT_LINE('FILA CAPITULOS DE TABLA SERIE ACTUALIZADA.');
END IF;
END;

INSERT INTO CAPITULO
VALUES ('BRGN', 1, 4, 'Prueba3', 50);

ALTER TABLE SERIE ADD DURACION_TOTAL NUMBER(4);
```

- Para cada serie, cuantos minutos totales hay.

```
UPDATE SERIE
SET DURACION_TOTAL = (SELECT NVL(SUM(DURACION),0)
                      FROM CAPITULO
                      WHERE SERIE_ID=SERIE.SERIE_ID);
```

EJERCICIOS DE TRIGGERS EN PUBS

1. **Añade la columna EJEMPLARES_VENDIDIOS a la tabla PUBLISHER. Rellénala con el resultado de sumar las unidades vendidas de todos sus libros (ytd_sale).**

```
ALTER TABLE PUBLISHER ADD EJEMPLARES_VENDIDOS NUMBER(7);

UPDATE PUBLISHER
SET EJEMPLARES_VENDIDOS = (SELECT SUM(YTD_SALE)
                    FROM TITLE
                    WHERE PUB_ID=PUBLISHER.PUB_ID)

UPDATE PUBLISHER
SET EJEMPLARES_VENDIDOS = 0 WHERE EJEMPLARES_VENDIDOS IS NULL
```

2. **Escribe un disparador que mantenga actualizada la columna del ejercicio anterior.**

```
CREATE OR REPLACE TRIGGER T1 AFTER INSERT OR DELETE OR UPDATE ON TITLE
FOR EACH ROW
BEGIN

 IF INSERTING THEN
   UPDATE PUBLISHER
   SET EJEMPLARES_VENDIDOS = EJEMPLARES_VENDIDOS + :NEW.YTD_SALE WHERE
:NEW.PUB_ID=PUBLISHER.PUB_ID;
   DBMS_OUTPUT.PUT_LINE('TABLA PUBLISHER ACTUALIZADA.');
 ELSE
   IF UPDATING THEN
     UPDATE PUBLISHER
     SET EJEMPLARES_VENDIDOS = EJEMPLARES_VENDIDOS + :NEW.YTD_SALE -
:OLD.YTD_SALE WHERE :NEW.PUB_ID=PUBLISHER.PUB_ID;
     DBMS_OUTPUT.PUT_LINE('TABLA PUBLISHER ACTUALIZADA.');
   ELSE
     UPDATE PUBLISHER
     SET EJEMPLARES_VENDIDOS = EJEMPLARES_VENDIDOS - :OLD.YTD_SALE WHERE
:OLD.PUB_ID=PUBLISHER.PUB_ID;
     DBMS_OUTPUT.PUT_LINE('TABLA PUBLISHER ACTUALIZADA.');
   END IF;
 END IF;
END;

INSERT INTO TITLE VALUES ('PC4000', 'Prueba', 'popular_comp', 1622, 20, 8000, 10, 10000,
'Prueba 2', SYSDATE);
```

3. **Añade a la tabla TITLE la columna LAST_UPDATE. Rellénala con la fecha de hoy para todos los títulos.**

```
ALTER TABLE TITLE ADD LAST_UPDATE DATE;

UPDATE TITLE
SET LAST_UPDATE = SYSDATE;
```

4. **Escribe un disparador que modifique el valor de LAST_UPDATE cada vez que se produzca una modificación de la información de libros.**

```
CREATE OR REPLACE TRIGGER T2 BEFORE UPDATE OR INSERT ON TITLE
FOR EACH ROW
BEGIN

:NEW.LAST_UPDATE := SYSDATE;

END;

UPDATE TITLE
SET TITLE = 'HOLA' WHERE TITLE_ID='PC4000';
```

5. **La tabla ROYSCHED establece el porcentaje que deben ganar los autores en función de las ventas de títulos. Basándote en la información de esta tabla, se debe mantener actualizado el valor de la columna ROYALTY que corresponde a los autores.**

```
CREATE OR REPLACE TRIGGER T3 BEFORE INSERT OR UPDATE ON TITLE
FOR EACH ROW
DECLARE
CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM ROYSCHED
WHERE TITLE_ID=:NEW.TITLE_ID
AND :NEW.YTD_SALE BETWEEN LORANGE AND HIRANGE;

IF CONTADOR=1 THEN
  SELECT ROYALTY INTO :NEW.ROYALTY
  FROM ROYSCHED
  WHERE TITLE_ID=:NEW.TITLE_ID
  AND :NEW.YTD_SALE BETWEEN LORANGE AND HIRANGE;
END IF;
END;


UPDATE TITLE
SET YTD_SALE=12000 WHERE TITLE_ID='PS3333'
```

6. **En la tabla EMPLOYEEPUB el valor de la columna JOB_LVL debe estar comprendida siempre entre los valores min_lvl y max_lvl establecidos para el job_id correspondiente. Emite un error en caso de que se pretenda establecer un valor incoherente.**

```
create or replace TRIGGER T4 BEFORE UPDATE ON EMPLOYEEPUB
FOR EACH ROW
DECLARE
CONTADOR NUMBER(1);
BEGIN

SELECT COUNT(*) INTO CONTADOR
FROM JOB
WHERE JOB_ID=:NEW.JOB_ID
AND :NEW.JOB_LVL BETWEEN MIN_LVL AND MAX_LVL;

IF CONTADOR=0 THEN
  RAISE_APPLICATION_ERROR(-20005, 'NUEVO VALOR DE JOB_LVL NO PERMITIDO.');
END IF;
END;

UPDATE EMPLOYEEPUB
SET JOB_LVL = 300 WHERE JOB_ID=2
```

7. **En la tabla TITLE, el título de un libro se puede repetir siempre y cuando se publique en una editorial distinta. Escribe un disparador que emita un error cuando un título de libro se pretenda duplicar de forma errónea.**

```
CREATE OR REPLACE TRIGGER T5 BEFORE INSERT ON TITLE
FOR EACH ROW
DECLARE
CONTADOR NUMBER(1);
BEGIN


  SELECT COUNT(*) INTO CONTADOR
  FROM TITLE
  WHERE TITLE_ID=:NEW.TITLE_ID
  AND PUB_ID=:NEW.PUB_ID;

  IF CONTADOR=0 THEN
  RAISE_APPLICATION_ERROR(-20006, 'TITULO DE LA MISMA EDITORIAL YA EXISTE.');
  END IF;
END;
```

8. **El valor de la columna ADVANCE de la tabla TITLE no se puede modificar (sin emitir error).**

```
CREATE OR REPLACE TRIGGER T6 BEFORE UPDATE ON TITLE
FOR EACH ROW
WHEN (NEW.ADVANCE!=OLD.ADVANCE)
BEGIN

:NEW.ADVANCE:=:OLD.ADVANCE;

END;

UPDATE TITLE
SET ADVANCE= 10000 WHERE TITLE_ID='PC1035';
```

9. **Un titulo no puede cambiar de editorial (sin emitir error).**

```
CREATE OR REPLACE TRIGGER T7 BEFORE UPDATE ON TITLE
FOR EACH ROW
WHEN (NEW.PUB_ID!=OLD.PUB_ID)
BEGIN

:NEW.PUB_ID:=:OLD.PUB_ID;

END;

UPDATE TITLE
SET PUB_ID=0736 WHERE TITLE_ID='PC1035';
```

Vista con toda la informacion de COUNTRIES donde aparezca el nombre de region que corresponde.

```
CREATE OR REPLACE VIEW VISTA3
AS SELECT COUNTRY_ID, COUNTRY_NAME, REGION_NAME
FROM COUNTRIES JOIN REGIONS USING (REGION_ID)
```

```
SELECT * FROM VISTA3
```

Disparador sobre VISTA3 que modifique/inserte/borre filas a través de la vista3.

```
CREATE OR REPLACE TRIGGER T10
INSTEAD OF INSERT OR UPDATE OR DELETE ON VISTA3 FOR EACH ROW
DECLARE
REGION_COD REGIONS.REGION_ID%TYPE;
```

```
BEGIN
IF INSERTING THEN
  SELECT REGION_ID INTO REGION_COD
  FROM REGIONS
  WHERE REGION_NAME=:NEW.REGION_NAME;

  INSERT INTO COUNTRIES VALUES (:NEW.COUNTRY_ID, :NEW.COUNTRY_NAME,
REGION_COD);
ELSE
  IF UPDATING THEN
    SELECT REGION_ID INTO REGION_COD
    FROM REGIONS
    WHERE REGION_NAME=:NEW.REGION_NAME;

    UPDATE COUNTRIES
    SET COUNTRY_NAME=:NEW.COUNTRY_NAME, REGION_ID=REGION_COD
    WHERE COUNTRY_ID=:NEW.COUNTRY_ID;
  ELSE
    DELETE FROM COUNTRIES
    WHERE COUNTRY_ID= :OLD.COUNTRY_ID OR COUNTRY_NAME= :OLD.COUNTRY_NAME;

  END IF;
END IF;
END;

INSERT INTO VISTA3 VALUES ('EE', 'EEEEEE', 'ASIA');

UPDATE VISTA3 SET REGION_NAME='EUROPE' WHERE COUNTRY_ID='BB';
UPDATE VISTA3 SET COUNTRY_NAME='CCCCCC' WHERE COUNTRY_ID='BB';

DELETE FROM VISTA3 WHERE COUNTRY_NAME='DDDDDD';
DELETE FROM VISTA3 WHERE COUNTRY_ID='EE';
```

Crea una vista con el nombre, apellido y título de película en la que participan los actores.

```
CREATE OR REPLACE VIEW VISTA4
AS SELECT FIRST_NAME, LAST_NAME, TITLE
FROM ACTORES JOIN FILM_ACTOR USING (ACTOR_ID)
        JOIN FILM USING (FILM_ID);
```

Crear un trigger que inserte/borre

```
CREATE OR REPLACE TRIGGER T1
INSTEAD OF INSERT OR DELETE ON VISTA4 FOR EACH ROW
DECLARE
FILM_COD FILM.FILM_ID%TYPE;
ACTOR_COD ACTORES.ACTOR_ID%TYPE;
```

```
BEGIN
IF INSERTING THEN
  SELECT FILM_ID INTO FILM_COD
  FROM FILM
  WHERE TITLE= :NEW.TITLE;

  SELECT ACTOR_ID INTO ACTOR_COD
  FROM ACTORES
  WHERE FIRST_NAME= :NEW.FIRST_NAME AND LAST_NAME= :NEW.LAST_NAME;

  INSERT INTO FILM_ACTOR VALUES (ACTOR_COD, FILM_COD, SYSDATE);

ELSE
  SELECT FILM_ID INTO FILM_COD
  FROM FILM
  WHERE TITLE= :OLD.TITLE;

  SELECT ACTOR_ID INTO ACTOR_COD
  FROM ACTORES
  WHERE FIRST_NAME= :OLD.FIRST_NAME AND LAST_NAME= :OLD.LAST_NAME;

  DELETE FROM FILM_ACTOR
  WHERE FILM_ID=FILM_COD AND ACTOR_ID=ACTOR_COD;

END IF;
END;
```