BD_Ropa_createBD.sql

```
/* @BD_Ropa_createBD.sql */
-- SET ECHO ON
SET LINESIZE 132
SET PAGESIZE 200

/* Crear las tablas y alguna vista, a modo de ejemplo
*/
CREATE TABLE prendas (
  nombre     varchar2(24),
  color      varchar2(24),
  talla      number(2)
);

/* mostrar la información básica de las tablas y vistas
*/
@BD_Ropa_infoBD.sql
```

BD_Ropa_infoBD.sql

```
/* @BD_Ropa_infoBD.sql */
-- SET ECHO ON

/* mostrar la información básica de las tablas y vistas
*/
-- SELECT * FROM prenda;

DESCRIBE prendas;
```

BD_Ropa_insertData.sql

```
/* @BD_Ropa_insertData.sql */
-- SET ECHO ON

/* eliminar los datos existentes en las tablas
*/
@BD_Ropa_deleteData.sql

/* añadir los datos de prueba a las tablas
*/

INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'ROJO', 4);
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'AZUL', 5);
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'AZUL', 4);
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'ROJO', 5);
INSERT INTO Prendas(nombre, color, talla) VALUES('PANTALON', 'VERDE', 6);
INSERT INTO Prendas(nombre, color, talla) VALUES('PANTALON', 'ROJO', 6);
INSERT INTO Prendas(nombre, color, talla) VALUES('CALCETIN', NULL, 8);

COMMIT;

/* mostrar los datos de las tablas y vista
*/
@BD_Ropa_showData.sql
```

BD_Ropa_deleteData.sql

```
/* @BD_Ropa_deleteData.sql */
-- SET ECHO ON

/* eliminar los datos de las tablas
*/

DELETE FROM prendas;

/* mostrar los datos de las tablas y vista
*/
@BD_Ropa_showData.sql
```

BD_Ropa_showData.sql

```
/* @BD_Ropa_showData.sql */
-- SET ECHO ON
SET LINESIZE 132
SET PAGESIZE 200

column talla      format 99
column nombre     format a24
column color      format a16

SET WRAP OFF

/* mostrar los datos de las tablas y vista
*/
SELECT * FROM prendas;
```

BD_Ropa_dropBD.sql

```
/* @BD_Ropa_dropBD.sql */
-- SET ECHO ON

/* eliminar las tablas y vistas creadas, en el orden
   adecuado, para no violar las restricciones
*/
--DROP VIEW  DxT;
DROP TABLE prendas;

/* mostrar la información básica de las tablas y vistas
*/
@BD_Ropa_infoBD.sql
```

tst_clave.sql

```
/*
  @tst_clave.sql  version 1.1
  Santiago Velilla 17 Mayo 2014
  Dpto. Informática e Ingeniería de Sistemas, Universidad Zaragoza

  Este script muestra si, en la tabla indicada por el primer parámetro, el conjunto
  de atributos especificados por el segundo parámetro constituyen, o no, una clave.

  uso:
  @tst_clave.sql nombre_tabla atributos

  ejemplos:
  @tst_clave prendas nombre
  @tst_clave prendas "nombre, talla, color"
*/
SET ECHO OFF

PROMPT
PROMPT listado de tuplas con valores repetidos de (&2) en la tabla &1
PROMPT
-- listado de tuplas con valores repetidos
SELECT &2, count(*) veces FROM &1 GROUP BY &2 HAVING count(*) > 1;

PROMPT
PROMPT comprobando si (&2) es una clave de la tabla &1
PROMPT
SELECT '   NO    ' "es clave" FROM dual
 WHERE EXISTS (
  SELECT * FROM &1 GROUP BY &2 HAVING count(*) > 1
 )
UNION
SELECT '   SI    ' "es clave" FROM dual
 WHERE NOT EXISTS (
  SELECT * FROM &1 GROUP BY &2 HAVING count(*) > 1
 );
```

info_TABLA.sql

```
/*
  @info_TABLA.sql version 1.1
  Santiago Velilla 19 Mayo 2014
  Dpto. Informática e Ingeniería de Sistemas, Universidad Zaragoza

  Este script muestra información básica de la tabla cuyo nombre se especifica
  a través del parámetro del script (&1)

  uso:
  @info_TABLA nombre_tabla
*/
-- SET ECHO ON
SET linesize 132
SET pagesize 200
-- modificar el ancho para columnas de tipo LONG (p.e. el cuerpo de un trigger)
SET LONG 255
column NOMBRE           format A18
column OWNER            format A12
column CONSTRAINT_NAME  format A20
column TABLE_NAME       format A16
column SEARCH_CONDITION format A24
column COLUMN_NAME      format A16
column POSITION         format 99

DEFINE tabla = '&1'

----------------------------------------------------------------------
/*
la tabla del diccionario USER_CONS_COLUMNS tiene los siguientes atributos:
OWNER, CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME, POSITION
y especifica los atributos que intervienen en la restricciones definidas por
el usuario e identificadas por (OWNER, CONSTRAINT_NAME, TABLE_NAME)
ALL_CONS_COLUMNS es similar, pero para las restricciones accesibles al usuario
*/
-- describe USER_CONS_COLUMNS
-- describe ALL_CONS_COLUMNS
-- SELECT * FROM USER_CONS_COLUMNS WHERE table_name = UPPER('&tabla');

/*
la tabla del diccionario USER_CONSTRAINTS contiene las restricciones definidas
por el usuario. Tiene, entre otros, los siguientes atributos:
OWNER, CONSTRAINT_NAME y TABLE_NAME que identifican la restricción
CONSTRAINT_TYPE, SEARCH_CONDITION   que especifican la restricción

CONSTRAINT_TYPE  VARCHAR2(1)      Type of constraint definition:
    C (check constraint on a table)
    P (primary key)
    U (unique key)
    R (referential integrity)
    V (with check option, on a view)
    O (with read only, on a view)

ALL_CONSTRAINTS es similar, pero para las restricciones accesibles al usuario
*/
-- describe USER_CONSTRAINTS
-- describe ALL_CONSTRAINTS
-- SELECT table_name, constraint_name, constraint_type, search_condition
--  FROM user_constraints WHERE table_name = UPPER('&tabla');

PROMPT
PROMPT atributos de la tabla  &tabla
describe &tabla

PROMPT
PROMPT restricciones definidas sobre la tabla  &tabla
SELECT b.OWNER, a.TABLE_NAME, a.CONSTRAINT_NAME, a.CONSTRAINT_TYPE, COLUMN_NAME, POSITION,
SEARCH_CONDITION
  FROM USER_CONSTRAINTS a, USER_CONS_COLUMNS b
  WHERE (b.CONSTRAINT_NAME = a.CONSTRAINT_NAME) AND (a.TABLE_NAME = UPPER('&tabla'));
```

```
-- especificar formato de las columnas a mostrar
column TRIGGER_NAME      format a24
column TRIGGERING_EVENT  format a30
column TABLE_OWNER       format a12
column COLUMN_NAME       format a14
column REFERENCING_NAMES format a40
column WHEN_CLAUSE       format a20
column DESCRIPTION       format a55
column TRIGGER_BODY      format a85

PROMPT
PROMPT información básica de los disparadores definidos sobre la tabla  &tabla
PROMPT
-- mostrar todos los disparadores asociados a la tabla, su estado y descripción
SELECT TRIGGER_NAME, TABLE_OWNER, STATUS, DESCRIPTION
 FROM USER_TRIGGERS
 WHERE TABLE_NAME = UPPER('&tabla');

-- mostrar todos los disparadores asociados a la tabla con su tipo, evento y cláusula when
SELECT TRIGGER_NAME, TRIGGER_TYPE, TRIGGERING_EVENT, WHEN_CLAUSE
 FROM USER_TRIGGERS
 WHERE TABLE_NAME = UPPER('&tabla');

-- mostrar el programa definido para los disparadores asociados a la tabla
SELECT TRIGGER_NAME, TRIGGER_BODY
 FROM USER_TRIGGERS
 WHERE TABLE_NAME = UPPER('&tabla');
```

---

tst_DF.sql

```
SET ECHO OFF
/*
  @tst_DF.sql  version 1.1
  Santiago Velilla 17 Mayo 2014
  Dpto. Informática e Ingeniería de Sistemas, Universidad Zaragoza

  En este script se comprueba si los valores de la tabla especificada por el primer
  parámetro verifican, o no, la dependencia funcional X --> Y especificada a través
  del resto de parámetros.  Los parámetros de llamada son:
  &1 representa el nombre de la tabla en la que hay que hacer la comprobación
  &2 es el conjunto de atributos X que constituyen el determinante de la DF
  &3 es el conjunto de atributos Y que, supuestamente, dependen funcionalmente de X

  Para ello muestra el porcentaje de tuplas que no verifican la DF

  uso:
  @tst_DF.sql nombre_tabla atributos_X atributos_Y
*/
-----------------------------------------------------------------------

PROMPT
PROMPT porcentaje mínimo de tuplas de la tabla &1 que no verifican la DF &2 --> &3
PROMPT

SELECT numFallos/tot_tuplas*100 "% fallos"
 FROM
  (SELECT sum(tuplas) numFallos
   FROM (SELECT sum(veces)-max(veces) tuplas
          FROM (SELECT &2, &3, count(*) veces FROM &1 GROUP BY &2, &3)
          GROUP BY &2)
  ), (SELECT count(*) tot_tuplas FROM &1);
```

show_constraints.sql

```
/*
  @show_constraints.sql version 1.1
  Santiago Velilla 17 Abril 2014
  Dpto. Informática e Ingeniería de Sistemas, Universidad Zaragoza

  Este script muestra las restricciones definidas sobre la tabla especificada
  Los parámetros son:
  &1 es el nombre de la tabla de la que hay que mostrar las restricciones

  uso:
  @show_constraints nombre_tabla
*/
SET ECHO ON
SET linesize 200
SET pagesize 200
column OWNER            format A12
column CONSTRAINT_NAME  format A20
column TABLE_NAME       format A16
column SEARCH_CONDITION format A24
column COLUMN_NAME      format A16
column POSITION         format 99
----------------------------------------------------------------------
/*
la tabla del diccionario USER_CONS_COLUMNS tiene los siguientes atributos:
OWNER, CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME, POSITION
y especifica los atributos que intervienen en la restricciones definidas por
el usuario e identificadas por (OWNER, CONSTRAINT_NAME, TABLE_NAME)
ALL_CONS_COLUMNS es similar, pero para las restricciones accesibles al usuario
*/
-- describe USER_CONS_COLUMNS
-- describe ALL_CONS_COLUMNS
-- SELECT * FROM USER_CONS_COLUMNS WHERE table_name = UPPER('&1');

/*
la tabla del diccionario USER_CONSTRAINTS contiene las restricciones definidas
por el usuario. Tiene, entre otros, los siguientes atributos:
OWNER, CONSTRAINT_NAME y TABLE_NAME que identifican la restricción
CONSTRAINT_TYPE, SEARCH_CONDITION   que especifican la restricción

CONSTRAINT_TYPE  VARCHAR2(1)      Type of constraint definition:
    C (check constraint on a table)
    P (primary key)
    U (unique key)
    R (referential integrity)
    V (with check option, on a view)
    O (with read only, on a view)

ALL_CONSTRAINTS es similar, pero para las restricciones accesibles al usuario
*/
-- describe USER_CONSTRAINTS
-- describe ALL_CONSTRAINTS
-- SELECT table_name, constraint_name, constraint_type, search_condition
--  FROM user_constraints WHERE table_name = UPPER('&1');

SELECT b.OWNER, a.TABLE_NAME, a.CONSTRAINT_NAME, a.CONSTRAINT_TYPE, COLUMN_NAME, POSITION,
SEARCH_CONDITION
  FROM USER_CONSTRAINTS a, USER_CONS_COLUMNS b
  WHERE (b.CONSTRAINT_NAME = a.CONSTRAINT_NAME) AND (a.TABLE_NAME = UPPER('&1'));
```

probar_consultas.sql

```
/* @probar_consultas.sql */
SET echo ON
SELECT sysdate FROM dual;

-- crear la BD de la tienda de ropa (tabla prendas)
@BD_Ropa_createBD.sql

-- añadir algunas prendas para probar
@BD_Ropa_insertData.sql

-- mostrar las tuplas de prendas
SELECT * FROM prendas;

-- insertar algunas tuplas repetidas
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'ROJO', 4);
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'AZUL', 5);

-- mostrar las tuplas de prendas
SELECT * FROM prendas;

-- mostrar las tuplas de prendas diferentes, y su número
SELECT prendas.*, count(*) veces FROM prendas GROUP BY nombre, color, talla;

-- mostrar las tuplas de prendas repetidas, y su número
SELECT prendas.*, count(*) veces
 FROM prendas
 GROUP BY nombre, color, talla
 HAVING count(*) > 1;

-- mostrar cuántas prendas están repetidas (sólo el número)
SELECT count(count(*)) FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1;

-------------------------------------------------------------------------------------------
-- eliminar en la tabla prendas las tuplas repetidas (usando una vista)
CREATE VIEW tuplasSelec (idtupla) AS (
  SELECT min(rowid)
   FROM prendas
   GROUP BY nombre, color, talla
);

DELETE FROM prendas WHERE rowid NOT IN (SELECT idtupla FROM tuplasSelec);

DROP VIEW tuplasSelec;
-------------------------------------------------------------------------------------------

-- mostrar cuántas prendas están repetidas (sólo el número)
SELECT count(count(*)) FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1;

-- insertar de nuevo algunas tuplas repetidas
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'ROJO', 4);
INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'AZUL', 5);

-- mostrar cuántas prendas están repetidas (sólo el número)
SELECT count(count(*)) FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1;

-------------------------------------------------------------------------------------------
-- eliminar en la tabla prendas las tuplas repetidas (directamente)
DELETE FROM prendas
 WHERE rowid NOT IN (SELECT min(rowid) FROM prendas GROUP BY nombre, color, talla);
-------------------------------------------------------------------------------------------

-- mostrar cuántas prendas están repetidas (sólo el número)
SELECT count(count(*)) FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1;

-- comprobar si (nombre, color, talla) es una clave de prendas
SELECT '   SI    ' "es clave" FROM dual
 WHERE NOT EXISTS (
  SELECT * FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1
 )
UNION
SELECT '   NO    ' "es clave" FROM dual
 WHERE EXISTS (
  SELECT * FROM prendas GROUP BY nombre, color, talla HAVING count(*) > 1
 );
```

```
--=============================================================================
-- Si se quiere implementar una función parametrizada con los atributos a comprobar y la
-- tabla a utilizar, hay que utilizar SQL DINÁMICO, como se muestra a continuación:
--
-- La función es_CLAVE devuelve true si los atributos especificados por el primer parámetro
-- son clave de la tabla especificada por el segundo parámetro, y false en caso contrario.
CREATE OR REPLACE FUNCTION es_CLAVE (atrib IN varchar2, tabla IN varchar2)
RETURN boolean
IS
--  query_str VARCHAR2(200);
  totalRepetidas number;
BEGIN
--  query_str := 'SELECT count(count(*)) FROM '||tabla||' GROUP BY '||atrib||' HAVING count(*)>1';
--  EXECUTE IMMEDIATE query_str INTO totalRepetidas;
  EXECUTE IMMEDIATE
   'SELECT count(count(*)) from '||tabla||' GROUP BY '||atrib||' HAVING count(*) > 1'
   INTO totalRepetidas;
--   DBMS_OUTPUT.PUT_LINE('hay '||totalRepetidas||' tuplas ('||atrib||') repetidas en '||tabla);
  RETURN totalRepetidas = 0;
END es_CLAVE;
/
SHOW ERRORS FUNCTION es_CLAVE;

-- prueba de la función implementada (no existe en SQL*PLUS el tipo boolean)
-- habilitar la salida de mensajes
SET SERVEROUTPUT ON
-- bloque PL/SQL para probar la función
BEGIN
  IF es_CLAVE('nombre, color, talla', 'prendas') THEN
    DBMS_OUTPUT.PUT_LINE('(nombre, color, talla) SI es una clave de prendas');
  ELSE
    DBMS_OUTPUT.PUT_LINE('(nombre, color, talla) NO es una clave de prendas');
  END IF;
  IF es_CLAVE('nombre, color', 'prendas') THEN
    DBMS_OUTPUT.PUT_LINE('(nombre, color) SI es una clave de prendas');
  ELSE
    DBMS_OUTPUT.PUT_LINE('(nombre, color) NO es una clave de prendas');
  END IF;
END;
/

-- eliminar la función
DROP FUNCTION es_CLAVE;

-- probar el script "tst_clave"
@tst_clave prendas "nombre, talla, color"
@tst_clave prendas "nombre, talla"
@tst_clave prendas "nombre, color"
@tst_clave prendas "talla, color"

-- obsérvese que con los datos del ejemplo también es clave, por casualidad, (talla, color)
-- para no permitir tuplas repetidas, se define la clave candidata correspondiente
-- (UNIQUE) ya que, como color y talla pueden tomar valores nulos, no es posible
-- definir una clave primaria (PRIMARY KEY)
ALTER TABLE prendas ADD CONSTRAINT prendas_UN UNIQUE(nombre, color, talla);

/*
  OBTENCIÓN DE DEPENDENCIAS FUNCIONALES
*/
-- como prueba previa para ver si nombre -> color se van a obtener las prendas de las que
-- hay más de 1 color, junto con el número de colores hay de cada prenda
SELECT nombre, count(*) veces
 FROM prendas
 GROUP BY nombre
 HAVING count(*) > 1;

-- se observa que la consulta anterior no proporciona el resultado deseado porque
-- hay pares de valores (nombre, color) repetidos.
-- Se repite la prueba contando los colores que hay de cada prenda
SELECT nombre, count(DISTINCT color) veces
 FROM prendas
 GROUP BY nombre;
```

```
-- obsérvese que no se cuentan los colores con valor NULL
-- comprobar las diferentes opciones de contar
SELECT nombre, count(*), count(color), count(DISTINCT color)
FROM prendas
GROUP BY nombre;

-- para ver los pares de valores (nombre, color) repetidos
SELECT nombre, color, count(*) veces
 FROM prendas
 GROUP BY nombre, color
 HAVING count(*) > 1;

-- valores de nombre para los que no se cumple la DF
-- aplicando directamente la definición de DF
SELECT DISTINCT nombre FROM prendas P1
 WHERE EXISTS (
   SELECT * FROM prendas P2
     WHERE P2.nombre = P1.nombre
      AND (P2.color <> P1.color
       OR P2.color IS NULL AND P1.color IS NOT NULL
       OR P2.color IS NOT NULL AND P1.color IS NULL)
 );

-- también se podrían obtener con un auto-join (puede ser más eficiente)
-- valores de nombre para los que no se cumple la DF
SELECT DISTINCT P1.nombre
 FROM prendas P1, prendas P2
 WHERE P1.nombre = P2.nombre
   AND (P1.color <> P2.color
    OR P2.color IS NULL     AND P1.color IS NOT NULL
    OR P2.color IS NOT NULL AND P1.color IS NULL);

 -- contar los nombres de prenda para los que no se cumple la DF
SELECT count(DISTINCT P1.nombre) "nombres fallan"
 FROM prendas P1, prendas P2
 WHERE P1.nombre = P2.nombre
   AND (P1.color <> P2.color
    OR P2.color IS NULL     AND P1.color IS NOT NULL
    OR P2.color IS NOT NULL AND P1.color IS NULL);

-- para obtener las prendas que no verifican la DF (de las que hay más de un color)
-- contando directamente, hay que operar sobre los pares (nombre, color) distintos
-- para ello se creará una vista con todos los atributos que intervienen en la DF
CREATE VIEW nombre_color (nombre, color, veces) AS
 SELECT nombre, color, count(*)
 FROM prendas
 GROUP BY nombre, color;

-- valores de nombre de prenda para los que no se cumple la DF
SELECT nombre, count(*)-1 C_sobran, sum(veces)-max(veces) min_Tpl_sobran
FROM nombre_color
GROUP BY nombre
HAVING count(*) > 1;

-- para obtener paso a paso el porcentaje de tuplas que no cumplen la DF
-- se crea una vista con el número mínimo de tuplas que "fallan" (las que,
-- como mínimo, habría que quitar para que se verificase la DF)
CREATE VIEW fallos(numFallos) AS
SELECT sum(tuplas)
FROM (SELECT sum(veces)-max(veces) tuplas
       FROM nombre_color
       GROUP BY nombre);

-- se crea otra vista con el número total de tuplas que hay
CREATE VIEW totPrendas(numPrendas) AS SELECT count(*) FROM prendas;

-- finalmente se obtiene el porcentaja de tuplas que "fallan"
SELECT numFallos/numPrendas*100 "% fallos" FROM fallos, totPrendas;

-- eliminar las vistas auxiliares
DROP VIEW fallos;
DROP VIEW totPrendas;
```

```
-- obtener directamente (sin usar vistas) el porcentaja de tuplas que "fallan"
SELECT numFallos/totPrendas*100 "% fallos"
 FROM
  (SELECT sum(tuplas) numFallos
    FROM (
     SELECT sum(veces)-max(veces) tuplas
      FROM (SELECT nombre, color, count(*) veces FROM prendas GROUP BY nombre, color)
      GROUP BY nombre
    )
  ), (SELECT count(*) totPrendas FROM prendas);

-- la consulta anterior se puede guardar parametrizada con el nombre de la tabla
-- y los conjuntos de atributos que intervienen en la DF, en el fichero tst_DF_nw.sql
-- esto permite obtener los resultados para otras DF, de un modo simple y rápido
@tst_DF_nw prendas "nombre,color" talla
@tst_DF_nw prendas nombre "color,talla"
@tst_DF_nw prendas color talla


--=============================================================================
-- estudio de dependencias multivaluadas                                      --
--=============================================================================
-- comprobar si, en la tabla  prendas,  nombre ->-> color es una DMV

CREATE VIEW nombresFallo AS (
  SELECT DISTINCT nombre FROM prendas PF
   WHERE EXISTS (
     SELECT talla FROM prendas TP
      WHERE nombre = PF.nombre
        AND EXISTS(
              SELECT color FROM prendas WHERE nombre = PF.nombre
              MINUS
              SELECT color FROM prendas WHERE nombre = PF.nombre AND talla = TP.talla
            )
   )
);
SELECT * FROM nombresFallo;

INSERT INTO Prendas(nombre, color, talla) VALUES('PANTALON', 'ROJO', 7);
SELECT * FROM nombresFallo;

INSERT INTO Prendas(nombre, color, talla) VALUES('FALDA', 'VERDE', 4);
SELECT * FROM nombresFallo;

-- se puede definir un disparador que garantice la integridad (mantenga la DMV)
CREATE OR REPLACE TRIGGER tst4FN
AFTER INSERT OR DELETE OR UPDATE ON prendas
DECLARE numFallos number(9);
BEGIN
 SELECT count(*) INTO numFallos FROM prendas PF
   WHERE EXISTS (
     SELECT talla FROM prendas TP
      WHERE nombre = PF.nombre
        AND EXISTS(
          SELECT color FROM prendas WHERE nombre = PF.nombre
          MINUS
          SELECT color FROM prendas WHERE nombre = PF.nombre AND talla = TP.talla
        )
   );
 IF numFallos > 0 THEN
  raise_application_error( -20501, 'violación de la DMV');
 END IF;
END;
/
SHOW ERRORS TRIGGER tst4FN
-- ALTER TRIGGER tst4FN DISABLE;
-- ALTER TRIGGER tst4FN ENABLE;

-- si se intenta actualizar la tabla prendas, se abortará la operación si la tabla no
-- verifica la DMV tras la operación (no se detecta que inicialmente no la verifique)
-- ejemplos:
-- la siguiente operación se abortará aunque no modifique la tabla si inicialmente no se
verifica la DMV
DELETE FROM prendas WHERE nombre = 'CAMISETA';
```

```
-- la siguiente operación no dará problemas pues la tabla final verifica la DMV
DELETE FROM prendas WHERE (nombre, color, talla) IN (
 ('PANTALON', 'ROJO', 7), ('FALDA', 'VERDE', 4),
 ('PANTALON', 'AZUL', 7), ('PANTALON', 'AZUL', NULL)
);
-- eliminar el disparador (ya no se quiere usar más)
DROP TRIGGER tst4FN;
-- Obtención de las tuplas que debería haber para que se verificase la DMV (usando UNION)
CREATE OR REPLACE VIEW coloresPrenda AS (
  SELECT DISTINCT nombre, color FROM prendas WHERE color IS NOT NULL
);

CREATE OR REPLACE  VIEW tallasPrenda AS (
  SELECT DISTINCT nombre, talla FROM prendas WHERE talla IS NOT NULL
);
SELECT * FROM coloresPrenda;
SELECT * FROM tallasPrenda;

-- tuplas que debería haber
SELECT CP.nombre, color, talla
 FROM coloresPrenda CP, tallasPrenda TP WHERE CP.nombre = TP.nombre
UNION
SELECT CP.nombre, color, NULL
 FROM coloresPrenda CP WHERE nombre NOT IN (SELECT nombre FROM tallasPrenda)
UNION
SELECT TP.nombre, NULL, talla
 FROM tallasPrenda TP WHERE nombre NOT IN (SELECT nombre FROM coloresPrenda);

-- tuplas que faltan
CREATE VIEW tuplasFaltan AS (
 SELECT CP.nombre, color, talla
  FROM coloresPrenda CP, tallasPrenda TP WHERE CP.nombre = TP.nombre
 UNION
 SELECT CP.nombre, color, NULL
  FROM coloresPrenda CP WHERE nombre NOT IN (SELECT nombre FROM tallasPrenda)
 UNION
 SELECT TP.nombre, NULL, talla
  FROM tallasPrenda TP WHERE nombre NOT IN (SELECT nombre FROM coloresPrenda)
 MINUS
 SELECT nombre, color, talla FROM prendas
);
SELECT * FROM tuplasFaltan;

-- tuplas que sobran
CREATE VIEW tuplasSobran AS (
 SELECT nombre, color, talla FROM prendas
 MINUS (
 SELECT CP.nombre, color, talla
  FROM coloresPrenda CP, tallasPrenda TP WHERE CP.nombre = TP.nombre
 UNION
 SELECT CP.nombre, color, NULL
  FROM coloresPrenda CP WHERE nombre NOT IN (SELECT nombre FROM tallasPrenda)
 UNION
 SELECT TP.nombre, NULL, talla
  FROM tallasPrenda TP WHERE nombre NOT IN (SELECT nombre FROM coloresPrenda)
 )
);
SELECT * FROM tuplasSobran;

INSERT INTO Prendas(nombre, color, talla) VALUES('CALCETIN', 'NEGRO', NULL);
SELECT * FROM tuplasFaltan;
SELECT * FROM tuplasSobran;
INSERT INTO Prendas(nombre, color, talla) VALUES('PANTALON', 'AZUL', 7);
INSERT INTO Prendas(nombre, color, talla) VALUES('CAMISETA', 'AZUL', NULL);
SELECT * FROM tuplasFaltan;
SELECT * FROM tuplasSobran;

DELETE FROM prendas WHERE (nombre, color, talla) IN (
 ('PANTALON', 'ROJO', 7), ('FALDA', 'VERDE', 4),
 ('PANTALON', 'AZUL', 7), ('PANTALON', 'AZUL', NULL)
);
```

```
-- eliminar el calcetín negro
DELETE FROM prendas WHERE color = 'NEGRO';
-- borrar las tuplas con color o talla NULL
DELETE FROM prendas WHERE color IS NULL OR talla IS NULL;

-- Obtención de las tuplas que debería haber para que se verificase la DMV
--
-- creación de tablas (o vistas) con los colores y tallas de las prendas
-- (son las tablas en que habría que descomponer la tabla prendas al normalizar)
CREATE TABLE coloresP AS (
  SELECT DISTINCT nombre, color FROM prendas WHERE color IS NOT NULL
);
CREATE TABLE tallasP AS (
  SELECT DISTINCT nombre, talla FROM prendas WHERE talla IS NOT NULL
);

SELECT * FROM coloresP;
SELECT * FROM tallasP;

-- tuplas que debería haber para que se verificase la DMV (usando OUTER JOINs)
SELECT coloresP.nombre nombre, color, talla
 FROM coloresP, tallasP WHERE coloresP.nombre = tallasP.nombre(+)
UNION
SELECT tallasP.nombre nombre, color, talla
 FROM coloresP, tallasP WHERE tallasP.nombre = coloresP.nombre(+);

-- tuplas que debería haber para que se verificase la DMV (sintaxis SQL2)
SELECT coloresP.nombre, color, talla
FROM tallasP FULL OUTER JOIN coloresP ON tallasP.nombre = coloresP.nombre;

-- para solventar que en el full join no sale el nombre si color es null
SELECT tallasP.nombre, color, talla
FROM tallasP LEFT OUTER JOIN coloresP ON tallasP.nombre = coloresP.nombre
UNION
SELECT coloresP.nombre, color, talla
FROM tallasP RIGHT OUTER JOIN coloresP ON tallasP.nombre = coloresP.nombre;

-- eliminar las tablas auxiliares
DROP TABLE coloresP;
DROP TABLE tallasP;

-- eliminar las vistas auxiliares
DROP VIEW nombresFallo;
DROP VIEW tuplasFaltan;
DROP VIEW tuplasSobran;
DROP VIEW coloresPrenda;
DROP VIEW tallasPrenda;
DROP VIEW nombre_color;

-- eliminar las restricciones definidas
ALTER TABLE prendas DROP CONSTRAINT prendas_UN;

-- eliminar la tabla prendas
DROP TABLE prendas;
```