ejercicios de consultas y normalización

Sea la tabla `prendas` *de una BD para la gestión de una tienda de ropa, definida como:*

```
CREATE TABLE prendas (
  nombre     VARCHAR2(24),
  color      VARCHAR2(24),
  talla      NUMBER(2)
);
```

A partir de los datos de la tabla se pretenden extraer las propiedades de los atributos. Ejemplo:

NOMBRE	COLOR	TALLA
FALDA	ROJO	4
FALDA	AZUL	5
FALDA	AZUL	4
FALDA	ROJO	5
PANTALON	VERDE	6
PANTALON	ROJO	6
CALCETIN		8
FALDA	ROJO	4
FALDA	AZUL	5

ejercicios de consultas y normalización

1 *Obtener las claves candidatas de la tabla* `prendas`

Para ello, comprobaremos si (`nombre`, `talla`, `color`*) es una clave de la tabla* `prendas`

```
SELECT prendas.*, count(*) veces
 FROM prendas
 GROUP BY nombre, color, talla
 HAVING count(*) > 1;
```

NOMBRE	COLOR	TALLA	VECES
FALDA	AZUL	5	2
FALDA	ROJO	4	2

¡ no debería salir ninguna !

problema: ***hay que***
- ***eliminar las tuplas repetidas***
- ***impedir que se repita esta situación***

ejercicios de consultas y normalización

2 *eliminar las tuplas repetidas de la tabla* `prendas`

```
-- eliminar tuplas repetidas (no se ha especificado clave primaria de prendas)
DELETE FROM prendas
 WHERE rowid NOT IN (
  SELECT min(rowid) FROM prendas GROUP BY nombre, color, talla
 );
```

se podría haber creado una vista para hacer la consulta paso a paso

Para impedir que puedan añadirse tuplas repetidas no puede especificarse clave primaria (hay nulos)

```
-- especificar la clave candidata necesaria
ALTER TABLE prendas
        ADD CONSTRAINT prendas_UN UNIQUE(nombre, color, talla);
```

a partir de aquí se supondrá que no hay tuplas repetidas

ejercicios de consultas y normalización

3 *También se puede crear un "script" para automatizar la comprobación de claves:*

tst_clave.sql

```
-- listar las tuplas con valores repetidos
SELECT &2, count(*) veces FROM &1 GROUP BY &2 HAVING count(*) > 1;

-- mostrar resultado final: "es clave" o "no es clave"
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
 );
```

para invocarlo: `@tst_clave prendas "nombre, talla, color"`

ejercicios de consultas y normalización

```
SQL> @tst_clave prendas "nombre, talla, color"

listado de tuplas con valores repetidos de (nombre, talla, color)

antiguo   1: select &2, count(*) veces from &1 group by &2 having count(*) > 1
nuevo   1: select nombre, talla, color, count(*) veces from prendas group by nombre,
talla, color having count(*) > 1

ninguna fila seleccionada

antiguo   3:   select * from &1 group by &2 having count(*) > 1
nuevo   3:   select * from prendas group by nombre, talla, color having count(*) > 1
antiguo   8:   select * from &1 group by &2 having count(*) > 1
nuevo   8:   select * from prendas group by nombre, talla, color having count(*) > 1

es clave
---------
   SI
```

sólo queda comprobar que (nombre, color), (nombre, talla) y (color, talla) no son claves

¿qué pasa con los atributos nulos?

ejercicios de consultas y normalización

4 *Dados dos conjuntos de atributos X, Y de una tabla, comprobar si* $X \rightarrow Y$

definición: $\boldsymbol{X \rightarrow Y}$ sii $\forall\, t_1, t_2 \in r(R): \prod_X(t_1) = \prod_X(t_2) \Rightarrow \prod_Y(t_1) = \prod_Y(t_2)$

como ejemplo, se comprobará si `nombre → color` *en la tabla* `prendas`

prueba previa: contar los colores que hay por cada nombre de prenda

```
SELECT nombre, count(color) veces
 FROM prendas
 GROUP BY nombre;
```

```
NOMBRE                         VECES
------------------------- ----------
CALCETIN                            0
FALDA                               4
PANTALON                            2
```

problemas:
- ***no se cuentan el color NULL***
- ***hay colores repetidos (<nombre, color> no es clave)***

ejercicios de consultas y normalización

se prueban otros modos de contar (tuplas, colores y colores distintos)

```
SELECT nombre, count(*), count(color), count(DISTINCT color)
FROM prendas
GROUP BY nombre;
```

```
NOMBRE                    COUNT(*) COUNT(COLOR) COUNT(DISTINCTCOLOR)
------------------------- ---------- ------------ --------------------
CALCETIN                           1            0                    0
FALDA                              4            4                    2
PANTALON                           2            2                    2
```

se pueden contar las tuplas <nombre, color> que están repetidas (no es una clave)

```
SELECT nombre, color, count(*) veces
 FROM prendas
 GROUP BY nombre, color
 HAVING count(*) > 1;
```

```
NOMBRE                    COLOR                     VECES
------------------------- ------------------------- ----------
FALDA                     AZUL                               2
FALDA                     ROJO                               2
```

esto podrá servir

ejercicios de consultas y normalización

4 *comprobar si* `nombre → color` *aplicando la definición de DF*

```
-- valores de nombre para los que no se cumple la DF (usando un auto-Join)
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)
 );
```

```
-- número de valores de nombre para los que no se cumple la DF (usando un auto-Join)
SELECT count(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)
 );
```

ejercicios de consultas y normalización

4 *se pueden mejorar las consultas (aunque el plan generado es el mismo)*

```
-- valores de nombre para los que no se cumple la DF (usando un auto-Join)
SELECT DISTINCT P1.nombre FROM prendas P1, prendas P2
 WHERE P1.nombre = P2.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);
```

```
-- número de valores de nombre para los que no se cumple la DF (usando un auto-Join)
SELECT count(DISTINCT P1.nombre) FROM prendas P1 , prendas P2
 WHERE P1.nombre = P2.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);
```


ejercicios de consultas y normalización

si sólo se tienen en cuenta los atributos que intervienen en la DF

```
SELECT nombre, color, count(*) veces
 FROM prendas
 GROUP BY nombre, color;
```

NOMBRE	COLOR	VECES
CALCETIN		1
FALDA	AZUL	2
FALDA	ROJO	2
PANTALON	ROJO	1
PANTALON	VERDE	1

para simplificar, se crea una vista sólo con 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;
```


ejercicios de consultas y normalización

4 *comprobar si, en la tabla* `prendas, nombre → color`

```
-- valores de nombre 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;
```

```
CREATE VIEW fallos(numFallos) AS
SELECT sum(tuplas)
FROM (SELECT sum(veces)-max(veces) tuplas
       FROM nombre_color
       GROUP BY nombre);
```

```
CREATE VIEW totPrendas(numPrendas) AS SELECT count(*) FROM prendas;
```

```
SELECT numFallos/numPrendas*100 "% fallos" FROM fallos, totPrendas;
```

% fallos
42,8571429

ejercicios de consultas y normalización

4 *sin utilizar vistas auxiliares*

```
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);
```

% fallos
42,8571429

ejercicios de consultas y normalización

5 *También se puede crear un "script" para automatizar la comprobación de DF*

tst_DF_nw.sql

```
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);
```

```
@tst_DF_nw prendas "nombre,color" talla
@tst_DF_nw prendas nombre "color,talla"
@tst_DF_nw prendas color talla
```


ejercicios de consultas y normalización

6 *comprobar si, en la tabla* `prendas`, `nombre` $\rightarrow\!\!\rightarrow$ `color` *es una DMV*

definición de dependencia multivaluada:

$\boldsymbol{X \rightarrow\!\!\rightarrow Y}$ sii si $\exists\ t_1, t_2 \in r(R)$ tales que $\prod_X(t_1) = \prod_X(t_2)$, entonces $\exists\ t_3, t_4 \in r(R)$ cumpliendo:

- $\prod_X(t_3) = \prod_X(t_4) = \prod_X(t_1) = \prod_X(t_2)$
- $\prod_Y(t_3) = \prod_Y(t_1)$, $\prod_Y(t_4) = \prod_Y(t_2)$
- $\prod_Z(t_3) = \prod_Z(t_2)$, $\prod_Z(t_4) = \prod_Z(t_1)$
  donde $Z = R - (X \cup Y)$

otra definición de dependencia multivaluada más útil para comprobar si existe una DMV es:

$\boldsymbol{X \rightarrow\!\!\rightarrow Y}$ sii $\prod_Y(\sigma_{X=x_i}(R)) = \prod_Y(\sigma_{X=x_i \wedge Z=z_i}(R))\ \ \forall x_i \in \boldsymbol{X}$ y $\forall z_i \in \boldsymbol{Z}$, siendo $Z = R - (X \cup Y)$

en el caso de `nombre` $\rightarrow\!\!\rightarrow$ `color` *significaría que los colores de una prenda cualquiera son exactamente los mismos que los que hay de esa prenda en cualquiera de sus tallas.*

ejercicios de consultas y normalización

6 *comprobar si, en la tabla* `prendas`, `nombre` $\rightarrow\!\!\rightarrow$ `color` *es una DMV*

se puede crear una vista con los nombres de las prendas que no verifican la 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
            )
   )
);
```


ejercicios de consultas y normalización

7 *se podría definir un disparador que garantice la integridad*

```
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 4FN');
 END IF;
END;
/
SHOW ERRORS TRIGGER tst4FN
```

EJERCICIO: *diseñar un trigger que añada y elimine las tuplas necesarias para garantizar la consistencia*

ejercicios de consultas y normalización

8 *Obtención de las tuplas que debería haber para que se verificase la DMV*

```
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
);
```

```
-- 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);
```

```
INSERT INTO Prendas(nombre, color, talla) VALUES('CALCETIN', 'NEGRO', NULL);
```


ejercicios de consultas y normalización

8 *Obtención de las tuplas que faltan para que se verifique la DMV*

```
-- 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 nombre, color, talla FROM tuplasFaltan;

NOMBRE                    COLOR                          TALLA
------------------------- ------------------------------ ----------
CALCETIN                  NEGRO                               8
```


ejercicios de consultas y normalización

8 *Obtención de las tuplas que sobran para que se verifique la DMV*

```
-- 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 nombre, color, talla FROM tuplasSobran;

NOMBRE                    COLOR                          TALLA
------------------------- ------------------------------ ----------
CALCETIN                  NEGRO
CALCETIN                                                      8
```


ejercicios de consultas y normalización

8 *Obtención de las tuplas que no verifican la DMV (utilizando JOINs externos)*

```
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 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(+);
```

mejor con vistas

ejercicios de consultas y normalización

8 *Obtención de las tuplas que no verifican la DMV (utilizando JOINs externos y SQL2)*

```
-- utilizando 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;
```


ejercicios de consultas y normalización

9 *Para mantener la consistencia, habrá que añadir las tuplas que "faltan"*

```
INSERT INTO Prendas(nombre, color, talla)
 SELECT nombre, color, talla FROM tuplasFaltan;
```

```
SQL> SELECT * FROM prendas;

NOMBRE                    COLOR                        TALLA
------------------------- ------------------------- ----------
FALDA                     ROJO                              4
FALDA                     AZUL                              5
FALDA                     AZUL                              4
FALDA                     ROJO                              5
PANTALON                  VERDE                             6
PANTALON                  ROJO                              6
CALCETIN                                                    8
CALCETIN                  NEGRO
CALCETIN                  NEGRO                             8

SQL> SELECT * FROM tuplasfaltan;

ninguna fila seleccionada
```


ejercicios de consultas y normalización

9 *También habrá que eliminar las tuplas que "sobran" (¡después de haber añadido las que faltan!)*

```
DELETE FROM prendas WHERE (nombre, color, talla) IN (
 SELECT nombre, color, talla FROM tuplasSobran);
```

```
0 filas suprimidas.

SQL> SELECT * FROM prendas;

NOMBRE                    COLOR                        TALLA
------------------------- ------------------------- ----------
FALDA                     ROJO                              4
FALDA                     AZUL                              5
FALDA                     AZUL                              4
FALDA                     ROJO                              5
PANTALON                  VERDE                             6
PANTALON                  ROJO                              6
CALCETIN                                                    8
CALCETIN                  NEGRO
CALCETIN                  NEGRO                             8

SQL> SELECT * FROM tuplasSobran;
NOMBRE                    COLOR                        TALLA
------------------------- ------------------------- ----------
CALCETIN                  NEGRO
CALCETIN                                                    8
```

¡ NO SE ELIMINAN !

pb. con valores null

ejercicios de consultas y normalización

9 *Para eliminar las tuplas que "sobran" habrá que tener en cuenta los valores NULL*

```
DELETE FROM prendas
  WHERE (nombre, color) IN (SELECT nombre, color FROM tuplasSobran WHERE talla IS NULL)
    AND talla IS NULL
     OR (nombre, talla) IN (SELECT nombre, talla FROM tuplasSobran WHERE color IS NULL)
    AND color IS NULL;
```

```
2 filas suprimidas.

SQL> SELECT * FROM prendas;

NOMBRE                    COLOR                        TALLA
------------------------- ------------------------- ----------
FALDA                     ROJO                              4
FALDA                     AZUL                              5
FALDA                     AZUL                              4
FALDA                     ROJO                              5
PANTALON                  VERDE                             6
PANTALON                  ROJO                              6
CALCETIN                  NEGRO                             8

SQL> SELECT * FROM tuplasSobran;

ninguna fila seleccionada
```