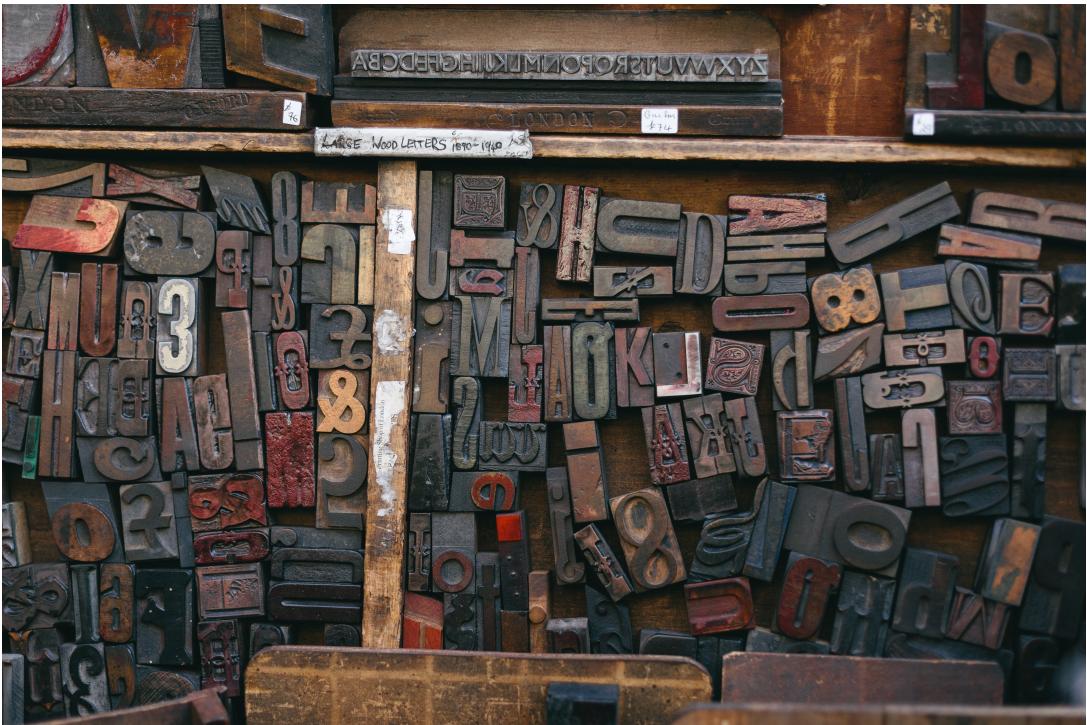


Cuaderno de prácticas FBD.



Francisco Navarro Morales - GRG121

Segundo curso del Grado de Ingeniería Informática
Universidad de Granada
curso 2016-2017

Índice

1. Punto 2. Creación de la BD	3
1.1. Ejercicio 2.6: Modifica el esquema de la tabla plantilla añadiendo un nuevo atributo llamado fechabaja de tipo date.	3
1.2. Ejercicio 2.8: Ejecuta la sentencia SELECT para mostrar el contenido de las tablas PRUEBA2 y PLANTILLA. Intenta mostrar sólo algunos campos de las mismas.	3
1.3. Ejercicio 2.9 Ejecuta la sentencia UPDATE sobre la tabla plantilla y cambia el nombre del trabajador con dni '12345678' a 'Luis'.	3
1.4. Ejercicio 2.10 Borra todas las tuplas de la tabla plantilla. SQL> DELETE FROM plantilla; En este caso da un mensaje de error (¿por qué?). Aunque sí podríamos borrar las tuplas de la tabla serjefe. SQL> DELETE FROM serjefe;	3
1.5. Ejercicio 2.11 A continuación vamos a tratar de insertar algunas tuplas nuevas en ventas. Comprueba que se introducen correctamente y, en caso contrario, razona por qué da error.	3
1.6. Ejercicio 2.12 Actualizar la fecha del proveedor S5 al año 2005'	3
1.7. Ejercicio 2.13 Para mostrar la columna FECHA con un formato específico e imprimirla, utilizar la siguiente sentencia:	4
1.8. Ejercicio 2.14 Dado el esquema siguiente de la base de datos de una liga de baloncesto:	4
1.9. Ejercicio 2.15 Preparar un archivo para la introducción de datos en las tablas Equipos, Jugadores, Encuentros y Faltas, conforme a los siguientes criterios, para que nos permitan realizar consultas con resultados significativos. Que se inserten 4 equipos con 5 jugadores en cada uno. Que se inserten 10 encuentros (no se ha terminado la liga). Que se inserten los resultados esos encuentros dejando un único equipo invicto.	5
2. Punto 3. PRACTICA 2, CONSULTAS.	5
2.1. Ejercicio 3.1 Comprueba el resultado de la proyección. ¿Es éste conforme a lo que se obtiene en el AR?	5
2.2. Ejercicio 3.2 Muestra los suministros realizados (tan solo los códigos de los componentes de una venta). ¿Es necesario utilizar DISTINCT?	5
2.3. Ejercicio 3.3 Muestra las piezas de Madrid que son grises o rojas.	5
2.4. Ejercicio 3.4 Encontrar todos los suministros cuya cantidad está entre 200 y 300, ambos inclusive.	6
2.5. Ejercicio 3.5 Mostrar las piezas que contengan la palabra tornillo con la t en mayúscula o en minúscula.	6
2.6. Ejercicio 3.6 Comprueba que no devuelve ninguna. Pero SI que hay!!!	6
2.7. Ejercicio 3.7 Resolver la consulta del ejemplo 3.8 utilizando el operador \cap	6
2.8. Ejercicio 3.8 Encontrar los códigos de aquellos proyectos a los que sólo abastece 'S1'.	6
2.9. Ejercicio 3.9 Mostrar todas las ciudades de la base de datos. Utilizar UNION	6
2.10. Ejercicio 3.10 Mostrar todas las ciudades de la base de datos. Utilizar UNION ALL	6
2.11. Ejercicio 3.11 Comprueba cuántas tuplas resultan del producto cartesiano aplicado a ventas y proveedor	6
2.12. Ejercicio 3.12 Mostrar las ternas que son de la misma ciudad pero que hayan realizado alguna venta.	7
2.13. Ejercicio 3.13 Encontrar parejas de proveedores que no viven en la misma ciudad.	7
2.14. Ejercicio 3.14 Encuentra las piezas con máximo peso.	7

2.15. Ejercicio 3.15 Mostrar las piezas vendidas por los proveedores de Madrid.	7
2.16. Ejercicio 3.16 Encuentra la ciudad y los códigos de las piezas suministradas a cualquier proyecto por un proveedor que está en la misma ciudad donde está el proyecto.	7
2.17. Ejercicio 3.17 Comprobar la salida de la consulta anterior sin la cláusula ORDER BY.	7
2.18. Ejercicio 3.18 Listar las ventas ordenadas por cantidad, si algunas ventas coinciden en la cantidad se ordenan en función de la fecha de manera descendente.	7
2.19. Ejercicio 3.19 Mostrar las piezas vendidas por los proveedores de Madrid. (Fragmentando la consulta con ayuda del operador IN.) Compara la solución con la del ejercicio 3.15.	8
2.20. Ejercicio 3.20 Encuentra los proyectos que están en una ciudad donde se fabrica alguna pieza.	8
2.21. Ejercicio 3.21 Encuentra los códigos de aquellos proyectos que no utilizan ninguna pieza roja que esté suministrada por un proveedor de Londres.	8
2.22. Ejercicio 3.22 Muestra el código de las piezas cuyo peso es mayor que el peso de cualquier 'tornillo'.	8
2.23. Ejercicio 3.23 Encuentra las piezas con peso máximo. Compara esta solución con la obtenida en el ejercicio 3.14	8

1. Punto 2. Creación de la BD

- 1.1. Ejercicio 2.6: Modifica el esquema de la tabla plantilla añadiendo un nuevo atributo llamado fechabaja de tipo date.**

ALTER TABLE ADD DATE FECHABAJA

- 1.2. Ejercicio 2.8: Ejecuta la sentencia SELECT para mostrar el contenido de las tablas PRUEBA2 y PLANTILLA. Intenta mostrar sólo algunos campos de las mismas.**

select cad, dni from prueba2, plantilla

- 1.3. Ejercicio 2.9 Ejecuta la sentencia UPDATE sobre la tabla plantilla y cambia el nombre del trabajador con dni '12345678' a 'Luis'.**

update table plantilla set nombre = 'Luis' where dni = '12345678';

- 1.4. Ejercicio 2.10 Borra todas las tuplas de la tabla plantilla. SQL> DELETE FROM plantilla; En este caso da un mensaje de error (¿por qué?). Aunque sí podríamos borrar las tuplas de la tabla serjefe. SQL> DELETE FROM serjefe;**

Se debe a que la tabla plantilla es referenciada por un atributo de la tabla serjefe y no puede ser borrada sin modificar antes esta circunstancia o borrar la tabla serjefe. Si borramos la tabla serjefe y, a continuación, borramos la tabla plantilla, no dará problemas.

- 1.5. Ejercicio 2.11 A continuación vamos a tratar de insertar algunas tuplas nuevas en ventas. Comprueba que se introducen correctamente y, en caso contrario, razona por qué da error.**

- *insert into ventas values ('S3','P1','J1',150,'24/12/05');* Da error porque el valor de la fecha no se está entendiendo como una fecha, hay que utilizar la directiva 'TO_DATE' de la forma:
insert into ventas values ('S3','P1','J1',150,TO_DATE('24/12/05','dd/mm/yyyy'));
- *insert into ventas (codpro,codpj) values ('S4','J2');* el error lo provoca no insertar ningún valor para el campo codpie, que tiene una restricción de *not null* que impide que no se le asigne ningún valor.
- *insert into ventas values('S5','P3','J6',400,TO_DATE('25/12/00'));* falla porque la sentencia TO_DATE está incompleta, ya que le falta el formato. Para que fuera correcta debería tener:
TO_DATE('25/12/00','dd/mm/yyyy')

- 1.6. Ejercicio 2.12 Actualizar la fecha del proveedor S5 al año 2005'**

No hay que hacer nada, funciona perfectamente. *UPDATE ventas SET fecha = TO_DATE(2005,'YYYY') WHERE S5';*

1.7. Ejercicio 2.13 Para mostrar la columna FECHA con un formato específico e imprimirla, utilizar la siguiente sentencia:

select codpro,codpie, to_char(fecha,'"Dia"day,dd/mm/yy') fromventas; funciona correctamente.

1.8. Ejercicio 2.14 Dado el esquema siguiente de la base de datos de una liga de baloncesto:

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'File', 'Edit', 'Tools', 'Database', 'Help', and a connection status 'X4202078'. The toolbar has icons for Undo, Redo, Save, Print, and others. The left sidebar has tabs for 'PIEZA' and 'VENTAS', with 'VENTAS' selected. The main area is titled '*<FBD> Script-3' and contains the following SQL code:

```
CREATE TABLE equipos (
    codE char(10) NOT NULL PRIMARY KEY,
    nombreE varchar(20) NOT NULL UNIQUE,
    localidad varchar(20) NOT NULL,
    entrenador varchar(20) NOT NULL,
    fecha_crea DATE)

CREATE TABLE jugadores(
    codJ char(10) NOT NULL PRIMARY KEY,
    codE char(10) NOT NULL REFERENCES equipos(codE),
    nombreJ varchar(20) NOT NULL)

CREATE TABLE encuentros(
    ELocal char(10) NOT NULL,
    EVisitante CHAR(10) NOT NULL,
    fecha DATE,
    PLocal integer DEFAULT 0 NOT null,
    PVisitante integer DEFAULT 0 NOT NULL ,
    PRIMARY KEY(ELocal,EVisitante),
    FOREIGN KEY (ELocal) REFERENCES equipos(codE),
    FOREIGN KEY (EVisitante) REFERENCES equipos(codE),
    check(PLocal >= 0), check(PVisitante >= 0) )

CREATE TABLE faltas(
    codJ char(10) NOT NULL REFERENCES jugadores(codJ),
    ELocal char(10) NOT NULL REFERENCES equipos(codE),
    EVisitante char(19) NOT NULL REFERENCES equipos(codE),
    num integer DEFAULT 0 NOT NULL CHECK(num BETWEEN 0 AND 5))
```

1.9. Ejercicio 2.15 Preparar un archivo para la introducción de datos en las tablas Equipos, Jugadores, Encuentros y Faltas, conforme a los siguientes criterios, para que nos permitan realizar consultas con resultados significativos. Que se inserten 4 equipos con 5 jugadores en cada uno. Que se inserten 10 encuentros (no se ha terminado la liga). Que se inserten los resultados esos encuentros dejando un único equipo invicto.

```

INSERT INTO equipos VALUES ('0000000000','FULANOS DEL DESIERTO','RUSIA','PUTIN',TO_DATE('11/10/2013','dd/mm/yyyy'));
INSERT INTO equipos VALUES ('0000000001','FULANOS DE LA TUNDRA','MARRUCOS','SOFIANE',TO_DATE('14/02/2012','dd/mm/yyyy'));
INSERT INTO equipos VALUES ('0000000002','MENGANOS 1','MURCIA','MARCELO',TO_DATE('13/03/2020','dd/mm/yyyy'));
INSERT INTO equipos VALUES ('0000000003','MENGANOS 2','MADRID','CARMENAS',TO_DATE('12/12/2009','dd/mm/yyyy'));

INSERT INTO JUGADORES VALUES ('0000000010','0000000000','fulanito0');
INSERT INTO JUGADORES VALUES ('0000000011','0000000000','fulanito2');
INSERT INTO JUGADORES VALUES ('0000000012','0000000000','fulanito3');
INSERT INTO JUGADORES VALUES ('0000000013','0000000000','fulanito4');
INSERT INTO JUGADORES VALUES ('0000000014','0000000000','fulanito5');

INSERT INTO JUGADORES VALUES ('0000000020','0000000001','menganito0');
INSERT INTO JUGADORES VALUES ('0000000021','0000000001','menganito2');
INSERT INTO JUGADORES VALUES ('0000000022','0000000001','menganito3');
INSERT INTO JUGADORES VALUES ('0000000023','0000000001','manganito');
INSERT INTO JUGADORES VALUES ('0000000024','0000000001','manganito');

INSERT INTO JUGADORES VALUES ('0000000030','0000000002','fulanito10');
INSERT INTO JUGADORES VALUES ('0000000031','0000000002','fulanito12');
INSERT INTO JUGADORES VALUES ('0000000032','0000000002','fulanito13');
INSERT INTO JUGADORES VALUES ('0000000033','0000000002','fulanito14');
INSERT INTO JUGADORES VALUES ('0000000034','0000000002','fulanito15');

INSERT INTO JUGADORES VALUES ('0000000040','0000000003','fulanito20');
INSERT INTO JUGADORES VALUES ('0000000041','0000000003','fulanito22');
INSERT INTO JUGADORES VALUES ('0000000042','0000000003','fulanito23');
INSERT INTO JUGADORES VALUES ('0000000043','0000000003','fulanito24');
INSERT INTO JUGADORES VALUES ('0000000044','0000000003','fulanito25');

INSERT INTO encuentros values('0000000000','0000000001',TO_DATE('01/02/2017','dd/mm/yyyy'),99,10);
INSERT INTO encuentros values('0000000000','0000000002',TO_DATE('02/02/2017','dd/mm/yyyy'),99,10);
INSERT INTO encuentros values('0000000000','0000000003',TO_DATE('03/02/2017','dd/mm/yyyy'),99,22);
INSERT INTO encuentros values('0000000001','0000000002',TO_DATE('04/02/2017','dd/mm/yyyy'),17,21);
INSERT INTO encuentros values('0000000001','0000000000',TO_DATE('05/02/2017','dd/mm/yyyy'),16,99);
INSERT INTO encuentros values('0000000003','0000000002',TO_DATE('06/02/2017','dd/mm/yyyy'),15,40);
INSERT INTO encuentros values('0000000001','0000000003',TO_DATE('07/02/2017','dd/mm/yyyy'),14,50);
INSERT INTO encuentros values('0000000002','0000000001',TO_DATE('08/02/2017','dd/mm/yyyy'),13,60);
INSERT INTO encuentros values('0000000002','0000000000',TO_DATE('09/02/2017','dd/mm/yyyy'),12,99);
INSERT INTO encuentros values('0000000002','0000000003',TO_DATE('10/02/2017','dd/mm/yyyy'),15,80);

```

Statistics

2. Punto 3. PRACTICA 2, CONSULTAS.

2.1. Ejercicio 3.1 Comprueba el resultado de la proyección. ¿Es éste conforme a lo que se obtiene en el AR?

Se obtiene Londres,Londres,Paris,Roma, que es distinto de lo obtenido en AR porque AR no da resultados repetidos. Para evitar que se repitan resultados en SQL habría que utilizar select distinct

2.2. Ejercicio 3.2 Muestra los suministros realizados (tan solo los códigos de los componentes de una venta). ¿Es necesario utilizar DISTINCT?

No hace falta poner unique porque uno de los atributos mostrados es clave primaria y por tanto no va a haber ninguna tupla repetida.

2.3. Ejercicio 3.3 Muestra las piezas de Madrid que son grises o rojas.

```
SELECT * FROM pieza WHERE color IN ('Rojo','Gris') AND ciudad = 'Madrid'
```

P1 Tuerca Gris 2.5 Madrid

2.4. Ejercicio 3.4 Encontrar todos los suministros cuya cantidad está entre 200 y 300, ambos inclusive.

SELECT * FROM ventas WHERE CANTIDAD BETWEEN 200 AND 300

2.5. Ejercicio 3.5 Mostrar las piezas que contengan la palabra tornillo con la t en mayúscula o en minúscula.

SELECT codpie FROM PIEZA WHERE nompie IN ('Tornillo', 'tornillo')

2.6. Ejercicio 3.6 Comprueba que no devuelve ninguna. Pero SI que hay!!!

efectivamente, no devuelve ninguna. Se debe a que existe la tabla ventas pero tiene el nombre en mayúsculas. Si utilizamos:

*Select tablename from ALL_TABLES where TABLE_NAME like 'VENTAS';
sí que funciona.*

2.7. Ejercicio 3.7 Resolver la consulta del ejemplo 3.8 utilizando el operador \cap .

la consulta: $\pi_{ciudad}(\sigma_{status>2}(proveedor)) - \pi_{ciudad}(\sigma_{cod_pie='P1'}(pieza))$

se puede escribir como:

$\pi_{ciudad}(\sigma_{status>2}(proveedor) \cap (\pi_{ciudad}(proveedor) - \pi_{ciudad}(\sigma_{cod_pie='P1'}(pieza)))$

2.8. Ejercicio 3.8 Encontrar los códigos de aquellos proyectos a los que sólo abastece 'S1'.

SELECT codpj FROM proyecto MINUS SELECT codpj FROM ventas WHERE codpro != 'S1'

2.9. Ejercicio 3.9 Mostrar todas las ciudades de la base de datos. Utilizar UNION

SELECT ciudad FROM PIEZA UNION SELECT ciudad FROM PROVEEDOR UNION SELECT ciudad FROM

2.10. Ejercicio 3.10 Mostrar todas las ciudades de la base de datos. Utilizar UNION ALL

SELECT ciudad FROM PIEZA UNION ALL SELECT ciudad FROM PROVEEDOR UNION ALL SELECT ciudad

La única diferencia es que UNION no devuelve repetidos y UNION ALL sí.

2.11. Ejercicio 3.11 Comprueba cuántas tuplas resultan del producto cartesiano aplicado a ventas y proveedor

SELECT count() FROM ventas, proveedor*

2.12. Ejercicio 3.12 Mostrar las ternas que son de la misma ciudad pero que hayan realizado alguna venta.

```
SELECT * FROM pieza,ventas,PROVEEDOR,PROYECTO WHERE pieza.ciudad = proveedor.ciudad AND proveedor.ciudad = proyecto.ciudad AND ventas.CODPRO = proveedor.codpro AND ventas.codpie = pieza.codpie AND ventas.CODPJ = proyecto.codpj
```

2.13. Ejercicio 3.13 Encontrar parejas de proveedores que no viven en la misma ciudad.

```
SELECT proveedor.CODPRO, pro.codpro FROM proveedor, proveedor pro WHERE proveedor.ciudad > pro.ciudad
```

2.14. Ejercicio 3.14 Encuentra las piezas con máximo peso.

```
SELECT pieza.codpie FROM pieza MINUS SELECT pieza.codpie FROM pieza, pieza pie WHERE pieza.peso < pie.PESO
```

2.15. Ejercicio 3.15 Mostrar las piezas vendidas por los proveedores de Madrid.

```
SELECT DISTINCT codpie FROM ventas NATURAL JOIN proveedor WHERE proveedor.ciudad = 'Madrid'
```

2.16. Ejercicio 3.16 Encuentra la ciudad y los códigos de las piezas suministradas a cualquier proyecto por un proveedor que está en la misma ciudad donde está el proyecto.

```
SELECT DISTINCT proveedor.ciudad, codpie FROM proveedor, proyecto, ventas WHERE proveedor.ciudad = proyecto.ciudad AND proveedor.codpro = ventas.codpro AND proyecto.codpj = ventas.codpj
```

2.17. Ejercicio 3.17 Comprobar la salida de la consulta anterior sin la cláusula ORDER BY.

con order by: Jose Fernandez Luisa Gomez Manuel Vidal Maria Reyes Pedro Sanchez.

sin order by: Jose Fernandez Manuel Vidal Luisa Gomez Pedro Sanchez Maria Reyes.

2.18. Ejercicio 3.18 Listar las ventas ordenadas por cantidad, si algunas ventas coinciden en la cantidad se ordenan en función de la fecha de manera descendente.

```
SELECT * FROM ventas ORDER BY cantidad ASC, fecha DESC
```

2.19. Ejercicio 3.19 Mostrar las piezas vendidas por los proveedores de Madrid.
(Fragmentando la consulta con ayuda del operador IN.) Compara la solución con la del ejercicio 3.15.

Select codpie from ventas where codpro IN (select codpro from proveedor where ciudad = 'Londres');

Esta solución no requiere el natural join y se lee mejor que la del ejercicio 3.15:

SELECT DISTINCT codpie FROM ventas NATURAL JOIN proveedor WHERE proveedor.ciudad = 'Madrid'

2.20. Ejercicio 3.20 Encuentra los proyectos que están en una ciudad donde se fabrica alguna pieza.

SELECT codpj FROM proyecto WHERE ciudad IN (SELECT ciudad FROM PIEZA)

2.21. Ejercicio 3.21 Encuentra los códigos de aquellos proyectos que no utilizan ninguna pieza roja que esté suministrada por un proveedor de Londres.

SELECT codpj FROM ventas MINUS SELECT codpj FROM ventas WHERE codpie IN (SELECT codpie FROM 'Rojo' AND codpro IN (SELECT codpro FROM PROVEEDOR WHERE ciudad = 'Londres'))

2.22. Ejercicio 3.22 Muestra el código de las piezas cuyo peso es mayor que el peso de cualquier 'tornillo'.

SELECT codpie FROM pieza WHERE NOT EXISTS (SELECT codpie FROM pieza pie WHERE pie.nompie = 'Tornillo' AND pie.peso > pieza.peso)

2.23. Ejercicio 3.23 Encuentra las piezas con peso máximo. Compara esta solución con la obtenida en el ejercicio 3.14

SELECT codpie FROM pieza WHERE NOT EXISTS (SELECT codpie FROM pieza pie WHERE pie.peso > pieza.peso)

en el ejercicio 14:

SELECT pieza.codpie FROM pieza MINUS SELECT pieza.codpie FROM pieza, pieza pie WHERE pieza.peso < pie.PESO

La principal diferencia es que al utilizar el minus restamos las piezas que pesan menos que alguna otra pieza (y quedan las que pesan mas que todas las demás) y al usar not exists seleccionamos aquellas piezas para las que no existe ninguna que pese más, es decir, se comprueba si existen piezas que pesen más. Luego la consulta es totalmente distinta.

2.24. Ejercicio 3.24 Encontrar los códigos de las piezas suministradas a todos los proyectos localizados en Londres.

```
SELECT codpie FROM pieza WHERE NOT EXISTS (SELECT codpj FROM proyecto WHERE ciudad = 'Londres' AND NOT EXISTS (SELECT * FROM ventas WHERE pieza.codpie = ventas.codpie AND proyecto.codpj = ventas.codpj))
```

2.25. Ejercicio 3.25 Encontrar aquellos proveedores que envían piezas procedentes de todas las ciudades donde hay un proyecto.

```
SELECT codpro FROM proveedor WHERE NOT exists( SELECT ciudad FROM pieza WHERE ciudad IN (SELECT ciudad FROM ventas WHERE proveedor.codpro = ventas.codpro AND pieza.codpie = ventas.codpie) )
```