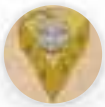


QuesoViejo_
www.wuolah.com/student/QuesoViejo_

Examen-3.pdf

Ejercicios Resueltos SQL

2° Ficheros y Bases de Datos

Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad Carlos III de Madrid

*NOTA: Todas las consultas han sido probadas satisfactoriamente.

Examen de prácticas de Bases de Datos

Parte 2. Consultas

¡¡¡Leer cuidadosamente antes de comenzar las consultas!!!

La siguiente base de datos almacena información sobre una vuelta ciclista:

Equipo (nomequipo, director)
Ciclista (dorsal, nombre, edad, equipo)
Etapa (numetapa, kms, salida, llegada, ganador)
Puerto (nompuerto, altura, categoria, pendiente, etapa, primero)
Maillot (codigo, tipo, color, premio)
Lleva (ciclista, maillot, etapa)

La clave primaria de cada relación está formada por los atributos que aparecen subrayados.
La información que contienen las relaciones anteriores se describe a continuación:

- **Equipo**: contiene los datos de los distintos equipos: nombre (nomequipo) y nombre de su director (director).
- **Ciclista**: contiene los datos de los ciclistas que componen los distintos equipos: número del dorsal (dorsal), nombre del ciclista (nombre), edad del ciclista (edad) y nombre del equipo al que pertenece (equipo).
- **Etapa**: contiene los datos de las etapas que componen la vuelta ciclista: número de la etapa (numetapa) (las etapas se numeran consecutivamente: 1, 2, ...), kilómetros que tiene la etapa (kms), nombre de la población de donde sale la etapa (salida), nombre de la población donde está la meta de la etapa (llegada) y número del dorsal del ciclista que ha ganado la etapa (ganador). Los atributos salida y llegada están definidas sobre el mismo dominio.
- **Puerto**: contiene los datos de los puertos de montaña que visita la vuelta ciclista: nombre del puerto (nompuerto), altura máxima del puerto (altura), categoria del puerto: primera, especial, etc. (categoria), porcentaje que indica la pendiente media del puerto (pendiente), número de la etapa donde se sube el puerto (etapa) y número del dorsal que ha ganado el puerto al pasar en primera posición (primero).
- **Maillot**: contiene los datos de los premios que se otorgan mediante los distintos maillots: código del maillot (código), clasificación que premia ese maillot: general, montaña, etc. (tipo), color de la camiseta asociada (color) e importe del premio que corresponde al ciclista que termine la vuelta llevando el maillot (premio).
- **Lleva**: contiene la información sobre el dorsal de qué ciclistas (ciclista) han llevado cada maillot representado por su código (maillot) en cada una de las etapas (etapa).

Las particularidades de cada tabla se recogen en la hoja anexa en dos formatos:

1) Código de creación de la base de datos.
2) Diagrama de la base de datos.

IRON HACK

¿Harto de chapar algo que no te renta?

Olvida tus apuntes este verano
y ponte a programar 👩‍💻

Si no encuentras tu crush, por lo menos
dedícate a algo que te guste.

👉 **http://bit.ly/necesitouncambio**

Consultas:

1. Listado de ciclistas que pertenecen a algún equipo cuyo nombre de equipo contenga la palabra "mobile". El listado debe mostrar el nombre del ciclista, el dorsal y el equipo al que pertenece. Debe estar ordenado alfabéticamente por equipo en primer lugar, y dentro de cada equipo, deberán ordenarse alfabéticamente por nombre de ciclistas.

```
SELECT c.nombre, c.dorsal, c.equipo
FROM Ciclista c
WHERE c.equipo = ANY ( SELECT e.nomequipo FROM Equipo e
    WHERE e.nomequipo LIKE "%mobile%")
ORDER BY c.equipo, c.nombre; // Este order by primero ordena
                             // por equipo. Para las tuplas con el
                             // mismo equipo, ordena por nombre
```

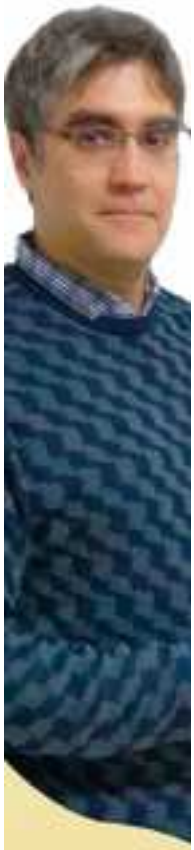
2. Listado de ciclistas que han llevado el maillot rojo. Para cada ciclista deberá aparecer el nombre del ciclista, dorsal, equipo al que pertenece y el número de veces que ha llevado dicho maillot. Deberá ordenarse de mayor a menor según el número de veces que llevaron el maillot rojo.

```
SELECT  c.nombre, c.dorsal, c.equipo, COUNT(*) nveces
FROM    Ciclista c, Maillot m, Lleva w
WHERE   w.ciclista = c.dorsal  AND  w.maillot = m.codigo
        AND m.color = "rojo"

GROUP BY c.dorsal // Solo la CP
ORDER BY  nveces  DESC ;
```

* NOTA: Después de hacer el producto Natural, tengo que agrupar por ciclista y contar el número de tuplas de cada grupo. Para agrupar, me basta con usar c.dorsal porque es la CP e identifica a cada ciclista inequívocamente

*Ejercicio 2: Igual que la anterior pero haciendo el producto natural con JOIN

```
SELECT c.nombre, c.dorsal, c.equipo, COUNT(*) n_veces
FROM (Lleva w JOIN Ciclista c on w.ciclista = c.dorsal)
   JOIN Maillot m on w.maillot = m.codigo

WHERE m.color = "rojo"
GROUP BY c.dorsal // Solo la CP
ORDER BY nveces DESC ;
```



* Ejercicio 2, otra manera

2. Listado de ciclistas que han llevado el maillot rojo. Para cada ciclista deberá aparecer el nombre del ciclista, dorsal, equipo al que pertenece y el número de veces que ha llevado dicho maillot. Deberá ordenarse de mayor a menor según el número de veces que llevaron el maillot rojo.

```
SELECT c.nombre, c.dorsal, c.equipo,
  ( SELECT COUNT (*)
    FROM Lleva w, Maillot m
    WHERE  w.maillot = m.codigo AND w.ciclista = c.dorsal
      AND m.color ="rojo"
  ) "nveces"

FROM Ciclista c
HAVING nveces  != 0 // nveces no puede ir en un where al
                    //ser resultado de
                    //una función de grupos
ORDER BY nveces DESC ;
```

* NOTA: A mí esta manera y la siguiente me resultan menos intuitivas, pero tal vez a ti te parezcan más claras 😉

* Ejercicio 2, otra manera más

```
SELECT c.nombre, c.dorsal, c.equipo,
  (SELECT COUNT(*)
    FROM Lleva w,
    WHERE EXIST
      (SELECT *
        FROM Maillot m
        WHERE m.color = "rojo"
          AND w.maillot=m.codigo
      ) AND w.ciclista = c.dorsal) "nveces"

FROM Ciclista c
HAVING nveces != 0 // nveces no puede ir en un where al
                   // ser resultado de
                   // una función de grupos
ORDER BY nveces DESC;
```

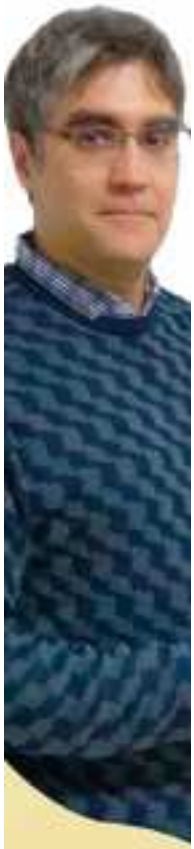
3. Realice una consulta que devuelva la siguiente información: Número de etapas de la vuelta ciclista, distancia que deben recorrer todos los ciclistas que completen todas las etapas y número diferentes de ganadores de etapa que ha habido en la vuelta ciclista.

```
SELECT COUNT(*) n_etapas, SUM(kms) Distancia,
       COUNT( DISTINCT ganador) ganadores
FROM Etapa ;
```

4. Listado de etapas en la que se muestre el número de puertos cuya categoría es "Primera" o "Segunda". Para cada registro deberá mostrarse la información tal y como se muestra en el siguiente ejemplo:

Etapa	Distancia	Salida	Llegada	Puertos de Cat 1 y 2
1	178	Zaragoza	Pamplona	4
2	214	Logroño	Aranda de Duero	2
3	159	Burgos	Bilbao	1

```
Select e.numetapa "Etapa", e.kms "Distancia",
e.salida "Salida", e.llegada "Llegada",
COUNT(*) "Puertos de Cat 1 y 2"

FROM Etapa e, Puerto p
WHERE e.numetapa = p.etapa
    AND (p.categoria = "Primera"
        OR p.categoria = "Segunda")

GROUP BY e.numetapa // Solo la CP
```



5. Listado de directores de equipo con menos de 5 victorias. Este listado no tendrá en cuenta las victorias de etapa de los ciclistas de nombre "Juan Pérez Casillas" ni las de "Manuel Sánchez Smith", pues fueron descalificados en el control antidoping. El listado deberá mostrar el nombre del director, el nombre del equipo y el número de victorias de etapa. Además, deberá ordenarse de mayor a menor número de victorias. Si dos directores tienen el mismo número de victorias deberá ordenarse alfabéticamente por sus nombres.

```
SELECT e.nomequipo, e.director, COUNT(*) "n_victorias"
FROM Equipo e, Etapa t, Ciclista c
WHERE c.dorsal=t.ganador AND c.equipo=e.nomequipo
  AND c.nombre != "Juan Pérez Casillas"
  AND c.nombre != "Manuel Sánchez Smith"
GROUP BY e.nom_equipo // Solo la CP
HAVING n_victorias < 5
ORDER BY n_victorias, e.director;
```

*NOTA: Este es similar al 2, también se puede hacer con JOIN.

6. Mostrar nombre, edad y equipo del ciclista más joven en haber ganado una etapa en la vuelta.

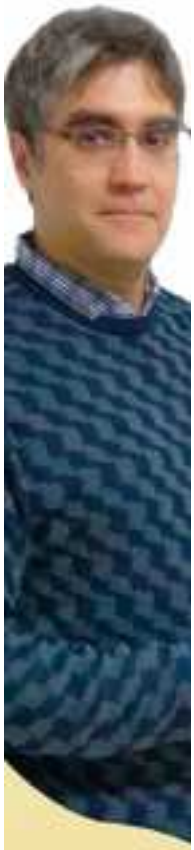
```
SELECT  c.nombre, c.edad, c.equipo
FROM    Ciclista c
WHERE   c.edad = ( SELECT MIN(d.edad)
                   FROM Ciclista d
                   WHERE EXISTS
                     (SELECT * FROM Etapa e
                      WHERE e.ganador = d.dorsal)
```

* Ejercicio 6, otra manera

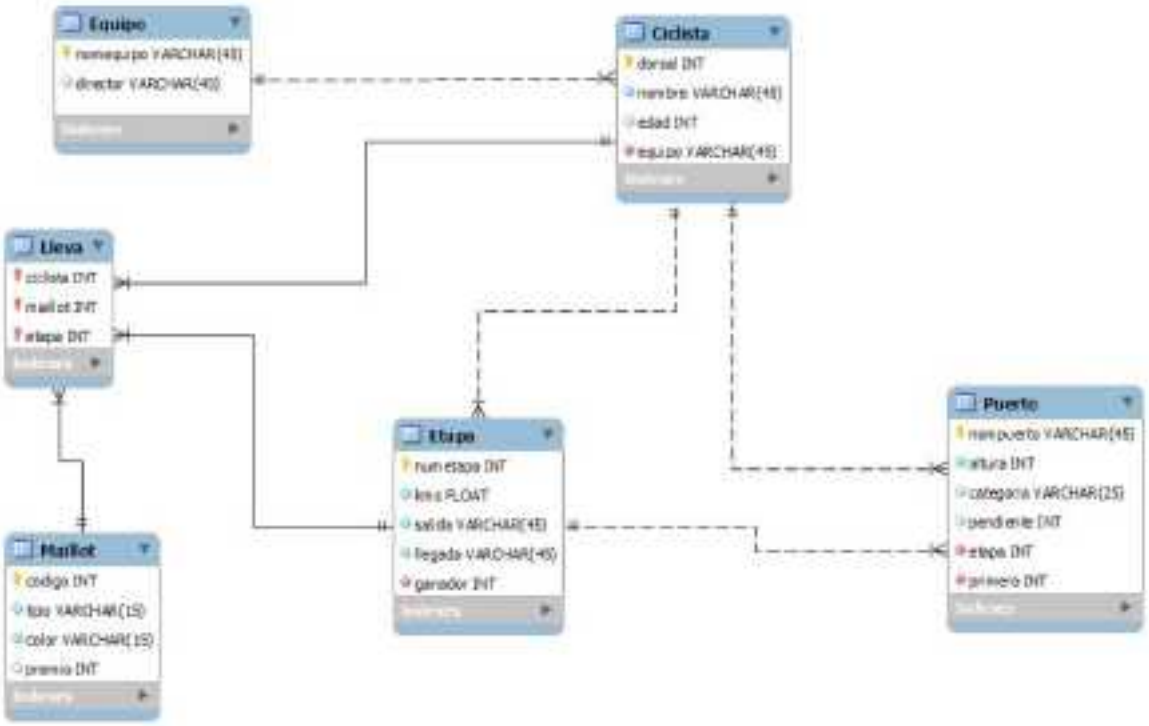
```
SELECT c.nombre, c.edad, c.equipo
FROM Ciclista c
WHERE c.EDAD <= ALL (SELECT d.edad
                     FROM Ciclista d
                     WHERE EXISTS
                       (SELECT * FROM Etapa e
                        WHERE e.ganador = d.dorsal))
```

Anexo con código de creación de tablas y diagrama correspondiente

```
CREATE TABLE Equipo (
  nomequipo VARCHAR(45) NOT NULL,
  director VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (nomequipo));

CREATE TABLE Ciclista (
  dorsal INT(11) NOT NULL,
  nombre VARCHAR(45) NOT NULL,
  edad INT(11) NULL DEFAULT NULL,
  equipo VARCHAR(45) NOT NULL,
  PRIMARY KEY (dorsal),
  FOREIGN KEY (equipo) REFERENCES Equipo (nomequipo));

CREATE TABLE Etapa (
  numetapa INT(11) NOT NULL,
  kms FLOAT(11) NOT NULL,
  salida VARCHAR(45) NOT NULL,
  llegada VARCHAR(45) NOT NULL,
  ganador INT(11) NOT NULL,
  PRIMARY KEY (numetapa),
  FOREIGN KEY (ganador) REFERENCES Ciclista (dorsal));

CREATE TABLE Puerto (
  nompuerto VARCHAR(45) NOT NULL,
  altura INT(11) NOT NULL,
  categoria VARCHAR(25) NULL DEFAULT NULL,
  pendiente INT(11) NULL DEFAULT NULL,
  etapa INT(11) NOT NULL,
  primero INT(11) NOT NULL,
  PRIMARY KEY (nompuerto),
  FOREIGN KEY (etapa)REFERENCES Etapa (numetapa),
  FOREIGN KEY (ciclista) REFERENCES Ciclista (primero));

CREATE TABLE Maillot (
  codigo INT(11) NOT NULL,
  tipo VARCHAR(15) NOT NULL,
  color VARCHAR(15) NOT NULL,
  premio INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (codigo));
```



```
CREATE TABLE Lleva (
  ciclista INT(11) NOT NULL,
  maillot INT(11) NOT NULL,
  etapaINT(11) NOT NULL,
  PRIMARY KEY (ciclista, maillot, etapa),
  FOREIGN KEY (ciclista) REFERENCES Ciclista (dorsal),
  FOREIGN KEY (maillot)REFERENCES Maillot (codigo)
  FOREIGN KEY (etapa)REFERENCES Etapa (numetapa));
```



La arista de unión entre dos tablas, muestra que hay una relación entre dichas tablas, pero para saber con certeza qué campos son los que están relacionados (clave primaria - clave foránea), observe el código SQL de creación de la tabla.