FICHEROS Y BASES DE DATOS **2º ITIG** **24/6/2003**

RESOLUCIÓN DEL EXAMEN

1. El agrupamiento es ..

Esto es lo que dice en los apuntes "En el agrupamiento los registros relacionados se almacenan juntos en el mismo fichero. Esta organización es apropiada cuando se espera utilizar muy a menudo una determinada relación en el acceso a los datos, ya que implementarla físicamente puede aumentar la eficiencia del sistema al acceder a los registros relacionados. "

(A) un modo de acceso a los datos, alternativo a los índices.

No es cierto, no es un modo de acceso, es un modo de almacenamiento.

(B) un modo de almacenamiento de datos, independiente de los índices.

Sí es cierto, es un modo de almacenamiento y es independiente de los índices.

(C) un modo de acceso a los datos que se puede utilizar junto con los índices.

No es cierto, no es un modo de acceso, es un modo de almacenamiento.

2. Una de las ventajas de los árboles B+ frente a los árboles B es que ...

(A) los algoritmos de inserción y borrado son muy eficientes.

No es cierto, esto no es una ventaja de los B+ frente a los B ya que los algoritmos son eficientes en ambos tipos de árboles.

(B) están equilibrados en altura (todos los nodos hoja están al mismo nivel).

No es cierto, esto no es una ventaja de los B+ frente a los B ya que ambos tipos de árboles están equilibrados (de ahí viene la B, de "balanced").

(C) permiten dos modos de acceso: directo y secuencial.

Sí es cierto, porque en los árboles B+se puede acceder también de modo secuencial a través de las hojas ya que éstas están encadenadas y en ellas se encuentran todos los punteros a los registros de datos.

3. Esta pregunta trata de redes. En una red de ordenadores hay varios switches numerados así: S01, S02, ... Cada switch tiene varios puertos numerados así: P1, P2, P3, ... Cada ordenador tiene un identificador único. En la base de datos queremos guardar información sobre los puertos a los que puede conectarse cada ordenador y hemos obtenido la siguiente tabla: CONEXIÓN(switch, puerto, ordenador). El puerto se identifica por la combinación (switch, puerto). ¿Cuál debe ser la clave primaria de la tabla de modo que se tenga en cuenta la siguiente restricción "dentro de un mismo switch, un ordenador sólo puede conectarse a uno de sus puertos"?

(A) La clave primaria estará formada por: switch, puerto, ordenador

No es cierto, porque con esta clave primaria podemos tener que un ordenador puede conectarse a varios puertos del mismo switch. Por ejemplo, podemos insertar estas dos filas (S01,P1,ord1) y (S01,P2,ord1).

(B) La clave primaria estará formada por: switch, ordenador

Sí es cierto, porque con esta clave primaria una misma pareja de switch y ordenador no se puede repetir.

(C) La clave primaria estará formada por: puerto, ordenador

No es cierto, porque con esta clave primaria un ordenador puede aparecer con varios puertos de un mismo switch.

4. ¿Cuándo es posible eliminar una etapa de la base de datos?

(A) Sólo cuando la etapa no se haya corrido.

No es cierto, las claves ajenas a ETAPA están en PUERTO y en LLEVA y la regla de borrado es propagar, por lo que siempre se podrá borrar una etapa, se haya corrido o no.

(B) Sólo si sus puertos aún no han sido ganados por ningún ciclista.

No es cierto, la clave ajena de PUERTO a CICLISTA no influye sobre el borrado en ETAPA. En todo caso, podrían influir las claves ajenas que hagan referencia a PUERTO o a LLEVA si las hubiera, pero no las hay.

(C) Siempre que se necesite hacerlo.

Sí, es cierto, y el borrado de la etapa implicará el borrado de todos sus puertos y de todas las filas de LLEVA que le correspondan.

5. Tras finalizar la vuelta ciclista, se desea obtener los datos de los equipos que no han ganado ninguna etapa. Dadas las siguientes expresiones:

```
E1: EQUIPOX WHERE ∀ETAPAX ∃CICLX
        (ETAPAX.dorsal=CICLX.dorsal AND CICLX.nomequipo<>EQUIPOX.nomequipo)
```

Esta expresión obtiene los datos de un equipo si los ciclistas que han ganado cada una de las etapas de la vuelta no son suyos, es decir, obtiene los equipos que no han ganado ninguna etapa.

```
E2: EQUIPOX WHERE ∀CICLX (IF CICLX.nomequipo=EQUIPOX.nomequipo
                          THEN NOT ∃ETAPAX(ETAPAX.dorsal=CICLX.dorsal))
```

Esta expresión obtiene los datos de un equipo si todos sus ciclistas cumplen que no han ganado ninguna etapa, es decir, obtiene los equipos que no han ganado ninguna etapa.

¿Cuál de las siguientes afirmaciones es cierta?

(A) La expresión E1 obtiene el resultado deseado, mientras que la expresión E2 no lo hace.

(B) La expresión E2 obtiene el resultado deseado, mientras que la expresión E1 no lo hace.

(C) Ambas expresiones obtienen el resultado deseado.

Sí es cierto, ambas expresiones son equivalentes.

6. Escribir una expresión del álgebra relacional que obtenga la suma total de kilómetros de todas las etapas que como mucho tienen tres puertos (hay que tener también en cuenta las que no tienen ninguno).

Una posible solución es la siguiente:

```
T1 := ETAPA[numetapa] MINUS PUERTO[numetapa]  -- etapas sin puertos
T2 := SUMMARIZE (ETAPA JOIN PUERTO) GROUPBY (numetapa) ADD COUNT(*) AS num_ptos
T3 := (T2 WHERE num_ptos <= 3) [numetapa]   -- etapas con 1, 2 ó 3 puertos
T4 := T1 UNION T2    -- etapas con 0, 1, 2 ó 3 puertos
RDO := SUMMARIZE (T4 JOIN ETAPA) GROUPBY( ) ADD SUM(kms) AS kms_totales
```

7. Escribir una expresión del cálculo relacional que obtenga los datos del ciclista/s que ha ganado todos los puertos de una misma etapa.

Una posible solución es la siguiente:

```
CICLISTAX WHERE ∃ ETAPAX ∃ PUERTOX
    (ETAPAX.numetapa = PUERTOX.numetapa AND
     ∀ PUERTOY ( IF ETAPAX.numetapa = PUERTOY.numetapa
                 THEN CICLISTAX.dorsal=PUERTOY.dorsal)
```

La comprobación de que la etapa tenga algún puerto (∃ PUERTOX) debe hacerse porque sino, con las etapas que no tienen puertos, el predicado ∀ PUERTOY (IF ...) es verdadero, cuando queremos que no lo sea.

8. Dadas las sentencias S1 y S2:

```
S1: SELECT count(*)
    FROM   puerto
    WHERE  dorsal is not null
    AND    pendiente > 10;
```

```
S2: SELECT count(dorsal)
    FROM   puerto
    WHERE  pendiente > 10;
```

La sentencia S1 obtiene los puertos que ya han sido ganados y que tienen una pendiente mayor del 10%, y los cuenta.

La sentencia S2 obtiene los puertos cuya pendiente es mayor del 10% y cuenta cuántos de ellos han sido ganados (los que no han sido ganados tienen dorsal NULL y se ignoran al contar).

(A) La sentencia S1 obtiene sólo un subconjunto de las filas que obtiene la sentencia S2.

(B) La sentencia S2 obtiene sólo un subconjunto de las filas que obtiene la sentencia S1.

(C) Ambas sentencias devuelven el mismo resultado.

Sí es cierto, ambas expresiones son equivalentes.

9. Se pretende obtener el nombre y el director de los equipos cuyos ciclistas no han ganado ninguna etapa o que, como mucho, han ganado una. Dada la siguiente sentencia SELECT ¿qué debe aparecer en la cláusula WHERE?

```
SELECT q.nomequipo, q.director
FROM   etapa e, ciclista c, equipo q
WHERE  ...
GROUP BY q.nomequipo, q.director
HAVING count(e.numetapa) < 2;
```

(A) `WHERE e.dorsal (+) = c.dorsal AND c.nomequipo = q.nomequipo`

Es correcto. Para que los ciclistas que no han ganado ninguna etapa no se pierdan al hacer el JOIN, debemos utilizar un OUTER JOIN, incluyendo una "etapa fantasma" para ellos .

(B) `WHERE e.dorsal = c.dorsal (+) AND c.nomequipo = q.nomequipo`

No es correcto, al hacer el OUTER JOIN con CICLISTA permanecen las etapas que aún no se han corrido. Estas etapas, que no se quieren para nada, luego se pierden al hacer el JOIN con equipo.

(C) `WHERE e.dorsal = c.dorsal (+) AND c.nomequipo (+) = q.nomequipo`

No es correcto, al hacer el OUTER JOIN con CICLISTA permanecen las etapas que aún no se han corrido. Al igual que en el apartado anterior, estas etapas, que no se quieren para nada, luego se pierden al hacer el OUTER JOIN con equipo (si esas etapas se quisieran conservar, que no es el caso, el "fantasma" se debería introducir en EQUIPO y no en CICLISTA).

10. Se desea obtener el nombre y el director de los equipos que, en alguna etapa, sus ciclistas han llevado tres o más maillots. Dada la siguiente sentencia SELECT:

```
SELECT DISTINCT q.nomequipo, q.director
FROM   lleva l, ciclista c, equipo q
WHERE  l.dorsal = c.dorsal
AND    c.nomequipo = q.nomequipo
GROUP BY ...
HAVING ...;
```

¿Qué debe aparecer en las cláusulas GROUP BY y HAVING?

(A) `GROUP BY q.nomequipo, q.director`
`HAVING COUNT(DISTINCT l.dorsal) >= 3;`

No es correcto. Así nos quedamos con los equipos que tienen tres o más ciclistas que han llevado maillot en alguna ocasión durante la vuelta ciclista.

(B) `GROUP BY l.numetapa, q.nomequipo, q.director`
`HAVING COUNT(*) >= 3;`

Es correcto: agrupamos por equipo y por etapa y contamos cuántas filas hay en el grupo (son filas de LLEVA); si hay tres filas o más es porque en esa etapa los ciclistas del equipo han llevado tres o más maillots.

(C) `GROUP BY q.nomequipo, q.director`
`HAVING COUNT(l.numetapa) >= 3;`

No es correcto. El atributo numetapa de LLEVA forma parte de su clave primaria, por lo que COUNT(l.numetapa) es aquí lo mismo que COUNT(*): nos quedamos con los equipos que han llevado maillots en tres o más ocasiones durante toda la vuelta ciclista.

11. Escribir una sentencia SELECT que obtenga el dorsal del ciclista que ha llevado el maillot amarillo durante más etapas.

```
SELECT l.dorsal
FROM lleva l, maillot m
WHERE l.codigo = m.codigo
AND m.color = 'amarillo'
GROUP BY l.dorsal
HAVING COUNT(*) = ( SELECT MAX(COUNT(*))
                    FROM lleva l, maillot m
                    WHERE l.codigo = m.codigo
                    AND m.color = 'amarillo'
                    GROUP BY l.dorsal );
```

12. Escribir una sentencia SELECT que muestre los datos de los ciclistas que solamente han ganado puertos de tercera categoría.

```
SELECT *
FROM ciclista
WHERE dorsal IN ( SELECT dorsal FROM puerto WHERE categoria = 'tercera'
                  MINUS
                  SELECT dorsal FROM puerto WHERE categoria <> 'tercera' );
```

13. Obtener el esquema de la base de datos correspondiente a las entidades **MONITOR**, **CLASE**, **SALA** y **APARATO** del esquema conceptual, teniendo también en cuenta las relaciones que existen entre ellas: **imparte**, **asignada** y **tiene**.

MONITOR(dni, nombre, telf, titu, exper)

MONITOR.titu acepta nulos

PREPARACION(dni, tipo, año)

PREPARACION.dni es clave ajena a MONITOR -- nulos: no -- borrado: propagar

SALA(numsala, ubicación, tipo, metros)

CLASE(codclase, tipo, día, hora, dni_monitor, numsala)

CLASE.dni_monitor es clave ajena a MONITOR -- nulos: no -- borrado: restringir/propagar

CLASE.numsala es clave ajena a SALA -- nulos: no -- borrado: restringir/propagar

APARATO(código, descripción, estado, numsala)

APARATO.numsala es clave ajena a SALA -- nulos: sí -- borrado: restringir/propagar

14. Añadir al esquema de la base de datos las tablas correspondientes a las entidades **SOCIO** y **SQUASH**, teniendo también en cuenta las relaciones **reserva** y **asiste**.

SOCIO(numsocio, nombre, telf, dirección, profesión, ctabanco)

ASISTE(numsocio, codclase)

ASISTE.numsocio es clave ajena a SOCIO -- nulos: no -- borrado: propagar

ASISTE.codclase es clave ajena a CLASE -- nulos: no -- borrado: restringir/propagar

SQUASH(numpista, ubicación, estado)

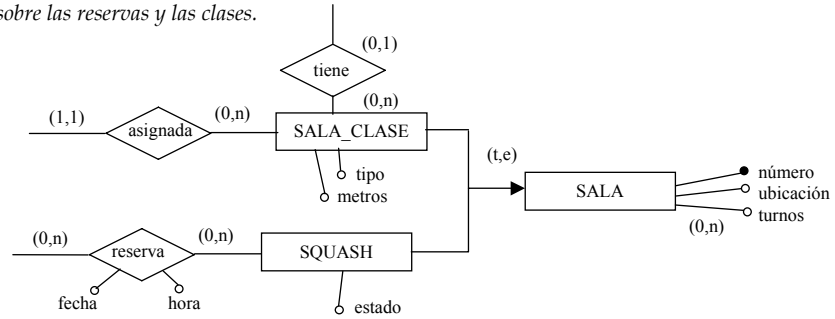
RESERVA(numpista, fecha, hora, numsocio)

RESERVA.numpista es clave ajena a SQUASH -- nulos: no -- borrado: restringir/propagar

RESERVA.numsocio es clave ajena a SOCIO -- nulos: sí -- borrado: anular/restringir/propagar

15. Modificar el esquema conceptual (es decir, el diagrama entidad-relación) para que se reflejen los turnos de limpieza de todas las salas, ya sean de clase o de squash. Los turnos de limpieza son una serie de horas en las que se debe acudir a la sala a limpiar.

Lo que debemos hacer es tratar las salas de clase y las pistas de squash como subconjuntos de una entidad genérica SALA en la que incluiremos el atributo correspondiente a los turnos de la limpieza. A continuación se muestra la parte del esquema que se ha modificado. Se puede ver que se respetan las restricciones sobre las reservas y las clases.



16. Introduce los cambios necesarios en el esquema de la base de datos que has obtenido en los apartados 13 y 14 para que refleje los cambios realizados en el apartado 15. No olvides especificar las reglas de las nuevas claves ajenas que aparezcan.

La tabla SALA la sustituimos por SALA_CLASE(número, tipo, metros)

y la clave ajena CLASE.numsala es ahora clave ajena a SALA_CLASE (mantiene las mismas reglas)

La tabla SQUASH la sustituimos por SQUASH(número, estado)

Incluimos una nueva tabla:

SALA(número, ubicación)

SALA_CLASE.número es clave ajena a SALA -- nulos: no -- borrado: propagar

SQASH.número es clave ajena a SALA -- nulos: no -- borrado: propagar

TURNOS(número, hora)

TURNOS.número es clave ajena a SALA -- nulos: no -- borrado: propagar