Examen de Ficheros y Bases de Datos – F47
21 de junio de 2002

1. Se ha diseñado un fichero disperso utilizando la dispersión extensible. ¿Cuándo aumentará el tamaño del directorio?

(A) Cuando el número de registros en un bloque supere el 69 %.

(B) Cuando se llene un bloque con profundidad local menor a la global.

(C) Cuando se llene un bloque con profundidad local igual a la global.

La respuesta correcta es la (C). Cuando se llena un bloque es necesario "desdoblarlo". Cuando su profundidad local es menor que la global, se tienen entradas en el directorio que se pueden utilizar para apuntar al nuevo bloque. Pero cuando su profundidad es igual que la global, es necesario aumentar el tamaño del directorio para crear la entrada que apuntará al nuevo bloque.

2. Dado un árbol B de orden 100 y dado un determinado nodo interno que tiene 95 punteros. ¿Cuántos valores del campo de indexación hay en dicho nodo?

(A) 47 valores.

(B) 94 valores.

(C) 95 valores.

La respuesta correcta es la (A). En un árbol B de orden 100 puede haber hasta 100 punteros a nodos del árbol, 99 punteros al fichero de datos y 99 valores. Si el nodo que estamos mirando tiene 95 punteros, 48 serán punteros a nodos internos y 47 serán punteros al fichero de datos. Como en cada nodo hay tantos valores del campo de indexación como punteros al fichero de datos, en este nodo habrá 47 varlores.

3. Los nodos de un árbol B+ ...

(A) ... siempre están por encima de la mitad de su capacidad.

(B) ... siempre están por encima del 69 % de su capacidad.

(C) ... siempre están entre el 50 % y el 69 % de su capacidad

La respuesta correcta es la (A). En la defición de la estructura de datos del árbol B+ se especifica que sus nodos siempre deben estar al 50 % de su capacidad. Del mantenimiento de esta propiedad se encargan los algoritmos que realizan las actualizaciones sobre el árbol.

4. Un índice, sea del tipo que sea, también se puede construir sobre una combinación de campos. Por ejemplo, sobre un fichero de personas se puede construir un índice combinado sobre <apellido1, apellido2, nombre> siendo éstos tres campos del fichero. Las entradas del índice estarán ordenadas del mismo modo que ordenaría la cláusla `ORDER BY apellido1, apellido2, nombre` en una consulta de SQL. ¿En qué tipo de consultas podemos utilizar el índice para acelerar el acceso a los datos?

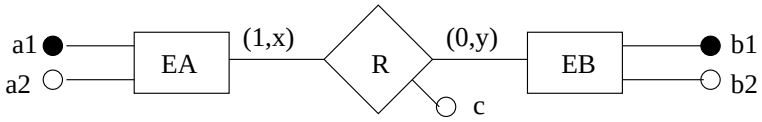
(A) En consultas en las que se especifica sólo apellido1, en consultas en las que se especifica sólo apellido1 y apellido2, y en consultas en las que se especifica apellido1, apellido2 y nombre.

(B) En consultas en las que se especifique sólo apellido1, o sólo apellido2, o sólo nombre. También en consultas en las que se especifique sólo apellido1 y apellido2, o sólo apellido1 y nombre, o sólo apellido2 y nombre. Y también en consultas en las que se especifiquen los tres campos.

(C) Ninguna. Un índice tan raro no puede servir para muchas cosas.

La respuesta correcta es la (A). En el índice los nombres completos aparecen ordenados como en listado de las notas de la asignatura. Si buscamos la nota de alguien que se llama López de primer apellido, podemos utilizar el índice haciendo una búsqueda binaria sobre él porque está ordenado por ese campo. Sin embargo, si buscamos la nota de alguien que se llama López de segundo apellido, el índice no nos ayuda a acelerar la búsqueda ya que hemos de recorrerlo completo al no estar ordenado por ese campo. El segundo apellido, en realidad, se utiliza en el índice para ordenar aquellas entradas que tienen el mismo valor en el primer apellido. Por lo tanto, las búsquedas que sacan partido del índice son aquellas que especifican todos los valores de los campos del índice o bien especifican los valores de un subconjunto contiguo de los campos en donde se especifique, por lo menos, el primer campo.

5. Dado el siguiente esquema conceptual:



donde $x \in \{1, n\}$ e $y \in \{1, n\}$ y dadas las siguientes tablas: `EA(a1,a2)`, `EB(b1,b2)` y `R(a1,b1,c)`.

(A) La clave ajena `R.a1` no acepta nulos y la clave ajena `R.b1` sí los acepta.

(B) Ambas claves ajenas `R.a1` y `R.b1` no aceptan nulos.

(C) Sólo sabremos si las claves ajenas `R.a1` y `R.b1` aceptan nulos una vez se decida qué atributos forman la clave primaria de `R` (será en función de las cardinalidades máximas).

La respuesta correcta es la (B). En el modelo relacional podemos expresar las relaciones entre entidades de dos formas. Una forma es mediante una clave ajena en la tabla que representa a una de las entidades. En esa tabla hay una fila por cada ocurrencia de la entidad y la clave ajena aceptará nulos en función de que la participación de esa entidad en la relación sea o no obligatoria (1 ó 0 en la cardinalidad mínima).

La otra forma es mediante una nueva tabla que tiene una fila por cada ocurrencia de la relación y tiene una clave ajena a cada una de las tablas que representan a las entidades participantes. Existe una ocurrencia de la relación cuando una ocurrencia de una de las entidades participantes se relaciona con una ocurrencia de la otra entidad. ¿Qué pasa si una de las entidades participa de forma opcional en la relación? Pues que aquellas ocurrencias de la entidad que no se relacionen con ninguna ocurrencia de la otra entidad NO aparecen en la tabla, sólo aparecen ocurrencias que están relacionadas por lo que ninguna de las claves ajenas a las entidades participantes debe aceptar nulos: si no hay relación, no hay fila.

6. Dada la siguiente expresión del álgebra relacional:
`((SUMMARIZE ETAPA GROUPBY() ADD MAX(kms) AS kms) JOIN ETAPA) JOIN CICLISTA`
¿A cuál de las siguientes expresiones es equivalente?

(A) `CICLISTAX WHERE ∀ ETAPAX (IF ETAPAX.dorsal=CICLISTAX.dorsal THEN NOT ∃ ETAPAY (ETAPAY.kms<=ETAPAX.kms))`

(B) `CICLISTAX WHERE ∃ ETAPAX (ETAPAX.dorsal=CICLISTAX.dorsal AND ∀ ETAPAY (ETAPAX.kms>=ETAPAY.kms))`

(C) `CICLISTAX WHERE ∀ ETAPAX (ETAPAX.dorsal=CICLISTAX.dorsal AND NOT ∃ ETAPAY (ETAPAY.kms<=ETAPAX.kms))`

La expresión del álgebra obtiene los datos del ciclista que ha ganado la etapa más larga (la que tiene más kilómetros). Si son varias las etapas "más largas", la expresión obtiene los datos de los ciclistas que han ganado cada una de ellas.

La expresión de (A) obtiene los datos de los ciclistas que para cada etapa que han ganado, no hay ninguna etapa que tenga los mismos o más kilómetros. Esta expresión no obtiene ninguna fila porque como también se compara cada etapa consigo misma, nunca será cierto que una etapa tenga más kilómetros que todas (por lo menos es igual a ella misma).

La expresión de (B) obtiene los datos de los ciclistas que han ganado alguna etapa cuyo número de kilómetros es mayor o igual que el número de kilómetros de todas las etapas, es decir, obtiene los datos de los ciclistas que han ganado las etapas con el mayor número de kilómetros. Esta expresión obtiene el mismo resultado que la del álgebra.

La expresión de (C) obtiene los datos del ciclista que ha ganado todas las etapas de la vuelta (¡tiene que haberlas ganado todas!) y además, cada una de esas etapas tiene una longitud superior a las demás. Tal y como pasa en (A), como también se compara cada etapa consigo misma, nunca será cierto que una etapa tenga más kilómetros que todas.

7. Dada la siguiente expresión del cálculo relacional:
`CICLISTAX WHERE NOT ∃ ETAPAX (ETAPAX.dorsal=CICLISTAX.dorsal) AND ∃ LLEVAX (LLEVAX.dorsal=CICLISTAX.dorsal)`
¿A cuál de las siguientes expresiones es equivalente?

(A) `(LLEVA[dorsal] INTERSECT ETAPA[dorsal]) JOIN CICLISTA`

(B) `(LLEVA[dorsal] JOIN ETAPA[dorsal]) JOIN CICLISTA`

(C) `(LLEVA[dorsal] MINUS ETAPA[dorsal]) JOIN CICLISTA`

La expresión del cálculo relacional obtiene los datos de los ciclistas que no han ganado ninguna etapa y que han llevado algún maillot.

La expresión de (A) obtiene los datos de los ciclistas que han llevado algún maillot y que además han ganado alguna etapa.

La expresión de (B) obtiene el mismo resultado que la expresión de (A).

Por eliminación, ya podríamos saber que la respuesta correcta es la (C) ya que las dos anteriores obtienen el mismo resultado y no puede haber dos respuestas verdaderas. Aún así, veamos qué obtiene la expresión de (C): los datos de los ciclistas que han llevado algún maillot y que no han ganado ninguna etapa, exactamente lo mismo que obtiene la expresión del cálculo del enunciado.

8. Escribe una expresión del álgebra relacional que obtenga el nombre de los equipos que tienen la media de edad mayor de todos los equipos.

Los equipos con una media de edad mayor son aquellos para los que la media del año de nacimiento de sus ciclistas es menor.

```
T1 := SUMMARIZE CICLISTA GROUPBY(nomequipo) ADD AVG(añonacim) AS media
T2 := SUMMARIZE T1 GROUPBY() ADD MIN(media) AS min_media
RDO := ((T1 TIMES T2) WHERE media = min_media)[nomequipo]
```

Si en T2 se llama a la nueva columna igual que a la columna que contiene la media, el resultado se obtiene con una operación de JOIN:

```
T2 := SUMMARIZE T1 GROUPBY() ADD MIN(media) AS media
RDO := (T1 JOIN T2) [nomequipo]
```

También es correcto si se obtiene el máximo de la media de la edad:

```
T1 := SUMMARIZE CICLISTA GROUPBY(nomequipo) ADD AVG(2002-añonacim) AS media
T2 := SUMMARIZE T1 GROUPBY() ADD MAX(media) AS max_media
RDO := ((T1 TIMES T2) WHERE media = max_media)[nomequipo]
```

9. Se desea conocer el nombre de cada ciclista mayor de 25 años junto con el número de puertos que ha ganado en toda la vuelta. El resultado debe mostrar todos los ciclistas mayores de 25 años, aunque no hayan ganado ningún puerto. Partimos de la siguiente sentencia:

```
SELECT c.dorsal, c.nombre
FROM puerto p, ciclista c
WHERE p.dorsal = c.dorsal
AND TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))−c.añonacim > 25
GROUP BY c.dorsal, c.nombre;
```

(A) Para obtener el resultado deseado, la primera condición de la cláusula `WHERE` debería ser "`p.dorsal (+) = c.dorsal`" y la cláusula `SELECT` debería ser "`SELECT c.dorsal, c.nombre, COUNT(*)`"

(B) Para obtener el resultado deseado, la cláusula `WHERE` debería ser "`WHERE p.dorsal = c.dorsal (+)`", la restricción sobre la edad se debería situar en el `HAVING` y la cláusula `SELECT` debería ser "`SELECT c.dorsal, c.nombre, COUNT(DISTINCT p.nompuerto)`"

(C) Para obtener el resultado deseado, la primera condición de la cláusula `WHERE` debería ser "`p.dorsal (+) = c.dorsal`" y la cláusula `SELECT` debería ser "`SELECT c.dorsal, c.nombre, COUNT(p.nompuerto)`"

La respuesta de (A) no es correcta porque al hacer el join externo, aquellos cicilistas que no han ganado ningún puerto obtienen como resultado del `COUNT(*)` un 1, ya que esta expresión cuenta filas y para los ciclistas que no han ganado puertos hay una fila en el resultado (la que resulta de concatenar el ciclista con una fila de nulos por parte del puerto "que no ha ganado").

La respuesta de (B) no es correcta porque el join externo hay que hacerlo por el lado de los puertos (ciclistas que no han ganado ningún puerto) y no por el lado de los ciclistas

(puertos que no han sido ganados por ningún ciclista). Además, la restricción sobre la edad no se debe situar en el `HAVING` ya que sólo se quiere tener en cuenta los ciclistas que la cumplen, por lo que se debe hacer en el `WHERE`.

La respuesta de (C) es la correcta. Es similar a la de (A) pero al contar sobre la columna `p.nompuerto` se consigue que los ciclistas que no han ganado ningún puerto tengan como resultado de `COUNT(p.nompuerto)` un cero, porque `p.nompuerto` es nulo y `COUNT` no cuenta los nulos.

10. Dada la siguiente sentencia ¿cuál de las siguientes afirmaciones es cierta?

```
SELECT DISTINCT l1.codigo
FROM lleva l1
WHERE NOT EXISTS (SELECT *
                  FROM lleva l2
                  WHERE l2.codigo = l1.codigo
                  AND l1.dorsal <> l2.dorsal);
```

(A) Obtiene los maillots que sólo han sido llevados por un ciclista.

(B) Obtiene los maillots que sólo han sido llevados por más de un ciclista.

(C) Obtiene los maillots que nunca han sido llevados por ningún ciclista.

La respuesta correcta es la (A) ya que un maillot cumplirá la restricción del `WHERE` si siempre ha sido llevado por un mismo ciclista: no existe ninguna otra fila en la tabla que corresponda al mismo maillot y a un ciclista distinto.

11. Dada la siguiente sentencia SQL:

```
SELECT COUNT(DISTINCT dorsal)
FROM etapa
WHERE dorsal IS NULL;
```

(A) Obtiene el número de etapas que restan para terminar la vuelta ciclista.

(B) Obtiene el valor 0.

(C) Obtiene el valor 1.

La respuesta correcta es la (B) porque `COUNT` no cuenta los nulos.

12. Dadas las sentencias que se muestran a continuación:

```
S1:SELECT c1.dorsal, c1.nombre, COUNT(*)
   FROM etapa e, ciclista c1
   WHERE 5 < (SELECT COUNT(*)
              FROM ciclista c2
              WHERE c2.nomequipo = c1.nomequipo)
   AND e.dorsal = c1.dorsal
   GROUP BY c1.dorsal, c1.nombre;
```

```
S2:SELECT c.dorsal, c.nombre, COUNT(*)
   FROM etapa e, ciclista c
   WHERE e.dorsal = c.dorsal
   GROUP BY c.dorsal, c.nombre;
   HAVING COUNT(DISTINCT c.nomequipo) > 5;
```

(A) Ambas sentencias obtienen siempre el mismo resultado.

(B) La sentencia S1 obtiene un número de filas mayor o igual que el número de filas que obtiene la sentencia S2.

(C) La sentencia S2 obtiene un número de filas mayor o igual que el número de filas que obtiene la sentencia S1.

La sentencia S1 obtiene los ciclistas que pertenecen a un equipo de más de cinco corredores y cuenta el número de etapas que ha ganado cada uno. La sentencia S2 no obtiene ninguna fila porque se está agrupando por la clave primaria de la tabla `CICLISTA` y la restricción del `HAVING` contiene `COUNT(DISTINCT c.nomequipo)`, que siempre valdrá 1 porque en cada grupo habrá un solo valor (cada ciclista está en un solo equipo).

Por lo tanto, la respuesta correcta es la (B).

13. Escribe una sentencia SQL que obtenga los datos de los ciclistas que sólo han ganado etapas con algún puerto de montaña.

```
SELECT c.dorsal, c.nombre, c.añonacim, c.nomequipo
FROM ciclista c, etapa e, puerto p
WHERE e.dorsal = c.dorsal
AND p.numetapa = e.numetapa
MINUS
SELECT c.dorsal, c.nombre, c.añonacim, c.nomequipo
FROM ciclista c, etapa e
WHERE e.dorsal = c.dorsal
AND e.numetapa not in ( SELECT numetapa FROM puerto );
```

14. Escribe una sentencia SQL que obtenga los datos del ciclista ganador de la etapa con el puerto más alto (una vez ha terminado la vuelta).

```
SELECT c.dorsal, c.nombre, c.añonacim, c.nomequipo
FROM ciclista c, etapa e, puerto p
WHERE e.dorsal = c.dorsal
AND p.numetapa = e.numetapa
AND p.altura = ( SELECT max( altura ) FROM puerto );
```

15. Obtener el esquema de la base de datos correspondiente a las entidades **ADMINISTRADOR** y **COMUNIDAD DE VECINOS** del esquema conceptual, teniendo también en cuenta la relación que existe entre ellas: **gestiona**.

ADMINISTRADOR(dni, nombre)
CODPOSTAL(codpostal, población)
COMUNIDAD(código, nombre, calle, codpostal, dni, honorarios)

codpostal es clave ajena a CODPOSTAL
no admite nulos
regla de borrado: restringir
dni es clave ajena a ADMINISTRADOR
no admite nulos
regla de borrado: restringir (propagar es menos probable)

16. Añadir al esquema de la base de datos la entidad **PROPIEDAD** con su jerarquía y la relación **consta**. Nótese que es posible que un propietario posea varias propiedades.

PROPIETARIO(dni, nombre, telf_contacto, calle, codpostal)
codpostal es clave ajena a CODPOSTAL
no admite nulos
regla de borrado: restringir
calle y codpostal admiten nulos (ambos a la vez)
PROPIEDAD(código, portal, planta, puerta, porcentaje, numcuenta,
dni_prop, nom_inquilno, telf_inquilino)
código es clave ajena a COMUNIDAD
no admite nulos
regla de borrado: propagar
dni_prop es clave ajena a PROPIETARIO
no admite nulos
regla de borrado: restringir
nom_inquilno y telf_inquilino admiten nulos (ambos a la vez)
VIVIENDA(código, portal, planta, puerta, num_habitaciones)
(código, portal, planta, puerta) forman una clave ajena a PROPIEDAD
no admite nulos
regla de borrado: propagar (porque es una subentidad de una jerarquía)
OFICINA(código, portal, planta, puerta, actividad)
(código, portal, planta, puerta) forman una clave ajena a PROPIEDAD
no admite nulos
regla de borrado: propagar (porque es una subentidad de una jerarquía)
LOCAL_COMERCIAL(código, portal, planta, puerta, tipo_comercio, horario)
(código, portal, planta, puerta) forman una clave ajena a PROPIEDAD
no admite nulos
regla de borrado: propagar (porque es una subentidad de una jerarquía)
horario admite nulos

17. Añadir al esquema de la base de datos la información de las relaciones **presidente** y **vocal**. Es importante que el propio esquema de la base de datos recoja la restricción de que un presidente y un vocal sólo pueden ejercer este papel en su propia comunidad.

PRESIDENTE(código, portal, planta, puerta)
(código, portal, planta, puerta) forman una clave ajena a PROPIEDAD
no admite nulos
regla de borrado: propagar/restringir
VOCAL(código, portal, planta, puerta)

(código, portal, planta, puerta) forman una clave ajena a PROPIEDAD
no admite nulos
regla de borrado: propagar/restringir

Las filas de ambas tablas son una clave ajena a la tabla PROPIEDAD. Si una propiedad es referenciada desde una fila de PRESIDENTE, es porque esa propiedad ejerce de presidente de la comunidad a la que pertenece. Del mismo modo, si una propiedad es referenciada desde una fila de VOCAL, esa propiedad ejerce de vocal en la comunidad a la que pertenece

La clave primaria de PRESIDENTE es sólo el código de la comunidad porque hay un único presidente por comunidad. La clave primaria de VOCAL es toda la clave ajena porque en una comunidad hay varios vocales.