FICHEROS Y BASES DE DATOS **2º ITIG** **26/6/99**

TEST (2.5 puntos)

1. ¿A qué se deben los problemas de fragmentación interna cuando un disco se divide en sectores?
A que sectores y clusters tienen tamaño fijo.

2. ¿Cómo se busca un registro por un campo clave en un fichero desordenado?
Mediante búsqueda lineal.

3. ¿En qué consiste el agrupamiento o *clustering*?
En almacenar juntos los registros de varios ficheros porque hay referencias entre ellos.

4. ¿En qué consiste el *hashing*?
En determinar la dirección de un registro en el fichero mediante la aplicación de una función sobre uno de sus campos.

5. ¿Por qué en el *hashing* extensible el acceso al directorio es mas eficiente que en el *hashing* dinámico?
Porque el directorio es un vector, mientras que en el dinámico es un árbol.

6. En un índice multinivel, a partir del segundo nivel ...
todos los índices son no densos.

7. En los arboles B+, los punteros a los registros de datos se encuentran ...
solamente en los nodos hoja.

8. ¿ Por qué no es conveniente utilizar un índice si se accede a un gran número de los registros de un fichero?
Porque el número de accesos a disco al utilizar el índice puede ser mayor que en el acceso secuencial.

9. En un sistema de ficheros ¿quién se encarga de mantener la integridad de los datos?
los programas de aplicación que gestionan los ficheros.

10. ¿Qué significa que un lenguaje de manejo de datos es navegacional?
Que los registros se seleccionan por posición lógica, haciendo referencia al último registro accedido.

11. El poder utilizar distintos formatos para los mismos datos es ...
un inconveniente de los sistemas ficheros.

12. ¿Por qué interesa que las tablas de una base de datos relacional se encuentren en primera forma normal?
Porque de este modo las operaciones sobre los datos son más simples.

13. ¿Cual de los siguientes tipos de relaciones no se almacenan físicamente como tales?
Las vistas.

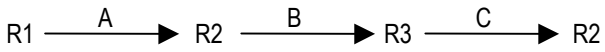
14. Las claves primarias deben cumplir la propiedad de unicidad ...
porque mediante ellas se direccionan las tuplas.

15. Que la clave primaria cumpla la propiedad de minimalidad (irreducibilidad) significa:
que si algún valor de la clave primaria tiene algún componente desconocido, no se puede identificar la tupla a la que corresponde.

16. Cada valor de una clave ajena compuesta por varios atributos ...
o tiene todos sus componentes nulos o todos no nulos.

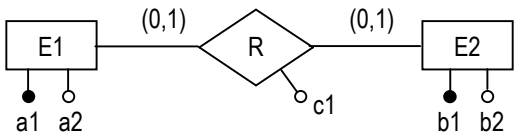
17. La regla de integridad referencial dice:
que la base de datos no debe contener ningún valor de clave ajena que siendo no nulo, no coincide con algún valor de la clave primaria a la que referencia.

18. Dado el siguiente diagrama referencial:



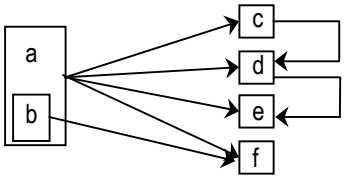
al menos una de las claves ajenas R2.B o R3.C debe aceptar nulos a causa del ciclo.

19. Dado el siguiente esquema conceptual, determinar el esquema lógico relacional correspondiente:



E1(<u>a1</u>,a2), E2(<u>b1</u>,b2), R(<u>a1</u>,b1,c1), R.a1 y R.b1 son claves ajenas a E1 y E2 respectivamente

20. Dada la relación R(a,b,c,d,e,f) en la que existen las siguientes dependencias funcionales:



¿de qué modo se debe descomponer R para evitar posibles anomalías en inserciones, borrados y modificaciones?

R1(a,b,c),R2(c,d),R3(d,e),R4(b,f)

CUESTIONES SQL (1 punto)

1. La siguiente sentencia trata de obtener el número de clientes de la ciudad de Castellón. Comenta cualquier observación que tengas acerca de su estructura.

```
SELECT COUNT(*) clientes
FROM   clientes c, pueblos p
WHERE  c.codpue = p.codpue
GROUP BY p.codpue, p.nombre
HAVING UPPER(p.nombre) = 'castellon';
```

En esta sentencia no hay que agrupar por pueblo ya que sólo interesan los clientes de un determinado pueblo. La restricción sobre el pueblo debe hacerse en el WHERE y además, cuando se compara con la función UPPER la cadena debe escribirse en mayúsculas. La sentencia correcta es la siguiente:

```
SELECT COUNT(*) clientes
FROM   clientes c, pueblos p
WHERE  c.codpue = p.codpue
AND    UPPER(p.nombre) = 'CASTELLON';
```

2. La siguiente sentencia trata de obtener un listado con todas las provincias y el número de facturas de cada una. ¿Obtiene su propósito? Razona la respuesta.

```
SELECT pr.nombre, COUNT(f.codfac)
FROM   facturas f, clientes c, pueblos pu, provincias pr
WHERE  f.codcli = c.codcli AND c.codpue = pu.codpue
AND    pu.codpro = pr.codpro
GROUP BY pr.codpro, pr.nombre;
```

No, sólo obtiene las provincias en donde se han hecho facturas. Las provincias sin facturas o sin clientes no se obtienen ya que se está haciendo un JOIN con FACTURAS y con CLIENTES. Una solución sería utilizar el JOIN EXTERNO entre FACTURAS y CLIENTES (f.codcli (+) = c.codcli) y entre CLIENTES y PUEBLOS (c.codpue (+) = pu.codpue). Otra solución sería hacer una UNION con una sentencia SELECT que obtenga las provincias sin facturas o sin clientes y las saque acompañadas de un cero indicando el número de facturas (SELECT pr.nombre, 0).

EJERCICIO 1 (2 puntos)

(a.1) Escribir una expresión del álgebra relacional que de los actores premiados, obtenga aquel que consiguió su primer premio con menor edad.

```
T1 := PREMIO JOIN ACTOR
     actores y premios que han ganado
T2 := (EXTEND T1 ADD año-año_nac AS edad) [codactor,edad]
     actores y edades en las que recibieron premios
T3 := SUMMARIZE T2 GROUP BY () ADD MIN(edad) AS edad
     edad mas pequeña en la que un actor recibió un premio
RDO := T2 JOIN T3 JOIN ACTOR
```

(a.2) Escribir una expresión del álgebra relacional que obtenga los datos de los socios que sólo han alquilado películas españolas realizadas antes de 1980.

```
T1 := PRESTAMO JOIN CINTA JOIN PELICULA
T2 := (T1 WHERE nacionalidad='española' AND año<1980) [codsocio]
T3 := (T1 WHERE nacionalidad<>'española' OR año>=1980) [codsocio]
RDO := (T2 MINUS T3) JOIN SOCIO
```

(b.1) Escribir una expresión del cálculo relacional que obtenga los datos de los socios que han tomado prestadas todas las películas realizadas el año pasado por directores menores de 30 años de edad (para las fechas utilizar las funciones de SQL).

```
SOCIOX WHERE ∀ PELICULAX ∃ DIRECTORX
   (IF PELICULAX.año = TO_NUMBER(TO_CHAR(SYSDATE,'YYYY'))-1
       AND
       TO_NUMBER(TO_CHAR(SYSDATE,'YYYY'))-DIRECTORX.año_nac < 30
    THEN ∃ PRESTAMOX ∃ CINTAX
         (PRESTAMOX.codsocio = SOCIOX.codsocio AND
          PRESTAMOX.codcinta = CINTAX.codcinta AND
          CINTAX.codpeli = PELICULAX.codpeli))
```

(b.2) Escribir una expresión del cálculo relacional que obtenga los datos de los socios que sólo han alquilado películas españolas realizadas antes de 1980.

```
SOCIOX WHERE ∀ PRESTAMOX
   (IF PRESTAMOX.codsocio = SOCIOX.codsocio
    THEN ∃ CINTAX ∃ PELICULAX
         (CINTAX.codcinta = PRESTAMOX.codcinta AND
          PELICULAX.codpeli = CINTAX.codpeli AND
          PELICULAX.nacionalidad = 'española' AND
          PELICULAX.año < 1980))
```

EJERCICIO 2 (2.5 puntos)

(a) ¿A qué consulta responde la siguiente sentencia SQL?

```
SELECT DISTINCT c.codcli, c.nombre
FROM   lineas_fac lf, facturas f, clientes c
WHERE  lf.codfac = f.codfac AND f.codcli = c.codcli
GROUP BY c.codcli, c.nombre, f.codfac
HAVING SUM(lf.cant*lf.precio*(1-lf.dto/100))>10000
MINUS
SELECT DISTINCT c.codcli, c.nombre
FROM   lineas_fac lf, facturas f, clientes c
WHERE  lf.codfac = f.codfac AND f.codcli = c.codcli
GROUP BY c.codcli, c.nombre, f.codfac
HAVING SUM(lf.cant*lf.precio*(1-lf.dto/100))<=10000
```

Esta sentencia obtiene un listado de los clientes que en todas y cada una de sus facturas han superado el importe de 10.000 pesetas.

(b) Escribir una sentencia SQL que responda a la siguiente consulta:

Obtener un listado de provincias (nombre y código) donde hay clientes en al menos diez de sus pueblos.

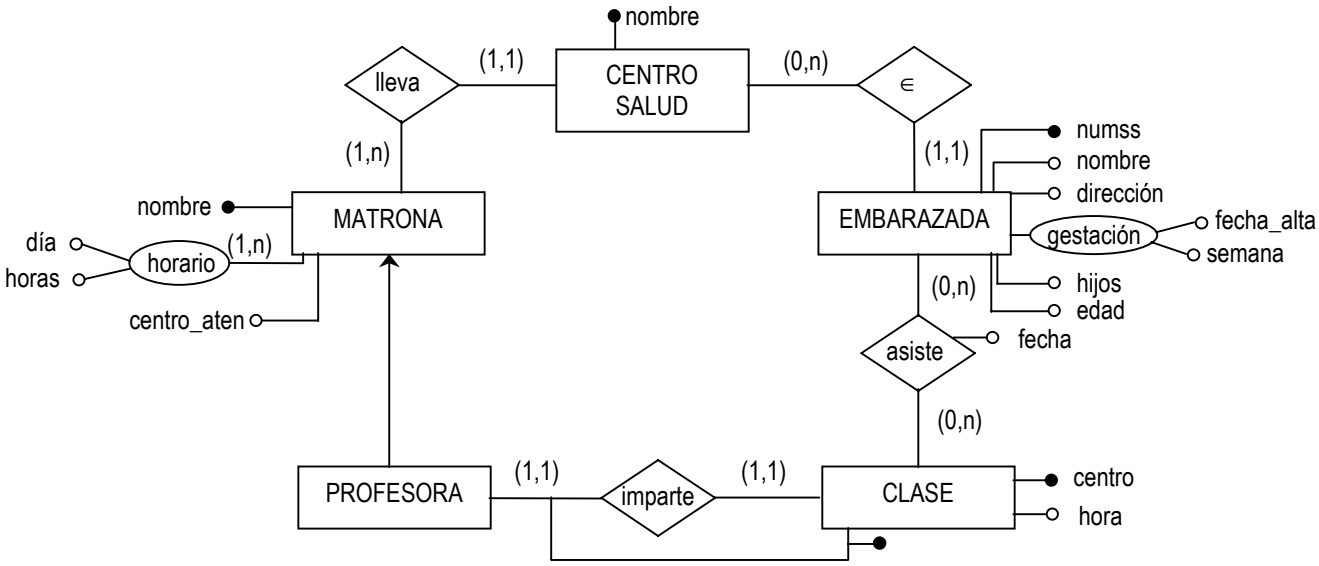
```
SELECT pr.codpro, pr.nombre
FROM   clientes c, pueblos pu, provincias pr
WHERE  c.codpue = pu.codpue AND pu.codpro = pr.codpro
GROUP BY pr.codpro, pr.nombre
HAVING COUNT(DISTINCT pu.codpue) >= 10;
```

(c) Escribir una sentencia SQL que responda a la siguiente consulta:

De cada provincia de la Comunidad Valenciana, obtener el código postal donde se encuentran más clientes.

```
SELECT pr.nombre, c.codpostal
FROM   clientes c, pueblos pu, provincias pr
WHERE  c.codpue = pu.codpue AND pu.codpro = pr.codpro
AND    UPPER(pr.nombre) IN ('CASTELLON','VALENCIA','ALICANTE')
GROUP BY pr.codpro, pr.nombre, c.codpostal
HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                         FROM   clientes c, pueblos pu
                         WHERE  c.codpue = pu.codpue
                         AND    pu.codpro = pr.codpro
                         GROUP BY c.codpostal);
```

EJERCICIO 3 (2 puntos)



El atributo compuesto **gestación** en la entidad **EMBARAZADA** lleva la fecha en que ésta se da de alta y la semana de gestación en la que se encuentra. De este modo se puede calcular la semana de gestación en cualquier momento utilizando la fecha actual y estos dos atributos.

Ya que cada matrona-profesora sólo da una clase a la semana y en un centro diferente, como identificador de la entidad **CLASE** se puede utilizar bien el nombre del centro o el nombre de la matrona.

En el esquema se ha representado que una embarazada puede asistir a clases de distintos centros. Ya que se mantiene una relación de muchos a muchos entre **EMBARAZADA** y **CLASE**, que tiene una ocurrencia por cada vez que se ha asistido a una clase (para eso se toma la **fecha**), no es necesario tener un atributo que indique el número de clases a las que se ha asistido.

Las relaciones base en 3FN que se obtienen a partir del esquema conceptual anterior son las siguientes:

MATRONA(<u>nombre</u>,centro_aten)

Esta tabla almacena el nombre de cada matrona y su centro de atención.

HORARIO_CONSULTA(<u>matrona,día</u>,horas)

HORARIO_CONSULTA.matrona es clave ajena a **MATRONA**

Esta tabla almacena el horario de consulta de cada matrona, teniendo en cuenta que puede ser diferente cada día de la semana.

PROFESORA(<u>nombre</u>,centro,hora)

PROFESORA.nombre es clave ajena a **MATRONA**

Esta tabla almacena las matronas que dan clases de preparación al parto, el centro donde lo hacen y la hora (sólo dan una clase de una hora a la semana).

CENTRO_SALUD(<u>nombre</u>,matrona)

CENTRO_SALUD.matrona es clave ajena a **MATRONA**

Esta tabla almacena el nombre de cada centro de salud y la matrona que lleva a sus embarazadas.

EMBARAZADA(<u>nss</u>,nombre,dirección,fecha_alta,semana,hijos,edad,centro_salud)

EMBARAZADA.centro_salud es clave ajena a **CENTRO_SALUD**

Esta tabla almacena los datos de cada embarazada, con su centro de salud.

ASISTENCIA(<u>nssembarazada,fecha</u>,matrona)

ASISTENCIA.nssembarazada es clave ajena **EMBARAZADA**

ASISTENCIA.matrona es clave ajena **PROFESORA**

Esta tabla almacena las clases a las que ha asistido cada embarazada. Se supone que en un mismo día no se acude a dos clases (aunque sean en centros diferentes).