FICHEROS Y BASES DE DATOS **2º ITIG** **13/7/2000**

TEST (1.75 puntos)

1. Una diferencia entre los sistemas de ficheros (SF) y los sistemas de bases de datos (SBD) es que ...

(-) ... los SF no pueden almacenar reglas de integridad, mientras que los SBD sí pueden hacerlo.

Verdadero.

(-) ... los SF pueden almacenar datos redundantes, mientras que los SBD no pueden.

No es cierto. Tanto los SF como los SBD pueden almacenar datos redundantes. Precisamente es importante realizar un buen diseño para evitar que en las bases de datos se almacenen datos redundantes.

(-) ... los SF utilizan lenguajes navegacionales, mientras que los SBD utilizan lenguajes no navegacionales.

No es cierto. Los SF se manipulan mediante programas de aplicación que pueden escribirse mediante cualquier lenguaje capaz de acceder a los ficheros. Hay SBD que tienen lenguajes navegacionales (sistemas jerárquicos y de red) y sistemas que tienen lenguajes no navegacionales (sistema relacional).

2. Un ejemplo de base de datos es ...

(-) ... las tablas S, P y SP.

Verdadero. Las tablas S, P y SP constituyen una base de datos relacional.

(-) ... Oracle.

No es cierto. *Oracle es un sistema de gestión de bases de datos. La base de datos es el conjunto de datos almacenados.*

(-) ... el modelo relacional.

No es cierto. *El modelo relacional es un modelo de datos compuesto por un conjunto de conceptos que sirven para definir la estructura de la base de datos y un conjunto de operaciones para manipularla.*

3. Algunas de las funciones de los sistemas de gestión de bases de datos son ...

(-) ... controlar la seguridad, la concurrencia y la recuperación ante fallos.

Verdadero.

(-) ... eliminar los datos redundantes y mantener las prestaciones.

No es cierto. *El SGBD no es capaz de reconocer las redundancias de datos, pero gracias a él podemos diseñar bases de datos que no las contengan. La tarea de mantener las prestaciones es función del administrador de la base de datos.*

(-) ... mejorar la productividad de los programadores.

No es cierto. Gracias a la existencia del SGBD los programadores pueden mejorar su productividad, ya que muchas tareas que antes se debían incluir en los programas de aplicación, las realizan ahora los SGBD. Es una ventaja, pero no una función.

4. Se tiene un fichero ordenado sobre el que se ha definido un índice primario sobre el campo clave de ordenación (IP) y un índice secundario sobre otro campo no clave (IS).

(-) En ambos índices se puede utilizar búsqueda binaria.

Verdadera. Ambos son índices de un solo nivel, que son ficheros ordenados y, por lo tanto, son índices sobre los que se pueden realizar búsquedas binarias.

(-) En IP sólo se puede utilizar búsqueda lineal y en IS se puede utilizar búsqueda binaria.

No es cierto.

(-) En IP se puede utilizar búsqueda binaria y en IS sólo se puede utilizar búsqueda lineal.

No es cierto.

5. Qué ventajas presentan los índices multinivel frente a los índices de un solo nivel?

(-) Que se reduce el número de accesos al hacer búsquedas.

Verdadero.

(-) Ninguna, cuantos más niveles, más búsquedas hay que realizar.

No es cierto, los índices multinivel sí presentan ventajas.

(-) Que en cada nivel las entradas del índice son más pequeñas, por lo que ocupa menos espacio.

No es cierto, ya que en un índice multinivel todas las entradas en todos los niveles son del mismo tamaño.

6. ¿Qué es la dispersión?

(-) Un modo de organizar los registros dentro de un fichero.

Verdadero.

(-) Un directorio que se puede añadir a los ficheros en cualquier momento a modo de índice.

No es cierto, cuando se construye el fichero disperso, el directorio, si lo hay (la dispersión lineal no lo tiene) se construye a la vez, no se añade después.

(-) Una estructura de datos que funciona bien para memoria principal, pero no para disco.

No es cierto, la dispersión funciona bien tanto para memoria principal, como para ficheros en disco.

7. Ya que los nodos internos de un árbol B+ no guardan punteros a los registros de datos ...

(-) ... se puede tener menos niveles que en un árbol B equivalente y, por lo tanto, llegar más rápido a los datos.

Verdadero. Es una de las ventajas importantes de los árboles B+ frente a los árboles B.

(-) ... el acceso a los datos es más lento que en un árbol B equivalente porque siempre hay que llegar a las hojas para encontrar el puntero a los datos.

No es cierto, por término medio, el acceso es más rápido por tener menos niveles.

(-) ... se ocupa menos espacio que un árbol B equivalente, pero el acceso a los datos es igual de rápido.

No es cierto que el acceso sea igual de rápido.

8. Un árbol B+ al que le caben hasta p punteros a nodos internos en cada bloque ...

(-) ... puede tener hasta p-1 valores del campo de indexación en cada nodo.

Verdadero.

(-) ... puede tener hasta $\log_2 p$ valores del campo de indexación en cada nodo.

No es cierto.

(-) ... puede tener hasta p/2 valores del campo de indexación en cada nodo.

No es cierto.

9. ¿Qué es una instantánea?

(-) Una relación (tabla) que se almacena en la base de datos y que es una "foto" de parte de la base de datos en un determinado momento.

Verdadero.

(-) Una sentencia SQL que se ejecuta muy rápidamente.

Pues no, esto no es cierto.

(-) Una consulta que se almacena en la base de datos y que se ejecuta cuando es accedida por el usuario.

No es cierto, eso es una vista.

10. ¿Qué tienen en común claves primarias y claves ajenas?

(-) ... que son identificadores de ocurrencias de entidades.

Verdadero. Una clave primaria es un identificador; una clave ajena que la referencia es ese mismo identificador, situado en otra relación (tabla) para mantener una relación entre entidades.

(-) ... que pueden aceptar nulos si es necesario.

No es cierto, las claves primarias nunca aceptan nulos.

(-) ... que son todas claves candidatas de la relación (tabla) en que se encuentran.

No es cierto, las claves ajenas no tienen porqué identificar de modo único las tuplas de la relación en la que se encuentran.

11. ¿Cómo se puede conseguir que un sistema de gestión de bases de datos (SGBD) relacional haga que se respete la integridad referencial?

(-) Basta con definir los atributos que son claves ajenas y a qué tablas hacen referencia.

Verdadero. Si al crear las tablas de la base de datos se especifica qué atributos son claves ajenas y a qué tablas hacen referencia, el SGBD se encarga de respetar la integridad referencial automáticamente. Por ejemplo, no se podrá insertar una factura de un cliente que no exista o no se podrá borrar un cliente que tenga facturas (por defecto la opción suele ser restringir).

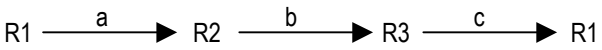
(-) Evitando la dependencia de datos (lógica-física).

No es cierto. *Esto no tiene nada que ver con la integridad referencial.*

(-) Eso no lo hacen los SGBD, hay que hacerlo desde los programas de aplicación.

No es cierto, eso sí lo hacen los SGBD. En prácticas hemos visto como lo hace Access.

12. Dado el siguiente diagrama referencial, donde ninguna clave ajena acepta nulos:



(-) Es imposible empezar a insertar filas en cualquiera de estas tablas.

Verdadero. Para poder insertar la primera fila en R1 hace falta que R2 tenga, al menos, la fila referenciada mediante la clave ajena (ya que no acepta nulos); para insertar la primera fila en R2 hace falta la fila referenciada en R3; para insertar la primera fila en R3 hace falta la fila referenciada en R1; y vuelta a empezar.

(-) Debe existir una relación redundante, porque hay un camino de R1 a R1.

No es cierto, un ciclo no implica una relación redundante.

(-) Si no aceptan nulos, la regla de borrado para todas ellas deberá ser propagar.

No es cierto, el que acepten o no nulos, no obliga a que la regla de borrado sea propagar.

13. Al realizar la operación CLIENTES JOIN PUEBLOS sobre la base de datos de prácticas, y teniendo en cuenta que ninguna clave ajena de la base de datos acepta nulos, ...

(-) ... el resultado tiene tantas filas como clientes hay en la base de datos.

Verdadero. El JOIN concatena cada cliente con los datos de su pueblo; al no aceptar nulos la clave ajena de CLIENTES a PUEBLOS, en el JOIN no se pierden clientes , por lo que se obtienen tantas filas como clientes hay en la base de datos.

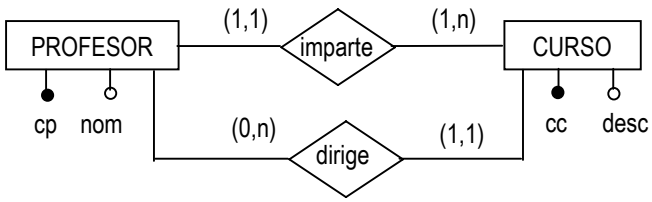
(-) ... el resultado tiene tantas filas como pueblos en donde hay clientes.

No es cierto.

(-) ... el resultado tiene tantas filas como pueblos hay en la base de datos.

No es cierto.

14. Dado el siguiente esquema conceptual ¿qué esquema lógico relacional le corresponde?



(-) PROFESOR(<u>cp</u>, cc-imparte, nom), CURSO(<u>cc</u>, cp-dirige, desc), PROFESOR.cc-imparte clave ajena a CURSO, CURSO.cp-dirige clave ajena a PROFESOR

Correcto. Cada profesor hace referencia al curso que imparte (sólo uno) y cada curso hace referencia al profesor que lo dirige (sólo uno).

(-) PROFESOR(<u>cp, cc-imparte</u>, nom), CURSO(<u>cc, cp-dirige</u>, desc), (CURSO.cc,CURSO.cp-imparte) clave ajena a PROFESOR

No es correcto, la clave primaria de cada tabla sólo la debe formar el identificador de la entidad correspondiente en el esquema.

(-) PROFESOR(<u>cp</u>, nom), CURSO(<u>cc</u>, desc), IMPARTE(<u>cp-dirige, cc-imparte</u>), IMPARTE.cp-dirige clave ajena a PROFESOR, IMPARTE.cc-imparte clave ajena a CURSO

No es correcto, las filas de IMPARTE no tienen ningún sentido.

CUESTIÓN (0.75 puntos)

Se tiene la siguiente relación MENÚ(<u>persona, clase</u>, plato) y la siguiente información:

- existen tres clases de platos (primero, segundo y postre),
- cada persona sólo come un plato de cada clase y
- cada plato sólo pertenece a una clase (por ejemplo, la sopa es un primero).

Esto último implica la siguiente dependencia funcional: plato → clase. ¿Es posible descomponer la relación MENÚ en otras relaciones, de modo que se evite esta dependencia en MENÚ y no se pierda información? Razona la respuesta.
AYUDA: Intenta hacer una descomposición coherente y razona sobre ella.

Si sacamos la dependencia funcional de la relación MENÚ, nos quedan las relaciones:

PLATO(<u>plato</u>, clase) y MENÚ(<u>persona, plato</u>) (ojo con la clave primaria de MENÚ)

donde MENU.plato es clave ajena a PLATO.

Lo que sucede es que, de este modo, se pierde información: ya no se refleja que cada persona come sólo un plato de cada clase. Por ejemplo, una persona podría comerse varios primeros, cuando no debe ser así. Por lo tanto se puede decir que MENÚ no se puede descomponer sin perder información.

CUESTIONES SQL (1 punto)

1. Se desea tener un listado de los nuevos clientes que se han conseguido este año. Para ello se ha escrito la siguiente sentencia SQL que busca los clientes que sólo tienen facturas en este año. ¿Es necesario el uso de DISTINCT? Razona la respuesta. ¿Es posible escribir una sentencia equivalente que no utilice subconsultas? Si es así, escríbela.

```
SELECT DISTINCT c.codcli, c.nombre
FROM   facturas f, clientes c
WHERE  f.codcli = c.codcli
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) =
       TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))
GROUP BY c.codcli, c.nombre
HAVING NOT EXISTS (SELECT *
                   FROM   facturas ff
                   WHERE  ff.codcli = c.codcli
                   AND    TO_NUMBER(TO_CHAR(ff.fecha,'yyyy')) <
                          TO_NUMBER(TO_CHAR(SYSDATE,'yyyy')));
```

El uso de DISTINCT no es necesario ya que se obtiene una fila por grupo, correspondiendo cada grupo a un cliente distinto. Además, la sentencia sí se puede escribir sin subconsultas: obtenemos los clientes que han hecho alguna compra este año MENOS los clientes que han hecho alguna compra en años anteriores.

```
SELECT c.codcli, c.nombre
FROM   facturas f, clientes c
WHERE  f.codcli = c.codcli
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) =
       TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))
MINUS
SELECT c.codcli, c.nombre
FROM   facturas f, clientes c
WHERE  f.codcli = c.codcli
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) <
       TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'));
```

En este caso no es necesario poner DISTINCT porque al hacer operaciones de conjuntos, los duplicados no aparecen.

2. ¿La siguiente consulta devuelve siempre el total de las ventas (sin IVA ni descuentos) por provincia? Razona la respuesta.

```
SELECT UPPER(NVL(pr.nombre,'SIN NOMBRE')), SUM(l.cant*l.precio) TOTAL_VENTAS
FROM   lineas_fac l, facturas f, clientes c, pueblos pu, provincias pr
WHERE  l.codfac = f.codfac
AND    f.codcli = c.codcli
AND    c.codpue = pu.codpue
AND    pu.codpro = pr.codpro
GROUP BY pr.codpro, NVL(pr.nombre,'SIN NOMBRE')
ORDER BY 1;
```

Las provincias en las que no existen clientes no aparecen, ya que, al hacer el join de clientes con pueblos, sólo se quedan los pueblos en donde hay clientes. Esto se puede solucionar haciendo una unión con las provincias en donde no hay clientes (el total de ventas será cero para todas ellas):

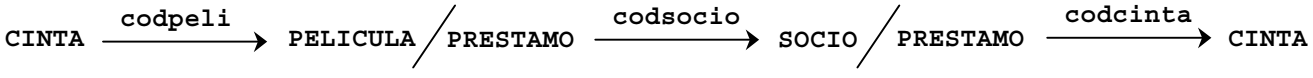
```
(...)
UNION
SELECT UPPER(NVL(pr.nombre,'SIN NOMBRE')), 0 TOTAL_VENTAS
FROM   provincias
WHERE  codpro NOT IN (SELECT pu.codpro
                      FROM   clientes c, pueblos pu
                      AND    c.codpue = pu.codpue);
```

EJERCICIO 1 (2 puntos)

Las relaciones que forman la base de datos de un vídeo club son las siguientes:

PELICULA(codpeli,título,año,género,clasificación,fecha_estreno)
CINTA(codcinta,codpeli,idioma,fecha_entrada,fecha_retiro)
SOCIO(codsocio,nombre,dirección,teléfono)
PRESTAMO(codsocio,codcinta,fecha,pres_dev)

En las relaciones anteriores, son claves primarias los atributos y grupos de atributos que aparecen subrayados. Las claves ajenas se muestran en los siguientes diagramas referenciales:



El vídeo club posee copias (**CINTA**) de películas (**PELICULA**) que presta (**PRESTAMO**) a sus socios (**SOCIO**). De las películas se guarda el código, título, año de realización, género cinematográfico, clasificación (todos los públicos, mayores de 13 años, etc.) y la fecha de estreno. Si la película se ha estrenado en España en los cines, esta es la fecha que se anota; si la película no se ha estrenado en los cines españoles, la fecha de estreno es la fecha en que la primera copia de la película llegó al vídeo club. De una misma película puede haber copias en idiomas distintos, por lo que en cada copia se indica el idioma y, también, la fecha en que llegó al vídeo club y la fecha en que se retiró (admite nulos). De los socios se guarda el código, nombre, dirección y teléfono. Por último, de los préstamos interesa la cinta que se presta, el socio al cual se presta y la fecha. Además, mientras una cinta está prestada, el atributo pres_dev tiene el valor '*prestada*', y al finalizar el préstamo su valor es '*devuelta*'.

1. Escribir una expresión del álgebra relacional que obtenga los datos de los socios que siempre toman prestadas películas que no se han estrenado en los cines españoles. (1 pto.)

```
T1 := SUMMARIZE CINTA GROUPBY (codpeli) ADD MIN(fecha_entrada) AS fecha_primera
```
Para cada película obtenemos la fecha en que llegó su primera copia al vídeo club.
```
T2 := ((PELICULA JOIN T1) WHERE fecha_estreno = fecha_primera) [codpeli]
```
Estas son las películas que no se han estrenado en los cines (la fecha del estreno es la fecha en que llegó la primera copia).
```
T3 := (PRESTAMO JOIN CINTA JOIN T2) [codsocio]
```
Estos son los socios que han tomado alguna de las películas que no se han estrenado en los cines (las de T2).
```
T4 := (PRESTAMO JOIN CINTA JOIN (PELICULA[codpeli] MINUS T2)) [codsocio]
```
Estos son los socios que han tomado alguna de las películas que se han estrenado en los cines (películas que no están en T2).
```
RDO := (T3 MINUS T4) JOIN SOCIO
```
Al hacer la diferencia quedan los socios que siempre han tomado películas que no se han estrenado en los cines.

2. ¿Qué consulta realiza la siguiente expresión del álgebra relacional? (0.5 ptos.)

```
((PELICULA JOIN CINTA) WHERE fecha_estreno = fecha_entrada) [codpeli,título]
MINUS
((PELICULA JOIN CINTA) WHERE fecha_estreno <> fecha_entrada) [codpeli,título]
```

Obtiene el código y el título de las películas que no se han estrenado en los cines españoles y cuyas copias llegaron todas el mismo día al vídeo club.

3. ¿Qué consulta realiza la siguiente expresión del cálculo relacional? Tener en cuenta que la diferencia de dos fechas obtiene el número de días que hay entre ellas. (0.5 ptos.)

```
PELICULAX WHERE ∀ SOCIOX ∃ PRESTAMOX ∃ CINTAX
    (IF SOCIOX.codsocio = PRESTAMOX.codsocio AND
        PRESTAMOX.codcinta = CINTAX.codcinta AND
        CINTAX.codpeli = PELICULAX.codpeli
     THEN
        PRESTAMOX.fecha - PELICULAX.fecha_estreno <= 30)
```

Una película sale en el resultado si todos los socios que la han tomado prestada, lo han hecho en los 30 primeros días de su estreno.

EJERCICIO 2 (2.5 puntos)

(a) ¿A qué consulta responde la siguiente sentencia SQL? Ilustra la respuesta mediante un ejemplo. (0.5 ptos.)

```
SELECT c.codcli, c.nombre, DECODE(MIN(iva), 7,' X ','') "7%",
                           DECODE(MAX(iva),16,' X ','') "16%"
FROM   facturas f, clientes c
WHERE  f.codcli = c.codcli
AND    (iva=7 OR iva=16)
GROUP BY c.codcli, c.nombre
ORDER BY 2;
```

La sentencia obtiene un listado de los clientes, ordenados por nombre, que tienen alguna factura con IVA 7% ó 16%. Para cada cliente, se indica, concretamente, qué tipos de IVA se le han aplicado: sólo 7%, sólo 16%, o ambos. Por ejemplo, si el cliente C1 sólo tiene facturas con IVA 7%, el cliente C2 sólo tiene facturas con IVA 16% y el cliente C3 tiene alguna factura con IVA 7% y alguna con IVA 16%, el resultado que se obtiene es el siguiente:

CODCLI	NOMBRE	7%	16%
C1	Gómez	X	
C2	López		X
C3	Pérez	X	X

(b) El impuesto en las facturas de clientes se aplica según el impuesto asociado a la provincia en la que se realiza la venta (la provincia del cliente). En el supuesto de que la nueva ley sobre el IVA, puesta en vigor el 1 de enero de 2000, establezca el 4% para los clientes de Ceuta y Melilla, y el 16% para el resto de provincias, se requiere un listado en donde, para cada provincia, se muestre el número de facturas a las que no se ha aplicado el impuesto que le corresponde. Sólo es preciso sacar las provincias en donde haya alguna de estas facturas. Escribir una sentencia SQL que obtenga el listado que se desea (1 pto.)

```
SELECT pr.codpro, pr.nombre, COUNT(*)
FROM   facturas f, clientes c, pueblos pu, provincias pr
WHERE  f.codcli = c.codcli
AND    c.codpue = pu.codpue
AND    pu.codpro = pr.codpro
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) = TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))
AND    f.iva <> DECODE(UPPER(pr.nombre),'CEUTA',4,'MELILLA',4,16)
GROUP BY pr.codpro, pr.nombre;
```

Otra solución (más lenta) que no utiliza DECODE (por si todavía no se sabe manejar :)

```
SELECT pr.codpro, pr.nombre, COUNT(*)
FROM   facturas f, clientes c, pueblos pu, provincias pr
WHERE  f.codcli = c.codcli
AND    c.codpue = pu.codpue
AND    pu.codpro = pr.codpro
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) = TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))
AND    UPPER(pr.nombre) IN ('CEUTA','MELILLA')
AND    f.iva <> 4
GROUP BY pr.codpro, pr.nombre
UNION
SELECT pr.codpro, pr.nombre, COUNT(*)
FROM   facturas f, clientes c, pueblos pu, provincias pr
WHERE  f.codcli = c.codcli
AND    c.codpue = pu.codpue
AND    pu.codpro = pr.codpro
AND    TO_NUMBER(TO_CHAR(f.fecha,'yyyy')) = TO_NUMBER(TO_CHAR(SYSDATE,'yyyy'))
AND    UPPER(pr.nombre) NOT IN ('CEUTA','MELILLA')
AND    f.iva <> 16
GROUP BY pr.codpro, pr.nombre
```

(c) Para proponer ofertas especiales a los buenos clientes, se necesita un listado con los datos de aquellos que, en los últimos quince meses, han hecho siempre facturas por un importe superior a 50.000 pesetas (sin tener en cuenta IVA ni descuentos). Escribir una sentencia SQL que obtenga el listado que se desea (1 pto.)

```
SELECT *
FROM   clientes
WHERE  codcli IN ( SELECT f.codcli
                   FROM   lineas_fac l, facturas f
                   WHERE  l.codfac = f.codfac
                   AND    MONTHS_BETWEEN(SYSDATE, f.fecha) <= 15
                   GROUP BY f.codcli, f.codfac
                   HAVING SUM(l.cant*l.precio) > 50000
                   MINUS
                   SELECT f.codcli
                   FROM   lineas_fac l, facturas f
                   WHERE  l.codfac = f.codfac
                   AND    MONTHS_BETWEEN(SYSDATE, f.fecha) <= 15
                   GROUP BY f.codcli, f.codfac
                   HAVING SUM(l.cant*l.precio) <= 50000);
```

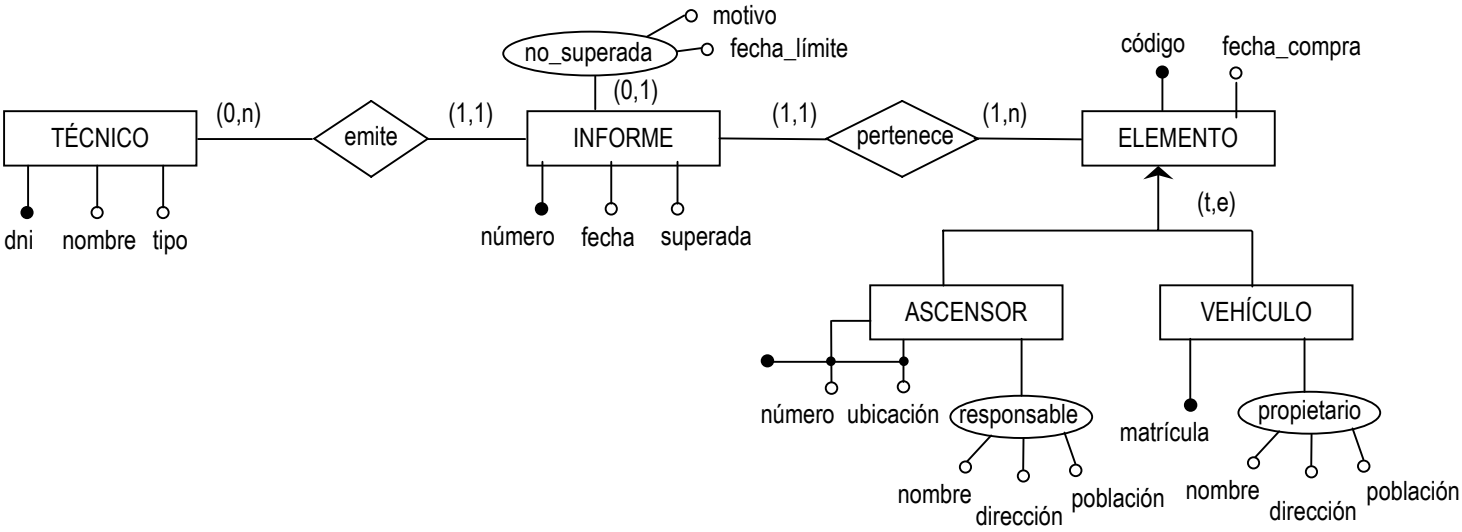
EJERCICIO 3 (2 puntos)

La Conselleria de Industria tiene un servicio que se encarga de realizar inspecciones técnicas de ascensores y vehículos. Cuando se realiza una inspección, el técnico que la lleva a cabo emite un informe en el que se especifica si se ha superado la inspección y, en caso contrario, se especifica el motivo por el que no la ha superado y la fecha límite para repetirla. Cuando se repite la inspección, se elabora un nuevo informe.

Se quiere construir una aplicación que, a partir de los informes de los técnicos, genere avisos automáticamente para los propietarios o responsables de los elementos que no han superado la inspección. Estos avisos se generan diez días hábiles antes de la fecha límite y tres días hábiles después, todo esto en caso de que no se haya repetido la inspección.

De los técnicos se quiere conocer el DNI, nombre y tipo (de ascensores o de vehículos). De los informes se quiere conocer el número, la fecha de elaboración y el técnico que lo ha elaborado, si se ha superado o no la inspección y, en este último caso, el motivo y la fecha límite para repetirla. Los elementos que se inspeccionan son, actualmente, ascensores y vehículos, y a todos ellos se les asigna un código que los identifica e interesa su fecha de compra. Los ascensores también se identifican por su número y ubicación (por ejemplo, el ascensor 13 del campus de Riu Sec de la UJI) y tienen un responsable que es quien debe recibir los avisos. Los vehículos también se identifican por su matrícula y tienen un propietario al que se envían los avisos. Tal y como se puede suponer, una misma persona puede ser responsable de varios ascensores (por ejemplo, una misma persona de la oficina técnica de la UJI puede ser la responsable de todos los ascensores del campus) y una misma persona puede ser propietaria de varios vehículos.

El esquema conceptual que se ha obtenido a partir de estos requerimientos es el siguiente:



1. Se pide diseñar un conjunto de relaciones en tercera forma normal que permitan almacenar toda la información descrita. En las relaciones (tablas) hay que señalar los atributos que son clave primaria y los que son clave ajena, determinando si éstas aceptan o no nulos y su comportamiento ante el borrado de tuplas en la tabla que referencian. Si se hace alguna suposición, hay que especificarla.

TÉCNICO(dni,nombre,tipo)
Datos de los técnicos que elaboran los informes.
tipo ∈ *{'de ascensores','de vehículos'}*

ELEMENTO(código,fecha_compra)
Elementos (ascensores y vehículos) a los que se ha realizado alguna inspección.

INFORME(número,fecha,superada,motivo,fecha_límite,dni_técnico,código_elemento)
Datos de los informes que elaboran los técnicos al pasar las inspecciones a los distintos elementos.
superada ∈ *{'si','no'}*
motivo *y* **fecha_límite** *aceptan nulos, ya que no toman valores cuando se supera la inspección.*
dni_técnico *es clave ajena a* **TÉCNICO** *(técnico que ha elaborado el informe):*
no acepta nulos y no se puede borrar un técnico que ha elaborado algún informe (restringir).
código_elemento *es clave ajena a* **ELEMENTO** *(elemento que se ha inspeccionado y al que corresponde el informe): no acepta nulos y si se borra un elemento, se borran sus informes (propagar).*

ASCENSOR(código,número,ubicación,nombre_responsable)
Datos de los ascensores a los que se ha hecho alguna inspección.
código *es clave ajena a* **ELEMENTO** *(un ascensor es un elemento que tiene informes):*
no acepta nulos y no se puede borrar un elemento que es un ascensor (restringir).
nombre_responsable *es clave ajena a* **RESPONSABLE** *(persona que recibe los avisos):*
no acepta nulos y no se puede borrar un responsable de un ascensor (restringir).
(número,ubicación) *es una clave alternativa.*

RESPONSABLE(nombre,dirección,población)

Datos de los responsables de ascensores. Una misma persona puede ser responsable de varios ascensores.

VEHÍCULO(código,matrícula,nombre_propietario)

código *es clave ajena a* **ELEMENTO** *(un vehículo es un elemento que tiene informes):*

no acepta nulos y no se puede borrar un elemento que es un vehículo (restringir).

nombre_ propietario *es clave ajena a* **PROPIETARIO** *(persona que recibe los avisos):*

no acepta nulos y no se puede borrar un propietario de un vehículo (restringir).

matrícula *es una clave alternativa.*

PROPIETARIO(nombre,dirección,población)

Datos de los propietarios de vehículos. Una misma persona puede se propietaria de varios vehículos.

Aunque responsables y propietarios pueden ser las mismas personas, no podemos suponer que haya datos redundantes por el hecho de estar en las dos tablas. Incluso tiene sentido que en tablas distintas, las mismas personas tengan direcciones diferentes. Por ejemplo, la persona responsable de los ascensores de la UJI tendrá una dirección de la UJI en RESPONSABLE y tendrá su dirección particular en PROPIETARIO, en el caso de las inspecciones de su propio vehículo.

2. Cada vez que se genera un aviso, debe quedar constancia de ello en la base de datos registrando la fecha del mismo. ¿Dónde deberían situarse las fechas de los avisos en la base de datos de modo que ésta sea completamente flexible frente a los posibles cambios en el número de avisos a enviar?

Habrá una relación (tabla) de avisos en donde para cada informe se puedan guardar todos los avisos realizados con su fecha de emisión:

AVISO(número,fecha)

número *es clave ajena a* **INFORME** *(informe al que corresponde el aviso):*

no acepta nulos y si se borra un informe, se borran todos sus avisos (propagar).