**Ficheros y Bases de Datos**
**Curso 2008-9**
**Ingeniería Técnica de Informática de Gestión**
**Primer Parcial. 11-Sept-2009.**

Nombre:
Grupo:

Se debe entregar esta hoja

1) (3,5 puntos) Se tiene un modelo relacional para un sistema de información con información de trabajadores y proyectos de una empresa de servicios
   Empleados(DNI, nombre, apellido, fechaNac*, salario, ciudad)
   Asignaciones(DNIEmpl, Proyecto, DNISupervisor, horas )
   Proyectos(nombre, empresa, ciudad)

Realizar las siguientes consultas en SQL:

a) (0,75 puntos) Mostrar el nombre y apellidos de los empleados que tienen algún proyecto en su ciudad. No debe haber duplicados. La salida debe estar ordenado por apellido, y en caso de mismo apellido, por nombre.

```
SELECT DISTINCT E.NOMBRE,APELLIDO
FROM EMPLEADOS E, ASIGNACIONES, PROYECTOS P
WHERE DNI = DNIEMPL
  AND PROYECTO = P.NOMBRE
  AND E.CIUDAD = P.CIUDAD
ORDER BY APELLIDO, NOMBRE
```

b) (0,75 puntos) Mostrar el nombre y apellidos de los empleados que ganan más que algún supervisor suyo que viva en su ciudad.

```
SELECT DISTINCT E.NOMBRE,E.APELLIDO
FROM EMPLEADOS E, ASIGNACIONES, EMPLEADOS S
WHERE E.DNI = DNIEMPL
  AND DNISUPERVSOR = S.DNI
  AND E.CIUDAD = S.CIUDAD
   AND E.SALARIO > S.SALARIO
```

c) (0,75 puntos) Listar el nombre y apellido de todos los empleados cuyo nombre empieza por 'A' con el número de proyectos asignados (que puede ser cero) y suma total de horas asignadas a proyectos.

```
SELECT NOMBRE, APELLIDO, COUNT(*), SUM(HORAS)
FROM EMPLEADOS LEFT OUTER JOIN ASIGNACIONES
ON DNI = DNIEMPL
WHERE NOMBRE LIKE 'A%'
GROUP BY DNI, NOMBRE, APELLIDO
```

d) (0,75 puntos) Mostrar las ciudades que sólo tienen un proyecto y ciudades de empleados con fecha de nacimiento desconocida

```
SELECT CIUDAD
FROM PROYECTO P
WHERE NOT EXISTS (SELECT *
            FROM PROYECTO
            WHERE NOMBRE < > P.NOMBRE
            AND CIUDAD = P.CIUDAD
             )
UNION
SELECT CIUDAD
FROM EMPLEADOS
WHERE FECHANAC IS NULL
```

```
Otra version:
SELECT CIUDAD
FROM PROYECTO P
GROUP BY CIUDAD
HAVING COUNT(*) = 1
UNION
SELECT CIUDAD
FROM EMPLEADOS
WHERE FECHANAC IS NULL
```

e) (0,5 puntos) Subir un 10% el salario de los empleados que tienen asignados más de dos proyectos

```
UPDATE EMPLEADOS
SET SALARIO = SALARIO * 1.1
WHERE DNI IN (SELECT DNIEMPL
                  FROM ASIGNACIONES
                  GROUP BY DNIEMPL
                  HAVING COUNT(*)> 2
                  )
```

2) (2 puntos) Dependencias funcionales y Normalización.
En la relación de asignaciones del ejercicio anterior se imponen dos reglas de negocio:

- Un empleado asignado a un proyecto sólo puede tener un supervisor (en cada proyecto) y un número de horas dedicadas a cada proyecto.
- Un empleado sólo puede ser supervisor de un proyecto (de empleados en un solo proyecto).

a) (0,5 puntos) Forzar dichas restricciones dando una lista de dependencias funcionales (una Dependencia funcional por cada regla de negocio)

**F = { {DNIEmpl, Proyecto}→{DNISupervisor, horas} DNISupervisor→ Proyecto }**

b) (0,5 punto) Teniendo en cuenta las dependencias funcionales del apartado a) ¿está Asignaciones en 3FN?

**Si, pues para toda dependencia funcional de F, el antecedente es superclave, o el consecuente forma parte de una clave candidata**

c) (0,5 punto) Descomponer Asignaciones para obtener un conjunto de esquemas en FNBC.

```
R1(DNISupervisor,Proyecto)
R2(DNIEmpl, DNISupervisor, horas) ;
```

d) (0,5 punto) Subrayar la clave candidata de la descomposición en FNBC

**R1(<u>DNISupervisor</u>,Proyecto)**
**R2(<u>DNIEmpl, DNISupervisor</u>, horas);**

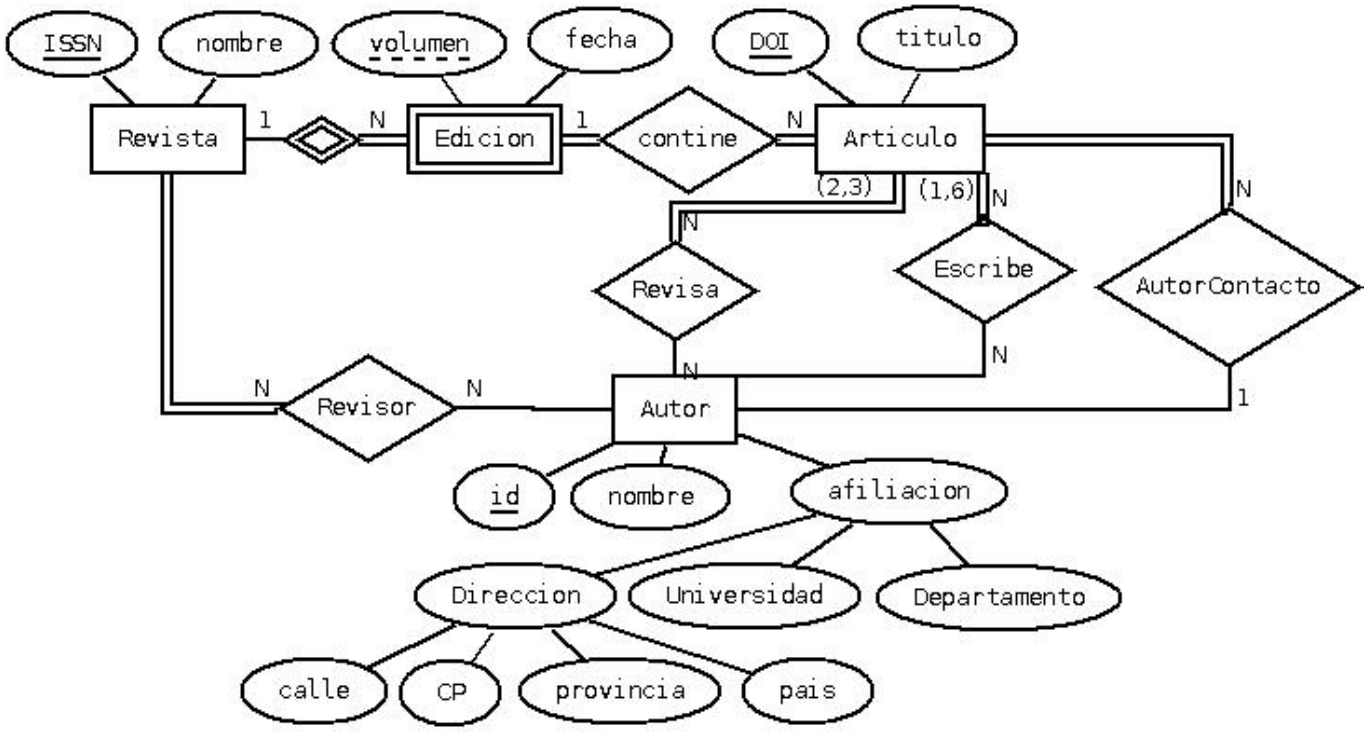
3) (2,5 puntos) Construye el esquema entidad / relación para la base de datos que se describe a continuación, incluyendo atributos clave y restricciones de cardinalidad y participación:

Un portal web gestiona la información relativa a las publicaciones científicas en revistas de múltiples autores. La base de datos debe guardar la siguiente información:

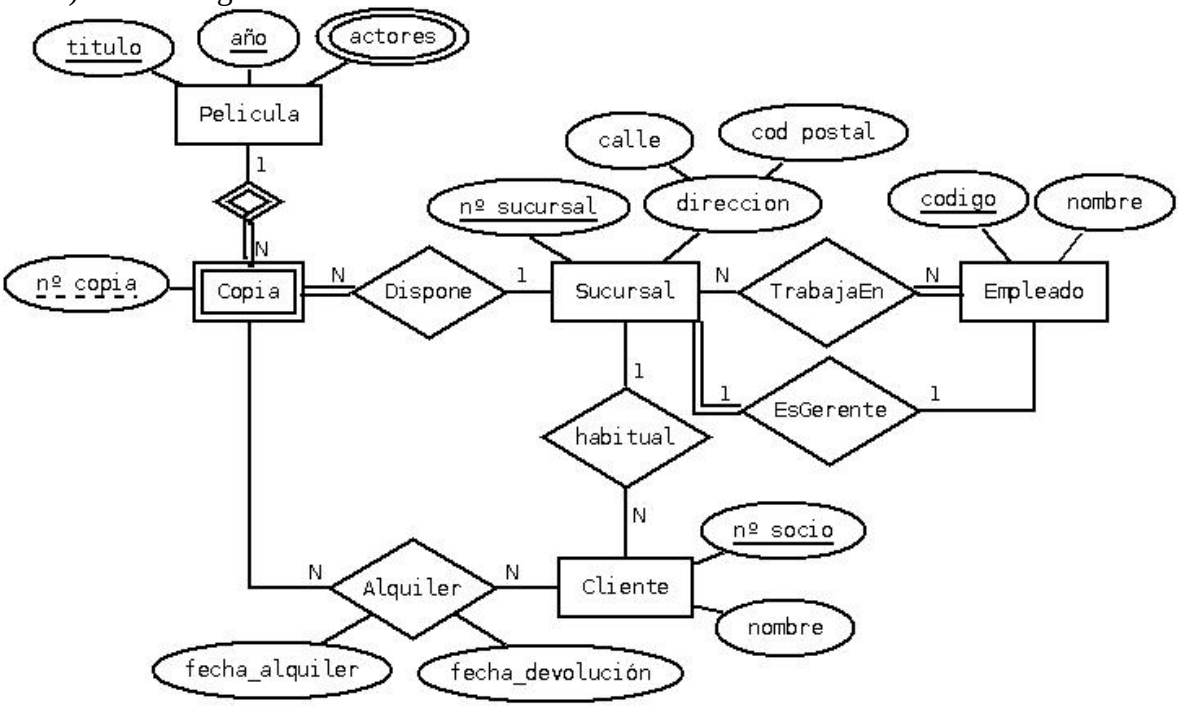
- Para cada revista se almacena su nombre y su ISSN (*International Standard Serial Number*) que la identifica unívocamente.
- Cada revista publica varias ediciones a lo largo del año. Para cada edición hay que almacenar la fecha y un número de volumen que identifica la edición en la revista. Una revista tiene 1 o varios revisores que serán los encargados de revisar los artículos. Un autor puede ser revisor de varias revistas.
- Para cada autor es necesario almacenar su nombre, afiliación y un identificador interno de la aplicación que permite distinguir a los autores en la BBDD. La afiliación consiste en el nombre de la universidad, nombre del departamento y dirección. La dirección consta de la calle, código postal, provincia y país.
- Para cada artículo es necesario almacenar su título y su DOI (*Digital Object Identifier*) que identifica de manera unívoca cada artículo. Un artículo está escrito por uno o varios autores. Cada artículo tiene asociado un autor de contacto. Finalmente, un artículo tiene asociados 2 o 3 revisores.

**Solución:**

**Si no se incluyen los atributos compuestos afiliación y dirección, pero se incluyen los atributos individuales no se considera fallo.**



4) (2 puntos) Dado el siguiente modelo E/R




a) (1 puntos) Pásalo al modelo relacional lo más eficientemente posible. Marca con un asterisco aquellas columnas que pueden tomar valores nulos.

**Película(título, ano)**
**Actores(titulo, ano, actor)**
**Copia(nCopia, titulo, año, sucursal)**
**Sucursal(nSucursal, calle, cp, gerente)**
**Empleado(codigo, nombre)**
**Cliente(nSocio, nombre, habitual*)**
**Alquiler(nCopia, titulo, ano, cliente, fecha_alquiler, fecha_devolucion)**
**TrabajaEn(empleado, sucursal)**

b) (0,5 puntos) Incluir restricciones de integridad referencial (puede hacerse sobre la respuesta del apartado a).

$\Pi_{(titulo,ano)}(Actores) \subseteq \Pi_{(titulo,ano)}(Pelicula)$

$\Pi_{(titulo,ano)}(Copia) \subseteq \Pi_{(titulo,ano)}(Pelicula)$

$\Pi_{(sucursal)}(Copia) \subseteq \Pi_{(nSucursal)}(Sucursal)$

$\Pi_{(gerente)}(Sucursal) \subseteq \Pi_{(codigo)}(Empleado)$

$\Pi_{(habitual)}(Cliente) \subseteq \Pi_{(nSucursal)}(Sucursal)$

$\Pi_{(nCopia,titulo,ano)}(Alquiler) \subseteq \Pi_{(nCopia,titulo,ano)}(Copia)$

$\Pi_{(cliente)}(Alquiler) \subseteq \Pi_{(nSocio)}(Cliente)$

$\Pi_{(empleado)}(TrabajaEn) \subseteq \Pi_{(codigo)}(Empleado)$

$\Pi_{(sucursal)}(TrabajaEn) \subseteq \Pi_{(nSucursal)}(Sucursal)$

c) (0,5 puntos) El modelo relacional resultante de la transformación, ¿fuerza que se cumplan todas las restricciones que aparecen en el modelo E/R?.

**No, la participación total de "Empleado" en "TrabajaEn" no se puede forzar en el modelo relacional resultante, ya que la relación está representada por una tabla aparte. Para forzar que se cumpla la relación, sería necesario añadir una restricción de tipo CHECK durante la creación de "TrabajaEn" o un disparador que se encargara de comprobar de la existencia de una fila en "TrabajaEn" para todo código de empleado en "Empleado".**

# Examen de Ficheros y bases de datos (cód. 520)
# Ingeniería Técnica en Informática de Gestión
# Convocatoria final de septiembre
# II Parcial

1) (3 puntos) Se desea indexar la tabla r(A: Integer, B: Integer, C: Integer, D: Integer) de 10.000.000 filas bajo el atributo A mediante una estructura de índice primario denso y asignación enlazada. En la tabla se pueden encontrar para este atributo los valores del 1 al 64K (65.536), con una distribución uniforme de estos valores. Los números enteros ocupan 8 bytes. Los bloques tienen un tamaño de 512 bytes, sus direcciones ocupan 4 bytes y tienen mapas de bits de existencia para los registros. Se pide:
   a) (0,8 puntos) Describir la estructura y tamaño de los ficheros de índice y de datos. Para calcular el tamaño considérese sólo lo ocupado por los registros de datos, sin considerar direcciones de enlace y mapas de bits de existencia.

**Fichero de índice:**
**Registro:**
  **Clave: 8 bytes.**
  **Dirección: 4 bytes.**
**Bloque:**
  **Mapa de bits de existencia.**
  **N registros.**
  **Tamaño: (8+4)*65.536 = 786.432 bytes = 768 KB**

**Fichero de datos:**
**Registro:**
  **A: 8 bytes.**
  **B: 8 bytes.**
  **C: 8 bytes.**
  **D: 8 bytes.**
**Bloque:**
  **Mapa de bits de existencia.**
  **M registros.**
  **Tamaño: (8*4)*10.000.000 = 320.000.000 bytes = 312.500 KB = 305 MB**

   b) (1,1 puntos) Calcular el factor de bloqueo de los ficheros de índice y de datos.

**Para el fichero de índice:**
**N*(8+4)*8+4*8+N ≤ 512*8, donde:**
- **N*(8+4)*8=N*96 es el tamaño en bits ocupado por un registro (8 bytes para el campo clave A y 4 bytes para la dirección de bloque).**
- **4*8=32 es el número de bits ocupado por la dirección del siguiente bloque para la asignación enlazada.**
- **N son los bits necesarios para el mapa de bits de existencia**
- **512*8=4.096 es el número de bits en los 512 bytes de un bloque**

**N*(96+1)≤ 4.096-32 → N≤41,89**
**Por tanto, el factor de bloqueo N del fichero de índice es de 41 registros por bloque.**

**Para el fichero de datos:**
**M*(4*8)*8+4*8+M ≤ 512*8, donde:**
- **M*(4*8)*8 = M*256 es el tamaño en bits ocupado por un registro (4 campos de 8 bytes).**
- **4*8 = 32 es el número de bits ocupado por la dirección del siguiente bloque para la asignación enlazada**
- **M son los bits necesarios para el mapa de bits de existencia**
- **512*8=4.096 es el número de bits en los 1.024 bytes de un bloque**

**M*(256+1)≤ 4.096-32**
**M≤15,81**
**Por tanto, el factor de bloqueo M del fichero de datos es de 15 registros por bloque.**

   c) (1,1 puntos) Usando el índice, ¿cuántos accesos serían necesarios en media para localizar un registro en concreto según el par de valores para el atributo A y el C suponiendo una distribución uniforme de datos

para este último campo? ¿Y sin usarlo? Calcular la ganancia de velocidad utilizando índice respecto a no utilizarlo.

**Usando el fichero de índice:**
**64K/2 valores por recorrer secuencialmente, que ocupan** $\lceil \lceil 64K/2 \rceil / 41 \rceil = 800$ **bloques. Una vez localizado el primer registro en el fichero de datos con el valor de A buscado, se procede a recorrer secuencialmente en este fichero la mitad de los registros para ese valor de A. Para cada valor de A hay** $\lceil 10.000.000/64K \rceil = 153$ **registros. Por lo tanto, hay que leer** $\lceil \lceil 153/2 \rceil / 15 \rceil = 6$ **bloques. En total: 806 accesos de lectura.**

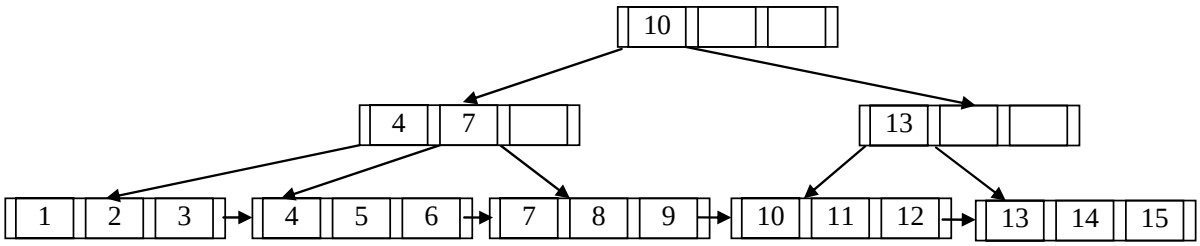
**Sin usar el fichero de índice:**
**Hay que recorrer secuencialmente la mitad de los registros del fichero de datos (10.000.000/2=5.000.000), que ocupan** $\lceil 5.000.000/15 \rceil = 333.334$ **bloques. Es decir 333.334 accesos de lectura.**

**Ganancia de velocidad:**
**Se calcula simplemente dividiendo los dos datos calculados para el número de accesos: 333.334/806 = 413,565.**

2) (2 puntos) A partir de los quince valores clave 1, 2, 3, ..., 15 y con *n*=4 (número máximo de hijos por nodo) se pide:
   a) (1 punto) Crear un árbol B+ con estos valores, indicando las restricciones que se deben cumplir, y con el mínimo número de nodos posible.

**Solución:**
**Las restricciones que debe cumplir el árbol son:**
- **El nodo raíz tiene entre 1 y 4 hijos**
- **Cada nodo interno entre** $\left\lceil \frac{4}{2} \right\rceil = 2$ **y 4 hijos**
- **Los nodos hoja contienen entre** $\left\lceil \frac{4-1}{2} \right\rceil = 2$ **y 3 valores**

**Como *n*=4, el número máximo de valores por nodo es 3. Por tanto, para alojar los 15 valores en los nodos hoja son necesarios** $\left\lceil \frac{15}{3} \right\rceil = 5$ **nodos. Como cada nodo puede tener hasta 4 hijos, no es posible direccionarlos a todos desde el nodo raíz. Así, es necesario incluir un nivel intermedio con 2 nodos que direccionen respectivamente a, por ejemplo, 3 y 2 nodos hoja. El árbol queda:**



b) (1 punto) Si se insertasen los valores del 16 al 49 en el árbol del apartado a) (no dibujar el árbol resultante), ¿cuánto crecería el árbol en número de nodos? ¿Se ha empeorado el tiempo de acceso?

**Es necesario añadir 49-16+1=34 valores al árbol y por tanto $\lceil 34/3 \rceil = 12$ nodos hoja adicionales. Al añadir 17 nodos hoja son necesarios como mínimo $\lceil (12+5)/4 \rceil = 5$ nodos internos y, por tanto, es necesario añadir un nivel adicional ya que la raíz sólo puede tener 4 hijos. Por tanto, El número de nodos ha aumentado de 8 a (1+2+5+(12+5))=25.**

**El tiempo de acceso empeora porque se ha aumentado en una unidad el número de niveles a recorrer para acceder a un nodo hoja.**

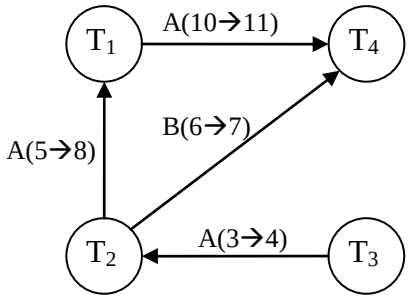
3) (1,5 puntos) Dada la siguiente planificación:

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| 1 | | LOCK(B) | | |
| 2 | | | LOCK(A) | |
| 3 | | | UNLOCK(A) | |
| 4 | | LOCK(A) | | |
| 5 | | UNLOCK(A) | | |
| 6 | | UNLOCK(B) | | |
| 7 | | | | LOCK(B) |
| 8 | LOCK(A) | | | |
| 9 | LOCK(C) | | | |
| 10 | UNLOCK(A) | | | |
| 11 | | | | LOCK(A) |
| 12 | | | | UNLOCK(B) |
| 13 | | | | UNLOCK(A) |
| 14 | UNLOCK(C) | | | |

a) (0,75 puntos) Decidir si es secuenciable y, si lo es, determínese una planificación secuencial equivalente.

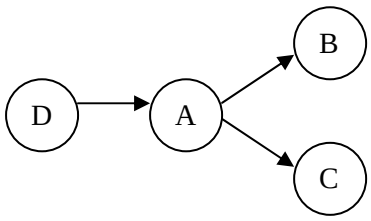
**Solución:**

**Las transacciones siguen el protocolo en 2 fases por lo que la planificación es secuenciable. En cualquier caso, ya que se pide una planificación secuencial equivalente es necesario aplicar el test de secuencialidad para crear el grafo de transacciones, obteniendo el siguiente grafo:**




**La planificación secuencial equivalente es T3→T2→T1→T4.**

b) (0,75 puntos) Dado el siguiente grafo para una base de datos decidir cuáles de las anteriores transacciones siguen el protocolo en árbol.




**Solución:**
**T1 y T3 son válidas siguiendo el protocolo basado en árbol. Sin embargo, T2 y T4 no son válidas. T2 y T4 bloquean B y después A, pero debería ser A y después B.**

4) (3,5 puntos) Dadas las siguientes tablas y restricciones:

Película(título, fecha_estreno)
Sucursal(nSucursal, calle, cp)
Copia(nCopia, titulo, ano, sucursal, alquiler*)
Cliente(nSocio, nombre, habitual*)
Alquiler(id, fecha_alquiler, fecha_devolucion, cliente, sucursal)

$\Pi_{(titulo,ano)}(Copia) \subseteq \Pi_{(titulo,ano)}(Pelicula)$

$\Pi_{(sucursal)}(Copia) \subseteq \Pi_{(nSucursal)}(Sucursal)$

$\Pi_{(alquiler)}(Copia) \subseteq \Pi_{(id)}(Alquiler)$

$\Pi_{(habitual)}(Cliente) \subseteq \Pi_{(nSucursal)}(Sucursal)$

$\Pi_{(cliente)}(Alquiler) \subseteq \Pi_{(nSocio)}(Cliente)$

$\Pi_{(sucursal)}(Alquiler) \subseteq \Pi_{(nSucursal)}(Sucursal)$

Nota: El '*' significa que la columna puede tomar un valor nulo.

Se pide:

a) (1 punto) Escribir las sentencias DDL para crear todas las tablas teniendo en cuenta lo siguiente:
- Para las fechas utiliza el tipo DATE de Oracle. Para el resto de columnas utiliza el tipo de datos que consideres oportuno.
- Las sentencias DDL deben incluir al menos las sentencias de gestión de integridad referencial.
- Incluir las restricciones adicionales que se consideren oportunas.

```
DROP TABLE Copia; DROP TABLE Alquiler; DROP TABLE Cliente;
DROP TABLE Sucursal; DROP TABLE "Película";

CREATE TABLE "Película" (
  "título" VARCHAR(20),
  fecha_estreno DATE,
  PRIMARY KEY ("título", fecha_estreno) );
CREATE TABLE Sucursal (
  nSucursal NUMBER(5) PRIMARY KEY,
  calle VARCHAR(20) NOT NULL,
  cp NUMBER(5) NOT NULL);
CREATE TABLE Cliente (
  nSocio NUMBER(6) PRIMARY KEY,
  nombre VARCHAR(20),
  habitual NUMBER(5) REFERENCES Sucursal(nSucursal) ON SET NULL);
CREATE TABLE Alquiler (
  id NUMBER(8) PRIMARY KEY,
  fecha_alquiler DATE NOT NULL,
  fecha_devolucion DATE NOT NULL,
  cliente NUMBER(6) REFERENCES Cliente(nSocio) ON DELETE CASCADE,
  sucursal NUMBER(5) REFERENCES Sucursal(nSucursal) ON DELETE CASCADE,
  CHECK ( fecha_devolucion > fecha_alquiler ) );
CREATE TABLE Copia (
  "título" VARCHAR(20),
  fecha_estreno DATE,
```

```
  nCopia NUMBER(3),
  sucursal NUMBER(5) NOT NULL REFERENCES Sucursal(nSucursal) ON DELETE CASCADE,
  alquiler NUMBER(8) REFERENCES Alquiler(id) ON DELETE SET NULL,
  FOREIGN KEY ("título", fecha_estreno) REFERENCES "Película"("título", fecha_estreno),
  PRIMARY KEY ("título", fecha_estreno, nCopia) );

INSERT INTO "Película" VALUES ( 'Peli1', DATE '2009-09-02');
INSERT INTO "Película" VALUES ( 'Peli2', DATE '2009-09-02');
INSERT INTO "Película" VALUES ( 'Peli3', DATE '2009-09-02');

INSERT INTO Sucursal VALUES ( 1, 'Calle1', 28001);
INSERT INTO Sucursal VALUES ( 2, 'Calle2', 28002);

INSERT INTO Copia VALUES ('Peli1', DATE '2009-09-02', 1, 1, NULL);
INSERT INTO Copia VALUES ('Peli1', DATE '2009-09-02', 2, 1, NULL);
INSERT INTO Copia VALUES ('Peli1', DATE '2009-09-02', 3, 1, NULL);
INSERT INTO Copia VALUES ('Peli1', DATE '2009-09-02', 4, 2, NULL);
INSERT INTO Copia VALUES ('Peli1', DATE '2009-09-02', 5, 2, NULL);
INSERT INTO Copia VALUES ('Peli2', DATE '2009-09-02', 1, 1, NULL);
INSERT INTO Copia VALUES ('Peli2', DATE '2009-09-02', 2, 1, NULL);
INSERT INTO Copia VALUES ('Peli3', DATE '2009-09-02', 1, 2, NULL);
INSERT INTO Copia VALUES ('Peli3', DATE '2009-09-02', 2, 2, NULL);

INSERT INTO Cliente VALUES ( 1, 'Cliente1', NULL);
INSERT INTO Cliente VALUES ( 2, 'Cliente2', NULL);
```

b) (1,25 puntos) Escríbase el código de un procedimiento almacenado que actualice la columna habitual de la tabla Cliente. Un cliente tiene como sucursal habitual aquélla en la que ha realizado más alquileres.

```
CREATE OR REPLACE PROCEDURE ACTUALIZA_HABITUAL AS
  CURSOR clientes IS
    SELECT NSOCIO
    FROM CLIENTE;

  CURSOR suc(cl NUMBER) IS
    SELECT NSUCURSAL
    FROM SUCURSAL
    WHERE
      NSUCURSAL IN ( SELECT ALQ.SUCURSAL
                     FROM ALQUILER ALQ
                     WHERE
                       ALQ.CLIENTE = cl
                     GROUP BY ALQ.SUCURSAL
                     HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                                              FROM ALQUILER ALQ2
                                              WHERE
                                                ALQ2.CLIENTE = cl
                                              GROUP BY ALQ2.SUCURSAL )
                                );
  sucursal_habitual NUMBER(5);
BEGIN
  FOR cl IN clientes LOOP
    OPEN suc(cl.NSOCIO);
    FETCH suc INTO sucursal_habitual;
    IF suc%FOUND THEN
      UPDATE CLIENTE SET habitual = sucursal_habitual WHERE nsocio = cl.NSOCIO;
    END IF;
    CLOSE suc;
  END LOOP;
END;
/
```

c) (1,25 puntos) Escríbase el código de un disparador en la tabla Copia que se encargue de comprobar que todas las copias que se prestan en un alquiler pertenecen a la sucursal donde se realiza el alquiler. Si la copia no pertenece a la sucursal del alquiler es necesario abortar la transacción activa.

```
CREATE OR REPLACE TRIGGER COMPRUEBA_SUCURSAL BEFORE INSERT OR UPDATE ON Copia
FOR EACH ROW
DECLARE
  sucursal_alquiler NUMBER(6);
BEGIN
  IF NOT :NEW.ALQUILER IS NULL THEN
    SELECT SUCURSAL INTO sucursal_alquiler
```

```
    FROM ALQUILER ALQ
    WHERE
      ALQ.id = :NEW.ALQUILER;
    IF sucursal_alquiler <> :NEW.SUCURSAL THEN
      RAISE_APPLICATION_ERROR(-20101, 'La copia no pertenece a la sucursal');
    END IF;
  END IF;
END;


DELETE FROM ALQUILER;

INSERT INTO ALQUILER VALUES (1, TO_DATE( '2009-09-02 13:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_DATE( '2009-09-05 13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1, 1);
UPDATE COPIA SET ALQUILER = 1 WHERE "título"='Peli1' AND fecha_estreno = DATE '2009-09-02'
AND ncopia = 1;
-- La siguiente instruccion falla ya que la copia ('Peli3', DATE '2009-09-02', 1) pertenece
a la sucursal 2
UPDATE COPIA SET ALQUILER = 1 WHERE "título"='Peli3' AND fecha_estreno = DATE '2009-09-02'
AND ncopia = 1;
INSERT INTO ALQUILER VALUES (2, TO_DATE( '2009-09-02 16:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_DATE( '2009-09-05 16:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1, 1);
UPDATE COPIA SET ALQUILER = 2 WHERE "título"='Peli2' AND fecha_estreno = DATE '2009-09-02'
AND ncopia = 1;

SELECT * FROM ALQUILER;
SELECT * FROM COPIA;
```