Universidad Politécnica de Madrid · ETSI SISTEMAS INFORMÁTICOS	Apellidos:		
	Nombre:		
	DNI:	Num. mat.:	Grupo:

Normativa de examen

- No está permitido el uso de dispositivos móviles ni otros dispositivos electrónicos, así como libros ni apuntes.
- Durante el examen, los profesores podrán solicitar acreditar la identidad de los participantes en el mismo. Deberá tener en todo momento su Documento Nacional de Identidad y/o Carné de la UPM visible sobre la mesa.
- Deberá escribir su nombre, con bolígrafo, en todas las hojas de las que consta el examen.
- No se permite abandonar el aula de examen durante los primeros 15 minutos. Transcurrido este tiempo, no se permitirá entrar al examen.
- Este examen consta de 4 ejercicios para un total de 10 puntos.
- El examen tiene una duración máxima de **2.5 horas**.
- Justifique sus respuestas lo mejor posible indicando, si fuese necesario, los pasos realizados.
- Las calificaciones provisionales serán publicadas en el Moodle de la asignatura el día 19 de Julio de 2021.
- La fecha para la revisión del examen se anunciará en el Moodle de la asignatura.

1. (2 Puntos) **Modelado**

   El gobierno de un país quiere gestionar la vacunación contra la COVID-19, desde el control de los pedidos hechos a las distintas farmacéuticas a la información referente a las personas vacunadas o en proceso de vacunación. Para ello realizaremos un modelo conceptual de datos mediante la técnica del modelo Entidad-Relación de Chen teniendo en cuenta la siguiente descripción:

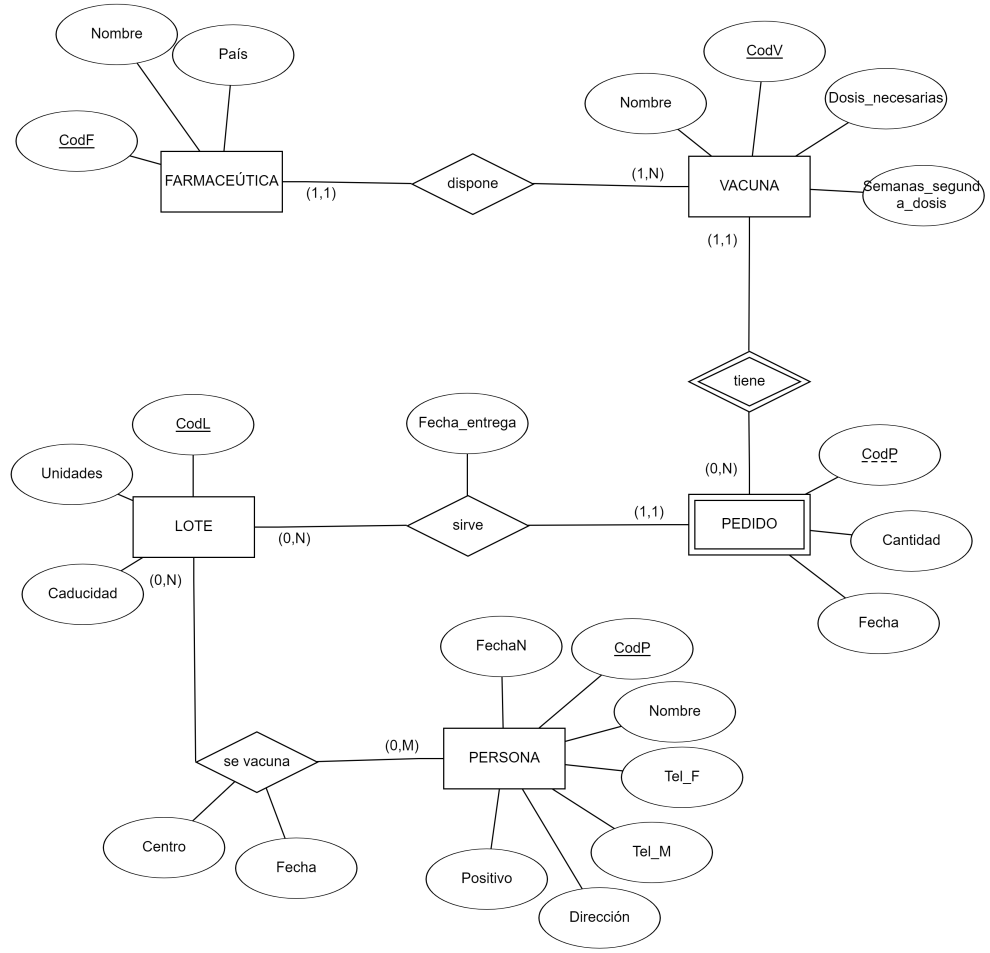
   > Las farmacéuticas, de las que se almacena un código único, su nombre y el país al que pertenecen, fabrican una serie de vacunas de las que se almacena un código de vacuna, su nombre y el número de dosis que hay que administrar para alcanzar la máxima protección y el número de semanas que hay que esperar entre dosis. Debido a la no liberación de patentes, una vacuna sólo puede ser fabricada por una farmacéutica.
   >
   > El gobierno realiza pedidos de las vacunas a las farmacéuticas. Es necesario conocer la fecha del pedido, la cantidad de unidades pedidas y si el pedido ha sido servido. De los pedidos se almacena también un código único. Cada pedido se refiere a una vacuna en particular; no hay pedidos mixtos.
   >
   > Los pedidos se entregan en lotes indicándose la fecha en la que se hace cada entrega. Un lote tiene un código de lote, el nombre de la vacuna al que pertenece, el número de unidades del lote y la fecha de caducidad del lote. Por último, se almacenan los datos de todas las personas del censo. De las personas se almacena un código de persona único, nombre y apellidos, fecha de nacimiento, dirección y teléfonos móvil y fijo. En los centros de vacunación también podrán darse de alta personas que no están censadas, para su control en proceso de vacunación.
   >
   > Cuando una persona es citada para vacunarse por primera vez, se registra la fecha y hora de esa cita y el centro de vacunación. Cuando la persona acuda a vacunarse, se le pregunta si ha pasado la enfermedad, en cuyo caso, se almacenará este dato y sólo se le administrará una dosis de la vacuna, dándose el proceso de vacunación por terminado. Si la persona no ha pasado la enfermedad, si la vacuna que hay disponible es monodosis, se le administra, se registra que ha sido vacunada con una monodosis y se da el proceso de vacunación por terminado. Si la vacuna en de dos dosis, dependiendo la vacuna que se le va a administrar (pueden tener diferentes periodos entre dosis), se le da cita para la segunda dosis. Todo esto debe quedar registrado. En cualquiera de los casos, es importante que quede registrado qué vacuna se les ha puesto y de qué lote.
   >
   > Cuando una persona va a vacunarse de la segunda dosis, se almacena la fecha y hora de vacunación, la vacuna (no necesariamente la misma que

en la primera dosis) y se da por terminado el proceso de vacunación. El gobierno quiere que se almacene un histórico que permita, entre otras cosas, conocer el porcentaje de vacunas entregadas en fecha por las farmacéuticas y la evolución global del porcentaje de población vacunada completa o parcialmente por edades con cada vacuna.

Se pide: Se pide realizar el Modelo Entidad-Relación del problema descrito anteriormente empleando la notación de Chen, la utilización de cualquier otro formato será penalizada.

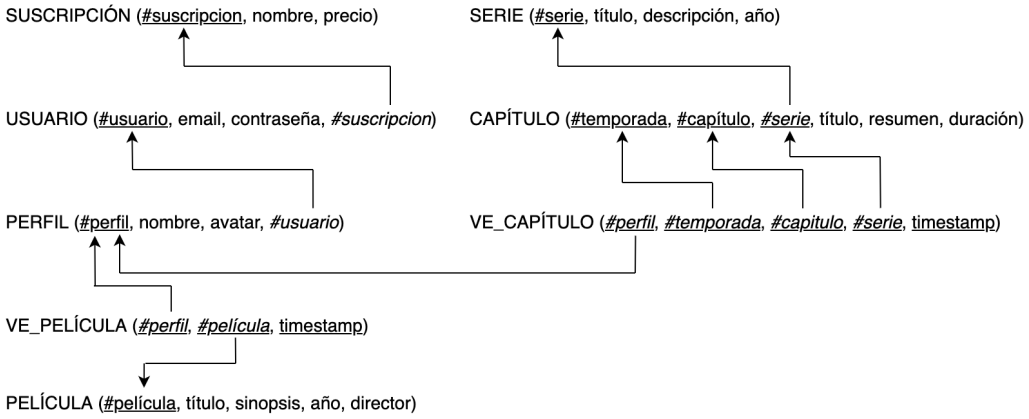
IMPORTANTE: Se penalizará la inclusión de elementos no necesarios o redundantes.

Solución propuesta:



2. (3½ puntos) **Consultas**

Utilizando el siguiente modelo relacional en el que las claves primarias aparecen subrayadas y las claves foráneas aparecen en *cursiva*, resuelva las siguientes consultas:



(a) **Álgebra relacional:**

I. (½ Punto) Obtener el `#perfil` de los perfiles que hayan visto las películas que llevan por título "*Casablanca*" y "*El Padrino*".

Solución propuesta:

$$\Pi_{\#perfil}\left(\sigma_{titulo="Casablanca"}\left(PELICULA\right) \bowtie VE\_PELICULA\right)$$
$$\bigcap$$
$$\Pi_{\#perfil}\left(\sigma_{titulo="ElPadrino"}\left(PELICULA\right) \bowtie VE\_PELICULA\right)$$

II. (½ Punto) Obtener el `#perfil` de los perfiles que hayan visto al menos un capítulo de todas las series del año 2020.

Solución propuesta:

$$\Pi_{\#perfil,\#serie}\left(VE\_CAPITULO\right) \div \Pi_{\#serie}\left(\sigma_{ano=2020}\left(SERIE\right)\right)$$

III. (½ Punto) Obtener el `título` de todas las series en las que todos los episodios tengan una duración igual o superior a 60 minutos.

Solución propuesta:

$$\Pi_{titulo}\left(SERIE \bowtie \left(\Pi_{\#serie}\left(SERIE\right) - Pi_{\#serie}\left(\sigma_{duracion<60}\left(CAPITULO\right)\right)\right)\right)$$

(b) **SQL:**

I. (½ Punto) Obtener el avatar y el nombre de aquellos perfiles de usuario que hayan visto todas las películas del director *"Steven Spielberg"*.

Solución propuesta:

```
SELECT nombre, avatar
FROM PERFIL perf
WHERE NOT EXISTS (
    SELECT *
    FROM PELÍCULA peli
    WHERE director = "Steven Spielberg" AND NOT EXISTS (
        SELECT *
        FROM VE_PELÍCULA ve
        WHERE ve.#película = peli.#película
                AND perf.#perfil = ve.#perfil))
```

II. (½ Punto) Obtener el título y número total de visualizaciones de aquellos capítulos que han sido vistos por más de 50 perfiles diferentes y que además la serie a la que pertenecen son del año 2010.

Solución propuesta:

```
SELECT cap.título, COUNT(*) AS  Total_visualizaciones
FROM CAPÍTULO cap
    INNER JOIN SERIE ON cap.#serie = serie.#serie
    INNER JOIN VE_CAPITULO ve ON cap.#temporada = ve.#temporada
                                AND cap.#capítulo = ve.#capítulo
                                AND cap.#serie = ve.#serie
WHERE SERIE.año = 2010
GROUP BY #capítulo, #temporada, #serie, cap.título
HAVING COUNT(DISTINCT #perfil) > 50
```

III. (½ Punto) Obtener el nombre y avatar así como el total de películas, del usuario que ha visualizado el mayor número de películas.

Solución propuesta:

```
SELECT nombre, avatar, COUNT(*) Total_visualizaciones
FROM PERFIL per
    INNER JOIN VE_PELÍCULA vep ON vep.#perfil = per.#perfil
GROUP BY #perfil, nombre, avatar
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM VE_PELÍCULA
                        GROUP BY #perfil)
```

IV. (½ Punto) Obtener el email de aquellos suscriptores que solo han visto películas.

Solución propuesta:

```
SELECT email
FROM USUARIO
    INNER JOIN PERFIL ON USUARIO.#usuario = PERFIL.#usuario
    INNER JOIN VE_PELICULA ON VE_PELICULA.#perfil = PERFIL.#perfil
WHERE USUARIO.#usuario
  NOT IN (SELECT USUARIO.#usuario
          FROM USUARIO
              INNER JOIN PERFIL
                ON PERFIL.#usuario = USUARIO.#usuario
              INNER JOIN VE_CAPITULO
                ON VE_CAPITULO.#perfil = PERFIL.#perfil)
```

3. (3 puntos) **Gestión**

   (a) (1 Punto) Diseñe un procedimiento que pasándole como parámetros de entrada dos años, devuelva como parámetro de salida el título de la película más vista por los usuarios entre los años pedidos. Añadir al final del ejercicio cómo ejecutaríamos el procedimiento y cómo obtendríamos el título de la película más vista.

Solución propuesta:

```
DELIMITER //
CREATE PROCEDURE peliculaMasVista(IN anyo1 INTEGER, IN anyo2 INTEGER,
                        OUT salida VARCHAR(200))
BEGIN
    SELECT titulo INTO salida
    FROM VE_PELICULA JOIN PELICULA
    ON VE_PELICULA.#pelicula=PELICULA.#pelicula
    WHERE año BETWEEN anyo1 AND anyo2
    GROUP BY titulo
    HAVING COUNT(*) >= ALL(SELECT COUNT(*)
                           FROM VE_PELICULA JOIN PELICULA
                           ON VE_PELICULA.#pelicula=PELICULA.#pelicula
                           WHERE año BETWEEN anyo1 AND anyo2
                           GROUP BY titulo);
END//
DELIMITER ;

CALL peliculaMasVista(2001,2002,@masVista);
SELECT @masVista;
```

(b) (1 ½ Puntos) Se debe diseñar un trigger que se encargará de eliminar los capítulos que ha visto un usuario (tabla VE_CAPITULO) cuando se borre de la BD un capítulo.

¡Atención! - El trigger tiene que eliminar las filas de una en una, por tanto, es **obligatorio** usar cursores.

Solución propuesta:

```
DELIMITER //
CREATE TRIGGER eliminar_vistos AFTER DELETE ON CAPÍTULO
FOR EACH ROW
BEGIN
    DECLARE aux_perfil INTEGER;
    DECLARE aux_temporada INTEGER;
    DECLARE aux_capitulo INTEGER;
    DECLARE aux_serie INTEGER;
    DECLARE aux_timestamp INTEGER;
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE c CURSOR FOR
        SELECT perfil, temporada, capitulo, serie, timestamp
        FROM VE_CAPITULO
        WHERE capitulo = OLD.capitulo AND temporada=OLD.temporada
                    AND serie=OLD.serie;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN c;
    read_loop: LOOP
        FETCH c INTO aux_perfil, aux_temporada, aux_capitulo, aux_serie,
                    aux_timestamp;
        IF done THEN
            LEAVE read_loop;
        END IF;
        DELETE FROM VE_CAPITULO WHERE perfil = aux_perfil
            AND temporada = aux_temporada
            AND capitulo = aux_capitulo AND serie = aux_serie
            AND timestamp = aux_timestamp;
END LOOP;
    CLOSE c;
END//
DELIMITER ;
```

(c) (½ Punto) Ana quiere comprar una entrada para ver "Rápido y Furioso XXV" en el cine del barrio. Inicia sesión en la plataforma web de compra de entradas y procede a seleccionar la butaca. En el mismo momento, su vecina Leonor se dispone también a seleccionar butaca para la misma película, día y hora. Ambas vecinas han iniciado sesión en la plataforma web prácticamente al mismo tiempo, por lo que la pantalla de selección muestra las mismas butacas disponibles. Ana es muy supersticiosa, por lo que estalla de alegría cuando selecciona la butaca de su número favorito, el 13. Está tan contenta que decide llamar a su hermano para contárselo. Mientras tanto, Leonor ha seleccionado la misma butaca número 13 y, como no cree en la suerte, simplemente ha efectuado el pago.

El día de emisión de la película, Ana y Leonor se encuentran en el cine y, sorprendentemente, ambas tienen la misma butaca asignada. **¿Qué problema concreto ha ocurrido en la base de datos de la plataforma de ventas? ¿Cómo solucionarías este problema para que en un futuro no se vuelva a repetir?**

Solución propuesta:
Tenemos un problema derivado del acceso concurrente a la base de datos. La plataforma de venta de entradas no utiliza transacciones, por lo que ha permitido que Ana compre su butaca 13, sobrescribiendo la asignación que el sistema había hecho previamente de esa butaca a Leonor. Por tanto, estamos ante una lectura sucia. Para solucionarlo tenemos que utilizar transacciones en todas las consultas que realice la plataforma de venta.

4. (1½ puntos) **Ficheros**

   (a) (1 Punto) Dado el siguiente fichero `books.xml`:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<books>
   <book category="JAVA">
      <title lang="en">Learn Java in 24 Hours</title>
      <author>Robert</author>
      <year>2005</year>
      <price>30.00</price>
   </book>

   <book category="DOTNET">
      <title lang="en">Learn .Net in 24 hours</title>
      <author>Peter</author>
      <year>2011</year>
      <price>40.50</price>
   </book>

   <book category="XML">
      <title lang="en">Learn XQuery in 24 hours</title>
      <author>Robert</author>
      <author>Peter</author>
      <year>2013</year>
      <price>50.00</price>
   </book>
</books>
```

   I. Diseña una consulta en XQuery para obtener información sobre el autor o autores del libro más caro. El resultado de la consulta debe ser el siguiente:

```xml
<authors>
    <author>Robert</author>
    <author>Peter</author>
</authors>
```

   Solución propuesta:

```
for $x in doc("books.xml")/books/book
where $x/price = max($x/price)
return
    <authors>
        $x/author
    </authors>
```

   II. ¿Cuál es el resultado **exacto** de la siguiente consulta XQuery?

```
for $x in doc("books.xml")/books/book
where $x/price > 30
return $x/title
```

   Solución propuesta:

```xml
<title lang="en">Learn .Net in 24 hours</title>
<title lang="en">Learn XQuery in 24 hours</title>
```

(b) (½ Punto) El siguiente fichero `asignaturas.json` usa un diseño embebido para representar asignaturas y los alumnos que están matriculados en ellas. Nombre y justifique todos los errores que encuentres en el fichero JSON:

```
BBDD = {
    Codigo = "bbdd123"
    Nombre = "Bases de Datos"
    Alumnos = {
        {
            ID = 1234
            Edad = 21
        }
        {
            ID = 2134
            Edad = 27
        }
    }
}
BDA = {
    Codigo = "bbddaa111"
    Nombre = "Bases de Datos Avanzadas"
    Alumnos = {
        {
            ID = 3325
            Edad = 25
        }
        {
            ID = 5547
            Edad = 25
        }
    }
}
```

Solución propuesta:

1. No existe un objeto contenedor común.
2. No hay comas separando los objetos.
3. Array de objetos definido con llaves en lugar de corchetes.
4. Se ha usado el símbolo = en lugar de : para separar la clave de su valor.
5. Nombre de las claves sin comillas.