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.
- 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 a los 15 días hábiles desde la fecha de realización del examen.
- La fecha para la revisión del examen se anunciará en el Moodle de la asignatura junto a la publicación de las calificaciones.

BLOQUE 1. (2½ Puntos) **Modelado**

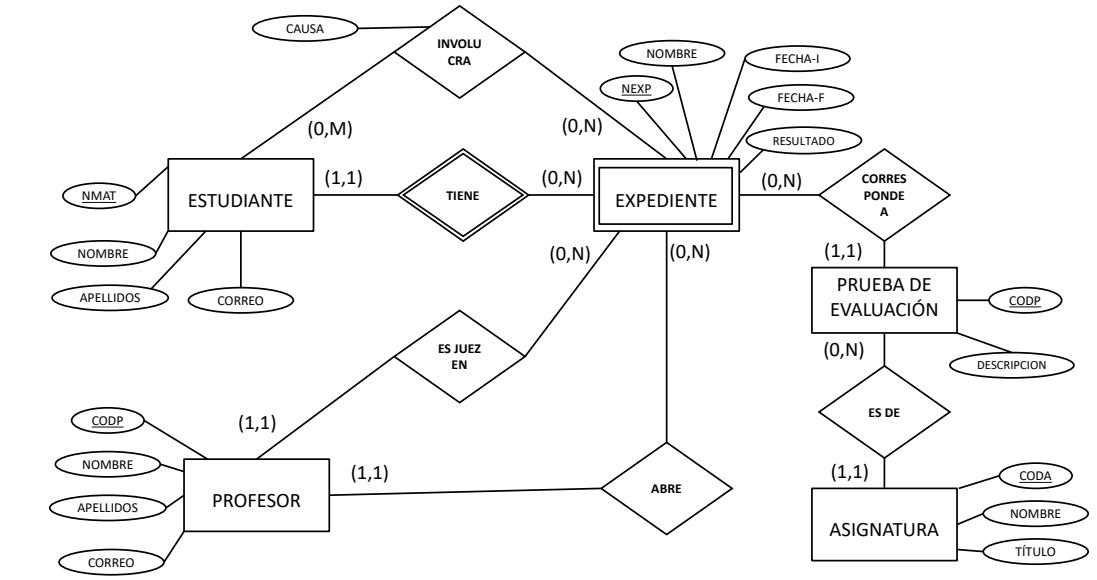
Realizar un modelo conceptual de datos mediante la técnica del **modelo Entidad-Relación de Chen** teniendo en cuenta la siguiente descripción:

> Una universidad quiere implantar un sistema para el seguimiento y control de los expedientes que se les abren a los estudiantes en caso de copia o cualquier otro tipo de fraude. Para ello debemos realizar un modelo de datos que soporte los siguientes requisitos:
>
> Cuando un profesor detecta una copia, pide la apertura de un expediente informativo. En ese momento, se crea un nuevo expediente, indicando la fecha de inicio, el estudiante al que se le abre el expediente, la prueba de evaluación en la que se ha detectado la copia y una descripción del fraude (podría no ser una copia y ser una suplantación, por ejemplo). Un expediente se referirá a un único estudiante, es abierto por un único profesor y se referirá a una única prueba de evaluación. Además, se asignará un único profesor que actuará como juez instructor del expediente.
>
> Los datos que se manejarán de los estudiantes serán su número de matrícula, nombre y apellidos, correo electrónico y número de expedientes condenatorios que ha tenido en la universidad. En cuanto a los profesores, se almacenará un código de profesor, nombre y apellidos y correo electrónico. De las pruebas de evaluación, se almacenará un código de prueba y una descripción.
>
> A lo largo de la investigación, pueden aparecer varios estudiantes relacionados con el expediente abierto, ya sea como colaboradores necesarios del fraude o como perjudicados, es decir, estudiantes a los que se les ha copiado. Es importante tener almacenados qué estudiantes están relacionados con los expedientes y la causa.
>
> Cuando se resuelve el expediente, se incluye la fecha de resolución y el resultado, que puede ser sobreseído o condenatorio. En caso de que sea sobreseído, desaparecerá de la base de datos, como si nunca hubiese ocurrido. En caso de que sea condenatorio, el estudiante tendrá que hacer un examen especial en la siguiente convocatoria de la asignatura a la que corresponde la prueba de evaluación en la que se produjo la copia. Esto debe de quedar registrado para que se realice su aplicación cuando llegue el momento. De las asignaturas se almacena un código de asignatura, el nombre y el curso al que pertenece. Un expediente condenatorio permanecerá en la base de datos hasta que el estudiante se gradúe o deje de pertenecer a la universidad.

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.

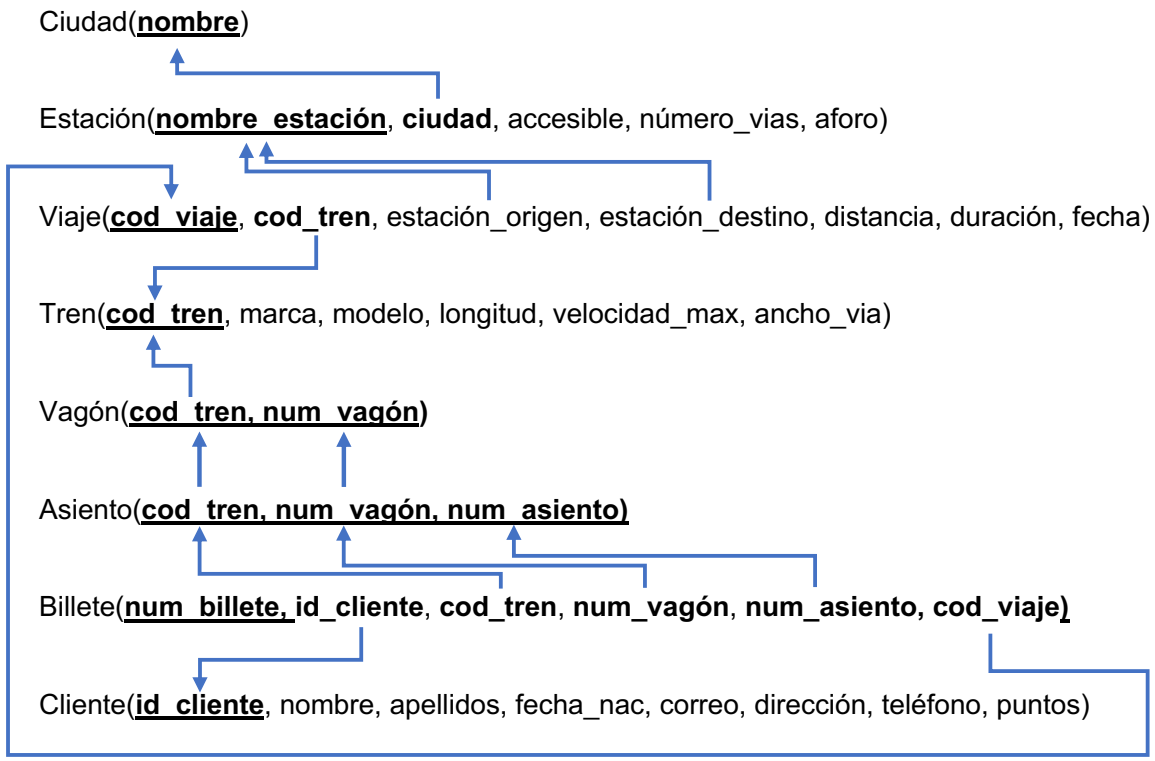
Solución propuesta:



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

BLOQUE 2. (3 puntos) Consultas

Dado el siguiente modelo relacional:



Se pide escribir la solución a las siguientes consultas.

(a) **Álgebra relacional:**

1. (½ Punto) Nombre y apellidos de los clientes que han viajado desde todas las estaciones de Madrid.

Solución propuesta:

$$\Pi_{nombre,apellidos,estacion\_origen}\left(Cliente \bowtie Billete \bowtie Viaje\right)$$
$$\div$$
$$\Pi_{nombre\_estacion}\left(\sigma_{ciudad='Madrid'}Estacion\right))$$

II. (½ Punto) Nombre de las estaciones de las que únicamente parten trenes modelo `AVE`.

Solución propuesta:

$$\Pi_{estacion\_origen}\,(Viaje \bowtie \sigma_{modelo='AVE'}\,(Tren))$$

$$-$$

$$\Pi_{estacion\_origen}\,(Viaje \bowtie \sigma_{modelo\neq'AVE'}\,(Tren))$$

III. (½ Punto) Nombre y apellidos de los pasajeros que han viajado desde o hacia Madrid, Barcelona y Valencia.

Solución propuesta:

$$\Pi_{nombre,apellidos}(Billete \bowtie Cliente \bowtie$$
$$\sigma_{est\_origen.ciudad='Madrid'\ \vee\ est\_destino.ciudad='Madrid'}\,(Estacion \bowtie Viaje))$$
$$\cap$$
$$\Pi_{nombre,apellidos}(Billete \bowtie Cliente \bowtie$$
$$\sigma_{est\_origen.ciudad='Barcelona'\ \vee\ est\_destino.ciudad='Barcelona'}\,(Estacion \bowtie Viaje))$$
$$\cap$$
$$\Pi_{nombre,apellidos}(Billete \bowtie Cliente \bowtie$$
$$\sigma_{est\_origen.ciudad='Valencia'\ \vee\ est\_destino.ciudad='Valencia'}\,(Estacion \bowtie Viaje))$$

(b) **SQL:**

I. (½ Punto) Obtener el número total de asientos disponibles para viajar de Madrid a Barcelona el 5 de julio de 2022.

Solución propuesta:

```
SELECT * from viaje
    INNER JOIN tren ON viaje.cod_tren = tren.cod_tren
    INNER JOIN vagon ON tren.cod_tren=vagon.cod_tren
    INNER JOIN asiento ON vagon.cod_tren=asiento.cod_tren
    AND vagon.num_vagon=asiento.num_vagon
    WHERE viaje.fecha="2022/07/05" AND
    viaje.estacion_origen IN
        (SELECT nombre_estacion
            FROM estacion WHERE ciudad="madrid") AND
    viaje.estacion_destino IN
        (SELECT nombre_estacion
            FROM estacion WHERE ciudad="barcelona");
```

II. (½ Punto) Obtener el cliente con mayor número de puntos, pero únicamente si ha viajado en trenes de la marca Talgo (puede darse el caso de que ningún cliente cumpla la condición).

Solución propuesta:

```
SELECT cliente.nombre, cliente.puntos FROM cliente
    INNER JOIN billete ON cliente.id_cliente=billete.id_cliente
        INNER JOIN viaje ON viaje.cod_viaje=billete.cod_viaje
    INNER JOIN tren ON billete.cod_tren=tren.cod_tren
        WHERE tren.marca="talgo"
        GROUP BY cliente.nombre, cliente.apellidos, cliente.puntos
        HAVING cliente.puntos >=
            ALL (SELECT cliente.puntos FROM cliente);
```

III. (½ Punto) Obtener el nombre y apellidos de los clientes que nunca han viajado en un tren modelo "AVE".

Solución propuesta:

```
SELECT cliente.nombre, cliente.apellidos
FROM cliente
WHERE id_cliente NOT IN (
    SELECT cliente.id_cliente FROM cliente INNER JOIN billete
        ON cliente.id_cliente=billete.id_cliente
        INNER JOIN tren ON billete.cod_tren=tren.cod_tren
        WHERE tren.modelo="AVE");
```

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

BLOQUE 3. (2½ puntos) Gestión

(a) (½ Punto) Se quiere mejorar la eficiencia de la consulta SQL III del Bloque 2 mediante la creación de un índice. Por lo tanto, en base a la consulta realizada anteriormente, genere la sentencia SQL de creación del índice adecuado, y explique el motivo por el que dicho índice conseguiría optimizar la consulta.

Solución propuesta:

```
ALTER TABLE Tren ADD INDEX optimizar (modelo);
```

Explicación: Si nos fijamos en la consulta que se quiere optimizar, vemos que se van a tener que buscar aquellos trenes que cumplan la condición de que su campo 'modelo' en la tabla 'tren' tenga el valor 'AVE'. Por lo tanto, para encontrar aquellas filas que cumplan esta condición de una manera más rápida, evitando tener que realizar la lectura secuencial de todas las filas, se tiene que añadir un índice que contenga el campo sobre el que se realiza la búsqueda (modelo) de la tabla afectada (tren).

(b) (1 Punto) En base al modelo relacional del Bloque 2, escriba una función que utilizando un cursor devuelva el número total de viajes que se ha realizado para un día concreto donde la estación de origen y destino son de la misma ciudad. El día se deberá pasar como parámetro de entrada.

Solución propuesta:

```
DELIMITER $$
CREATE FUNCTION totalViajes (dia DATE)
RETURNS INTEGER
DETERMINISTIC
BEGIN
    DECLARE total INTEGER DEFAULT 0;
    DECLARE done INT DEFAULT FALSE;
    DECLARE ciudad_origen, ciudad_destino VARCHAR(50);
    DECLARE cur1 CURSOR FOR SELECT e_ori.ciudad as ciudad_o,
                                   e_des.ciudad as ciudad_d
                            FROM Viaje v, Estacion e_ori, Estacion e_des
                            WHERE v.estacion_origen = e_ori.nombre_estacion
                            AND v.estacion_destino = e_des.nombre_estacion
                            AND v.fecha = dia;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur1;
    read_loop: LOOP
```

```
        FETCH cur1 INTO estacion_origen, estacion_destino;
        IF done THEN
            LEAVE read_loop;
        END IF;
        IF ciudad_origen = ciudad_destino THEN
            SET total = total + 1;
    END LOOP;
    CLOSE cur1;

    RETURN (total);
END$$
DELIMITER ;
```

(c) (1 Punto) En base también al modelo relacional del Bloque 2, cree un *trigger* que impida que un cliente pueda comprar más de 10 billetes con la misma estación origen y destino en el mismo día de la compra.

Solución propuesta:

```
DELIMITER $$
CREATE TRIGGER noMasDeDiezBilletes
BEFORE INSERT ON Billete
FOR EACH ROW
BEGIN
    DECLARE numResults INTEGER;
    DECLARE numDrivers INTEGER;

    SELECT COUNT(*) INTO numResults
        FROM Billete INNER JOIN Viaje on Billete.cod_viaje = Viaje.cod_viaje
        WHERE id_cliente = NEW.id_cliente
        AND estacion_origen = NEW.estacion_origen
        AND estacion_destino =  NEW.estacion_destino
        AND day = CURDATE();

    IF numResults > 10 THEN
        SIGNAL SQLSTATE '03000'
        SET MESSAGE_TEXT = 'Error: no puede comprar mas de 10 billetes hoy';
    END IF;

END$$
DELIMITER ;
```

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

BLOQUE 4. (1 puntos) Ficheros

(a) (1 Punto) Modifique los siguientes documentos en formato JSON para que represente la relación 1:N embebida.

Miniaturas

```
{
  _id: 1234,
  nombre: "Ghazghkull Thraka",
  autor: [ "Brian Nelson", "Citadel" ],
  precio: 80
}
{
  _id: 1235,
  nombre: "Deffkoptas",
  autor: "Citadel",
  precio: 115
}
{
  _id: 1236,
  nombre: "Mozrog Skragbad",
  autor: "Citadel",
  precio: 95
}
```

Comentarios

```
{
    _id: 1,
    autor: "Bruce Wayne",
    miniatura_id="1236",
    comentario: "A mi me encantó"
}
{
    _id: 2,
    autor: "Bruce Wayne",
    miniatura_id="1234",
    comentario: "Podría ser mejor"
}
{
    _id= 3,
    autor: "Felix Paniagua",
    miniatura_id="1234",
    comentario: "No me gusta"
}
```

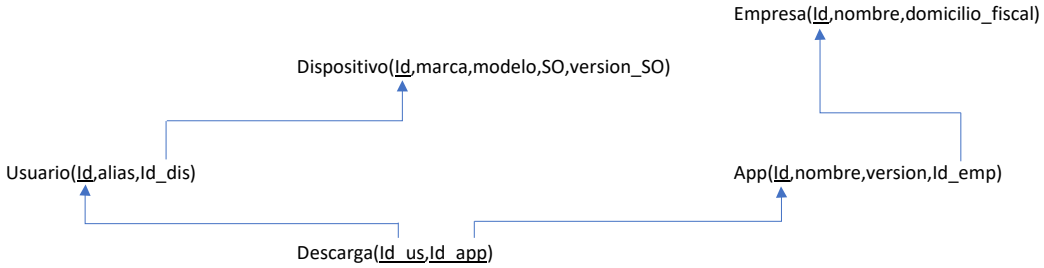
Solución propuesta:

```
{
  _id: 1234,
  nombre: "Ghazghkull Thraka",
  autor: [ "Brian Nelson", "Citadel" ],
  precio: 80,
  comentarios: [
      {
        _id: 2,
        autor: "Bruce Wayne",
        comentario: "Podría ser mejor"
    },
    {
        _id= 3,
        autor: "Felix Paniagua",
        comentario: "Quizas hubiera dado más detalle a las armas"
    }
  ]
}
{
  _id: 1235,
  nombre: "Deffkoptas",
  autor: "Citadel",
  precio: 115
}
{
  _id: 1236,
  nombre: "Mozrog Skragbad",
  autor: "Citadel",
  precio: 95,
  comentarios: [ { _id: 1, autor: "Bruce Wayne", comentario: "A mi me encantó" } ]
}
```

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

BLOQUE 5. (1 puntos) Programación contra bases de datos

(a) (1 Punto) Una *start-up* ha desarrollado una tienda de aplicaciones para sus clientes y te contrata para cubrir la baja de paternidad del ingeniero que se encargaba de conectar la base de datos al *back-end* a través del ORM para Java **Hibernate**. El modelo relacional de la base de datos se muestra en la siguiente imagen:



En resumen, lo que tratan de almacenar son las aplicaciones que se descarga cada usuario. También les interesa llevar un registro de las aplicaciones que desarrollan las empresas con las que colaboran. Ten en cuenta que una empresa puede crear más de una aplicación. Por último, les gustaría almacenar datos sobre el dispositivo en el cual se registra cada usuario. En este punto es importante saber que un usuario sólo se puede registrar en un dispositivo a la vez.

Te toca completar la tarea y para ello te piden que empieces por etiquetar las clases y sus atributos para que Hibernate pueda realizar la conexión con la base de datos satisfactoriamente cumpliendo con el modelo relacional mostrado anteriormente.

```
public class Usuario {

    private Long id;

    private String alias;

    private Set<App> aplicaciones;

    private Dispositivo dispositivo;

    /*Constructor de la clase, getters, setters,...*/
}
```

```
public class App {

    private Long id;

    private String nombre;

    private String version;

    private Set<Usuario> usuarios;

    private Empresa empresa;

    /*Constructor de la clase, getters, setters,...*/
}

public class Empresa {

    private Long id;

    private String nombre;

    private String domicilio_fiscal;

    private Set<App> apps;

    /*Constructor de la clase, getters, setters,...*/
}

public class Dispositivo {

    private Long id;

    private String make;

    private String model;

    private String os;

    private String os_version;

    /*Constructor de la clase, getters, setters,...*/
}
```

Solución propuesta:

```
@Entity
@Table(name="User")
public class Usuario {
    @Id
    @Column(name="id", unique = true, nullable = false)
    private Long id;

    @Column(name="alias", nullable = false)
    private String alias;

    @ManyToMany(mappedBy="usuarios")
    private Set<App> aplicaciones;

    @OneToOne
    @JoinColumn(name="Device_id")
    private Dispositivo dispositivo;
    /*Constructor de la clase, getters, setters,...*/
}
@Entity
@Table(name="App")
public class App {
    @Id
    @Column(name="id", unique = true, nullable = false)
    private Long id;

    @Column(name="name", nullable = false)
    private String nombre;

    @Column(name="version", nullable = false)
    private String version;

    @ManyToMany
    @JoinTable(name="Download")
    private Set<Usuario> usuarios;

    @ManyToOne(optional = false)
    @JoinColumn(name="company")
    private Empresa empresa;
    /*Constructor de la clase, getters, setters,...*/
}

@Entity
@Table(name="Company")
```

```java
public class Empresa {
    @Id
    @Column(name="id", unique = true, nullable = false)
    private Long id;

    @Column(name="name", nullable = false)
    private String nombre;

    @Column(name="fiscalAdd", nullable = false)
    private String domicilio_fiscal;

    @OneToMany(mappedBy = "empresa")
    private Set<App> apps;
    /*Constructor de la clase, getters, setters,...*/
}

@Entity
@Table(name="Device")
public class Dispositivo {
    @Id
    @Column(name="id", unique = true, nullable = false)
    private Long id;

    @Column(name="make", nullable = false)
    private String make;

    @Column(name="model", nullable = false)
    private String model;

    @Column(name="os", nullable = false)
    private String os;

    @Column(name="os_version", nullable = false)
    private String os_version;
    /*Constructor de la clase, getters, setters,...*/
}
```