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. (3 Puntos) Modelado

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

La ONG "Salvemos a las aves" se dedica a prestar servicios a sus asociados interesados en el conocimiento y estudio de aves. Su principal cometido es colocar anillas identificativas en las patas de diferentes especies de aves en el territorio nacional para facilitar el estudio de sus poblaciones, costumbres, rutas migratorias, etc.

Esta ONG pretende diseñar una BD Relacional para almacenar toda la información relativa a sus anillamientos y servicios prestados.

La ONG dará una serie de servicios a sus asociados de los que se almacena el nombre y el tipo (personas u organizaciones). Los servicios ofrecidos por la ONG pueden ser, por ejemplo, distribución de publicaciones, visitas a reservas naturales o cursos de cría y cuidado de aves. Cada servicio tiene asignado un código, un coste y una descripción del servicio.

Cada anilla lleva una clave que identifica al ave que lo lleva. Cuando la ONG pone una anilla a un ave en particular, se le asigna un "padrino" que será alguno de los asociados, sin importar si se trata de personas u organizaciones. Cada ave sólo puede tener un único padrino. Se han de almacenar también el nombre científico del ave identificativo de la especie que controla, las características generales de cada una de las especies (color, tipo de pico, tipo de vuelo, alas), así como un campo para las características particulares de cada ave (longitud del pico, longitud de las alas, edad, etc).

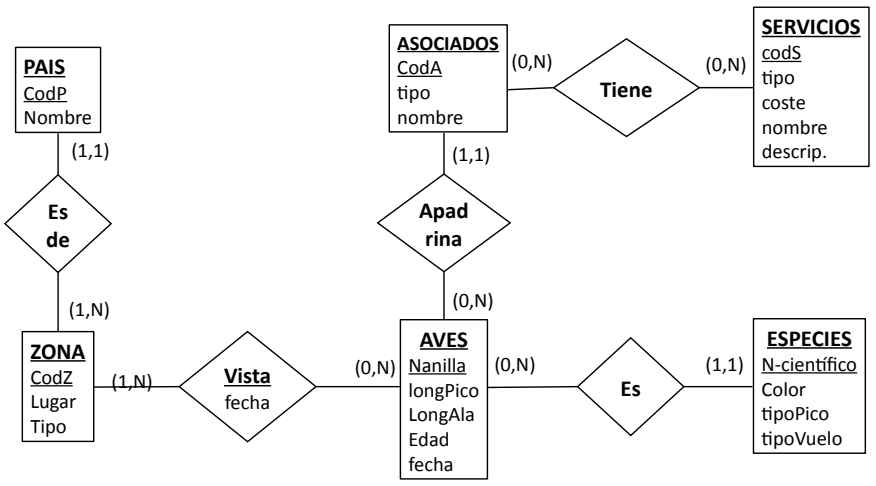
Para poder estudiar las rutas migratorias, se han de almacenar los países donde se ha observado cada ave particular, la fecha de la observación y el lugar de observación y tipo de ecosistema donde se encontraba el ave. Hay que tener en cuenta que algunas especies recorren grandes distancias cada día, por lo que no es inusual que sean observadas el mismo día en zonas muy diferentes.

`Las restricciones a considerar son las siguientes:`

- Un ave concreta pertenece a una única especie.
- Un ave concreta tiene una sola descripción de sus características.
- El color, tipo de pico, de vuelo y las alas son características únicas asociadas a cada especie.

Se pide realizar un modelo entidad-relación de Chen justificando todas las cardinalidades mínimas.

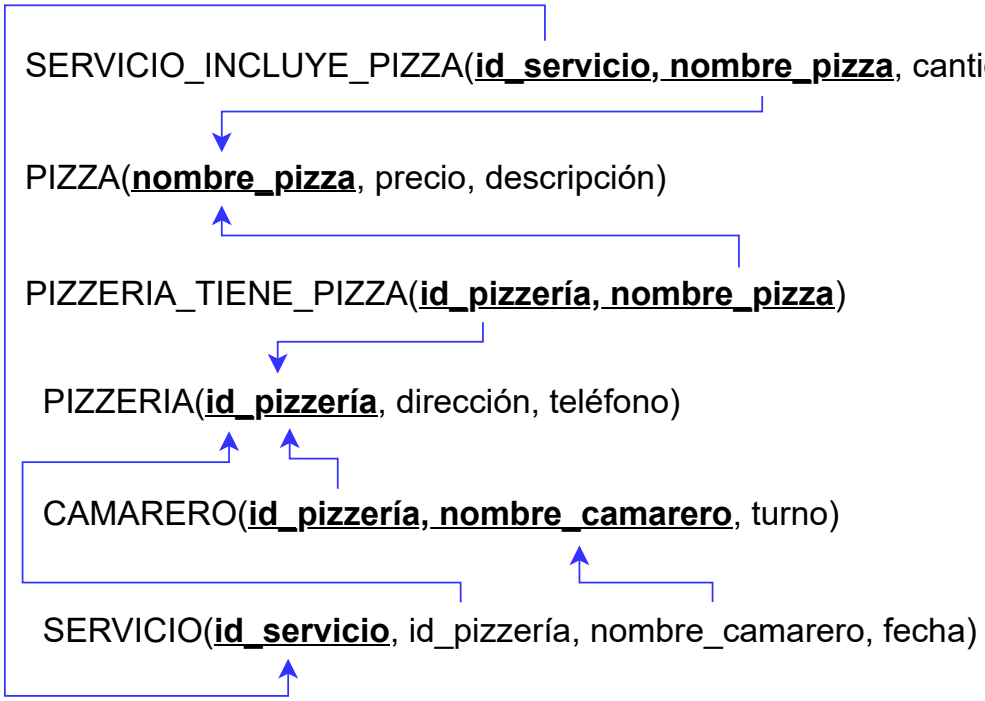
Solución propuesta:



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

BLOQUE 2. (3 puntos) Consultas

Una conocida franquicia de pizzerias está modernizando su sistema de gestión y ha incluído una base de datos a la que nos pide que hagamos ciertas consultas. Para ello nos proporciona el diagrama relacional de dicha base de datos en producción.



Se pide escribir la solución a las siguientes consultas.

(a) **Álgebra relacional:**

I. (½ Punto) Obtener el nombre y precio de las pizzas de las que dispone la pizzería situada en "Ronda de Segovia" .

Solución propuesta:

$$\Pi_{nombre\_pizza,precio}(Pizza \bowtie Pizzeria\_tiene\_pizza \bowtie \sigma_{direccion='Ronda\ de\ Segovia'}(Pizzeria))$$

II. (½ Punto) Obtener el teléfono de las pizzerías que tienen camareros que trabajen únicamente en el turno de "Mañana".

Solución propuesta:

$$\Pi_{telefono}(Pizzeria \bowtie \sigma_{turno='Manana'}(Camarero)) - \Pi_{telefono}(Pizzeria \bowtie \sigma_{turno!='Manana'}(Camarero))$$

III. (½ Punto) Obtener las direcciones de las pizzerías que tienen todas las pizzas disponibles.

Solución propuesta:

$$\Pi_{direccion}(Pizzeria \bowtie \Pi_{id\_pizzeria,nombre\_pizza}(Pizzeria\_tiene\_pizza) \div \Pi_{nombre\_pizza}(Pizza))$$

(b) **SQL:**

I. (½ Punto) Obtener la pizzería que más pizzas sirvió el día 23/06/2023.

Solución propuesta:

```
SELECT pizzeria.id_pizzeria, pizzeria.direccion,
       SUM(servicio_incluye_pizza.cantidad)
FROM pizzeria INNER JOIN servicio ON
 pizzeria.id_pizzeria=servicio.id_pizzeria
 INNER JOIN servicio_incluye_pizza ON
 servicio.id_servicio=servicio_incluye_pizza.id_servicio
WHERE servicio.fecha="23/06/2023"
GROUP BY pizzeria.id_pizzeria
HAVING SUM(servicio_incluye_pizza.cantidad)
  >= ALL (SELECT SUM(cantidad)
          FROM servicio_incluye_pizza INNER JOIN servicio
           ON servicio_incluye_pizza.id_servicio=servicio.id_servicio
          WHERE servicio.fecha="23/06/2023"
          GROUP BY id_pizzeria);
```

II. (½ Punto) Obtener los datos de aquellos servicios en los que se hayan servido todas las pizzas.

Solución propuesta:

```
SELECT * FROM servicio
WHERE NOT EXISTS (
 SELECT * FROM pizza
 WHERE NOT EXISTS(
  SELECT * FROM servicio_incluye_pizza
  WHERE servicio.id_servicio=servicio_incluye_pizza.id_servicio));
```

III. (½ Punto) Obtener el id de los servicios que hayan pedido dos "Margarita" y al menos una "Diavola".

Solución propuesta:

```
SELECT servicio.id_servicio
FROM servicio
WHERE id_servicio IN (SELECT id_servicio
 FROM servicio_incluye_pizza
  WHERE nombre_pizza="Margarita" AND cantidad=2)
AND id_servicio IN (SELECT id_servicio
 FROM servicio_incluye_pizza
 WHERE nombre_pizza="Diavola");
```

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

BLOQUE 3. (2 puntos) Gestión

(a) (1 Punto) En base al modelo relacional del Bloque 2, crear un *TRIGGER* que impida cambiar de pizzería en la que trabajan a aquellos camareros que han realizado servicios en más de tres pizzerías distintas en el último mes (30 días).

Solución propuesta:

```
DELIMITER $$
CREATE TRIGGER chequeoCambioPizzeria
BEFORE UPDATE ON CAMARERO
FOR EACH ROW
BEGIN
    DECLARE num_cambios INTEGER;
    IF OLD.id_pizzería <> NEW.id_pizzería THEN
        SELECT COUNT(DISTINT id_pizzería) INTO num_cambios
        FROM SERVICIO
        WHERE nombre_camarero = NEW.nombre_camarero
        AND fecha BETWEEN DATE_ADD(CURDATE(), INTERVAL -30 DAY) AND CURDATE();

        IF num_cambios > 3 THEN
            SIGNAL SQLSTATE '03000'
            SET MESSAGE_TEXT = 'Error: no se pueden hacer mas de tres
                cambios de pizzerias en el ultimo mes';
        END IF;
    END IF;
END$$
DELIMITER ;
```

(b) (1 Punto) Explica qué es y para qué sirve una transacción en una base de datos, indicando cuáles son las propiedades que debe cumplir. Además, realiza un ejemplo concreto en SQL de cómo se realizaría el borrado de todos los camareros cuyo nombre comienza por "MARIA" utilizando una transacción.

Solución propuesta:

Una transacción en un base de datos es una unidad de trabajo compuesta por un conjunto de operaciones, cuyo resultado final debe ser que se ejecuten todas o ninguna de ellas. Sus propiedades, que se conocen como ACID, son las siguientes:

1. Atomicidad: las operaciones que componen una transaccion deben considerarse como una sola.

2. Consistencia: una operación nunca deberá dejar datos incoherentes.
3. Aislamiento: los datos "sucios" deben estar aislados, y evitar que los usuarios utilicen información que aún no está confirmada o validada.
4. Durabilidad: una vez completada la transacción los datos actualizados ya seran permanentes y confirmados.

   El SQL de borrado controlado con una transacción sería:

```
START TRANSACTION;
DELETE FROM CAMARERO WHERE nombre_camarero LIKE 'MARIA%';
COMMIT;
```

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

BLOQUE 4. (1 puntos) Ficheros

(a) (1 Punto) Seleccione una opción para cada pregunta, las respuestas **incorrectas restarán 0,1** puntos:

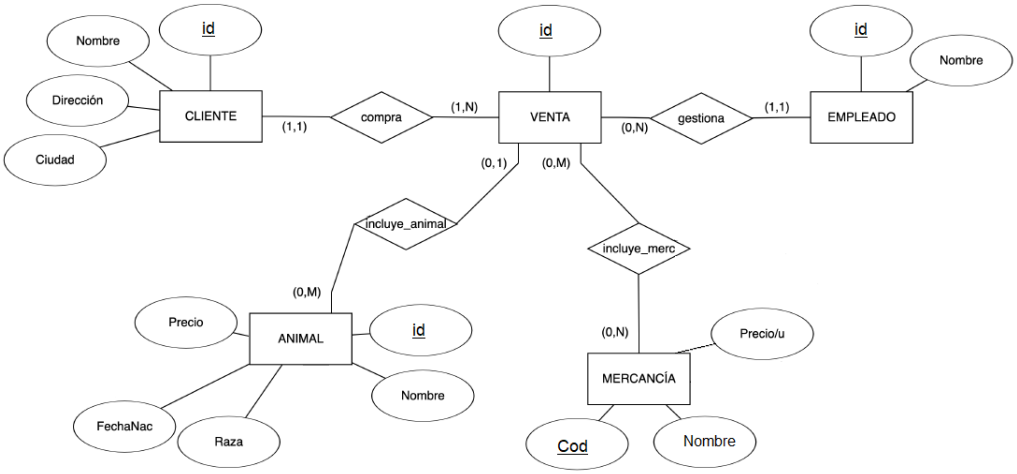
1. ¿Qué significa CSV y cuál es su estructura básica?
   a) Comma-Separated Values; estructura basada en pares clave-valor.
   b) Common Structured Values; estructura tabular con delimitadores de coma.
   c) Compressed Serial Values; estructura jerárquica con etiquetas.
2. ¿Cuáles son las ventajas y desventajas de utilizar XML como formato de intercambio de datos?
   a) Ventajas: legible para humanos, estructura jerárquica; Desventajas: mayor tamaño de archivo, mayor complejidad en el procesamiento.
   b) Ventajas: estructura tabular, soporte para metadatos; Desventajas: menor legibilidad para humanos, limitaciones en la representación de datos complejos.
   c) Ventajas: menor tamaño de archivo, mayor eficiencia; Desventajas: menor legibilidad para humanos, estructura rígida.
3. ¿Cuáles son las ventajas y desventajas de utilizar JSON como formato de intercambio de datos?
   a) Ventajas: legible para humanos, estructura flexible; Desventajas: menor eficiencia en tamaño de archivo, limitaciones en la representación de datos tabulares.
   b) Ventajas: estructura tabular, soporte para metadatos; Desventajas: mayor complejidad en el procesamiento, menor legibilidad para humanos.
   c) Ventajas: menor tamaño de archivo, mayor eficiencia; Desventajas: menor legibilidad para humanos, estructura jerárquica rígida.
4. ¿En qué escenarios es más apropiado utilizar el formato CSV en comparación con XML y JSON?
   a) Escenarios que requieren representar datos complejos y jerárquicos.
   b) Escenarios que requieren una estructura tabular para datos simples.
   c) Escenarios que requieren una representación legible para humanos con metadatos.
5. ¿En qué escenarios es más apropiado utilizar el formato XML en comparación con CSV y JSON?
   a) Escenarios que requieren una estructura tabular para datos simples.
   b) Escenarios que requieren representar datos complejos y jerárquicos.
   c) Escenarios que requieren una representación legible para humanos con metadatos.

Solución propuesta:
a a a b b

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) El siguiente modelo entidad-relación pertenece a una tienda de mascotas.



Realice el etiquetado de clases y sus atributos para que mediante el ORM **Hibernate** de Java pueda realizar la conexión con la base de datos satisfactoriamente cumpliendo con el modelo mostrado.

```
public class Cliente {

    private Long id;

    private String nombre;

    private String direccion;

    private String ciudad;

    private Set<Venta> compras;

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

```java
public class Empleado {

    private Long id;

    private String nombre;

    private Set<Venta> ventas;

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

```java
public class Animal{

    private Long id;

    private String nombre;

    private String raza;

    private Float precio;

    private Date fechaNac;

    private Venta venta;

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

```
public class Mercancia {

    private Long cod;

    private String nombre;

    private Float precioU;

    private Set<Venta> ventas;

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


public class Venta {

    private Long id;

    private Empleado empleado;

    private Cliente cliente;

    private Set<Animal> animales;

    private Set<Mercancia> mercancias;

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

Solución propuesta:

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

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

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

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

    @OneToMany(mappedBy = "cliente")
    private Set<Venta> compras;
     /*Constructor de la clase, getters, setters,...*/
}

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

    @Column(name="name")
    private String nombre;

    @OneToMany(mappedBy = "empleado")
    private Set<Venta> ventas;
    /*Constructor de la clase, getters, setters,...*/
}

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

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

    @Column(name="breed")
    private String raza;

    @Column(name="price", nullable = false)
    private Float precio;
```

```
    @Column(name="birthday", nullable = false)
    private Date fechaNac;

    @ManyToOne
    @JoinColumn(name = "sale_id")
    private Venta venta;
    /*Constructor de la clase, getters, setters,...*/
}

@Entity
@Table(name="goods")
public class Mercancia {
    @Id
    @GeneratedValue
    @Column(name="cod", unique = true, nullable = false)
    private Long cod;

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

    @Column(name="price", nullable = false)
    private Float precioU;

    @ManyToMany(mappedBy = "mercancias")
    private Set<Venta> ventas;
    /*Constructor de la clase, getters, setters,...*/
}

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

    @ManyToOne(optional = false)
    @JoinColumn(name = "employee_id")
    private Empleado empleado;

    @ManyToOne(optional = false)
    @JoinColumn(name = "client_id")
    private Cliente cliente;

    @OneToMany(mappedBy = "venta")
```

```
    private Set<Animal> animales;

    @ManyToMany
    @JoinTable(name = "Merch_is_in")
    private Set<Mercancia> mercancias;
    /*Constructor de la clase, getters, setters,...*/
}
```