EJERCICIO 1. (2½ Puntos) Modelado entidad relación

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

La organización EURO-NCAP es la entidad Europea de referencia encargada de evaluar y certificar la seguridad de vehículos a motor de distintas marcas y modelos mediante la realización de pruebas especializadas. Se fundó con el objetivo de mejorar los estándares de seguridad vial en Europa, de forma que se pudiera garantizar que los vehículos comercializados cumplieran con los requisitos más exigentes en materia de protección de los ocupantes y peatones. Para ello, EURO-NCAP, en colaboración con fabricantes, organismos reguladores y laboratorios independientes, diseña y ejecuta un conjunto de pruebas especializadas que abarcan distintos aspectos de la seguridad del vehículo, como, por ejemplo, la resistencia estructural, la eficacia de los sistemas de retención o el comportamiento ante colisiones.

Cada vez que un vehículo, identificado por su marca y modelo, es sometido a una de estas pruebas, se registra la calificación obtenida y la fecha en la que se realizó. Las pruebas se identifican por un nombre y tienen una duración determinada y un protocolo específico por el cual se rigen. Es importante destacar que un mismo vehículo puede participar en varias pruebas a lo largo de su vida, ya sea para evaluar diferentes aspectos de su seguridad o para comprobar mejoras introducidas por el fabricante. Asimismo, una misma prueba puede ser aplicada a diferentes vehículos, lo que permite comparar resultados y establecer clasificaciones entre modelos.

Cada vehículo está compuesto por diversos componentes, como el *airbag*, los frenos o el chasis, de los que se desea almacenar su nombre, que será único, y su descripción. Cada componente puede encontrarse en uno o más vehículos. Además, algunos componentes pueden estar formados a su vez por otros componentes del mismo vehículo.

En la realización de las pruebas participan ingenieros e ingenieras que pueden intervenir en varias pruebas y desempeñar diferentes roles en cada una de ellas (supervisor/a, técnico/a u observador/a). De cada ingeniero/a se conoce un identificador único, su nombre y su especialidad profesional. Es fundamental poder registrar no solo quien han participado en cada prueba, sino también el rol concreto que han desempeñado en cada una de ellas.

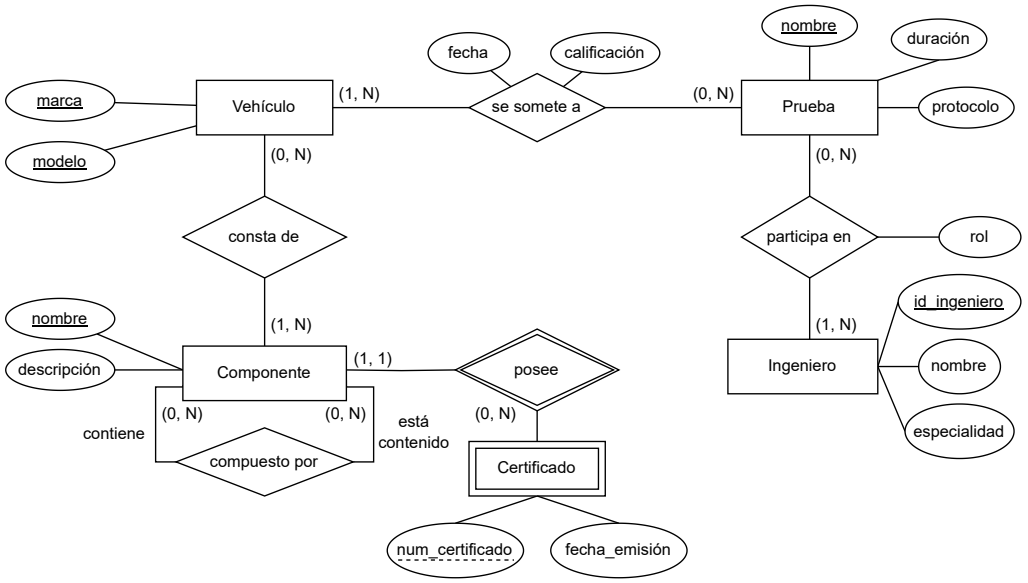
Por otro lado, cada componente puede estar asociado a uno o varios certificados de calidad. Un certificado de calidad tiene un número identificador (que puede repetirse) y una fecha de emisión, pero depende del componente al que está asociado, ya que un mismo número de certificado puede repetirse en componentes de distintos vehículos. Es necesario poder registrar los certificados de calidad de cada componente, así como la fecha en la que fueron emitidos.

Ante la complejidad y el volumen de información que maneja, la organización EURO-NCAP desea disponer de un sistema de información robusto y flexible que permita gestionar todos estos datos de manera estructurada. Este sistema debe facilitar la consulta y el análisis de las pruebas realizadas, las calificaciones obtenidas por los vehículos, la composición detallada

de los vehículos y sus componentes, los certificados de calidad asociados a cada componente, así como la participación de los ingenieros en las distintas pruebas y los roles que han desempeñado. De este modo, EURO-NCAP podrá optimizar sus procesos de evaluación, mejorar la transparencia de sus resultados y contribuir de manera más eficaz a la seguridad vial en Europa.

Se pide realizar un **modelo entidad-relación de Chen** justificando las **cardinalidades mínimas** de al menos dos relaciones (**ambos lados de la relación**).

Solución propuesta:



Cardinalidades mínimas:

- La cardinalidad mínima de vehículo con respecto a prueba es 1, ya que un vehículo no se registra en el sistema hasta ser sometido a alguna prueba. Por otra parte, la cardinalidad mínima de prueba con respecto a vehículo es 0, pues es posible definir una prueba y almacenarla en el sistema antes de haber sometido ningún vehículo a la misma.
- La cardinalidad mínima de componente con respecto a certificado es 0, pues puede haber componentes que no dispongan de ningún certificado. Por otra parte, la cardinalidad mínima de certificado con respecto a componente es 1, pues no puede existir un certificado sin estar asociado a un componente.

EJERCICIO 2. (1½ Puntos) Dado el siguiente modelo relacional[1] que representa la base de datos de las pruebas Euro-NCAP de seguridad de los vehículos ante colisiones:

`VEHICULO (idV, marca, modelo)`

`PRUEBA (idP, tipo, duracion)`

`PUNTUACION (`*`idV`*`FK, `*`idP`*`FK, fecha, calificacion)`

Donde `calificación` tendrá un valor entre 1 y 5 y se refiere a las estrellas obtenidas en la prueba correspondiente.

Se pide dar respuesta con álgebra relacional a las siguientes cuestiones:

(a) (½ Punto) "*Obtener el identificador de aquellos vehículos que hayan pasado con al menos 3 estrellas todas las pruebas que existen*".

Solución propuesta:

$\Pi_{idV,idP}(\sigma_{\text{calificación}>=3}(PUNTUACION)) \div \Pi_{idP}(PRUEBA)$

(b) (½ Punto) "*Obtener el identificador de aquellos vehículos que hayan pasado con al menos 3 estrellas todas las pruebas a las que han sido sometidos*".

Solución propuesta:

$\Pi_{idV}(PUNTUACION) - \Pi_{idV}[\sigma_{\text{calificación}<3}(PUNTUACION)]$

(c) (½ Punto) "*Obtener las marcas que tengan algún vehículo que haya sido sometido tanto a la prueba de tipo 'choque frontal' como a la de tipo 'choque lateral'*".

Solución propuesta:

$\Pi_{marca}[VEHICULO \bowtie ($
$\Pi_{idV}[\sigma_{\text{tipo='choque frontal'}}(PRUEBA \bowtie PUNTUACION)]$
$\cap$
$\Pi_{idV}[\sigma_{\text{tipo='choque lateral'}}(PRUEBA \bowtie PUNTUACION)]$
$)]$

[1] Tenga en cuenta que las claves primarias aparecen subrayadas mientras que las claves foráneas aparece en *cursiva* junto con el superíndice FK y comparten nombre con la clave primaria a la que referencian.

EJERCICIO 3. (4 Puntos) Dado el siguiente modelo relacional[2] ampliado y modificado de la base de datos Euro-NCAP del Ejercicio 2:

FABRICANTE (idF, marca, pais)

PLANTA_PRODUCCION (*idF*^{FK}, numero, capacidad, localizacion)

VEHICULO (idV, modelo, año, *idF*^{FK}, *numero*^{FK})

PRUEBA (idP, tipo, duracion)

PUNTUACION (*idV*^{FK}, *idP*^{FK}, fecha, calificacion)

Se pide dar respuesta a las siguientes cuestiones:

(a) (½ Punto) Resuelva mediante SQL la siguiente consulta: "*obtener todos los datos de los fabricantes que no han fabricado ningún vehículo en plantas de producción ubicadas en Madrid*".

Solución propuesta:

```
SELECT *
FROM FABRICANTE f
WHERE f.idF NOT IN
  (SELECT DISTINCT v.idF
   FROM VEHICULO v
   INNER JOIN PLANTA_PRODUCCION pp
        ON v.idF = pp.idF AND v.numero = pp.numero
   WHERE pp.localizacion = 'Madrid');
```

(b) (½ Punto) Resuelva mediante SQL la siguiente consulta: "*obtener los vehículos (identificador) del año 2023, junto con la marca de su fabricante, y su calificación promedio en pruebas de calidad, de aquellos vehículos con una calificación promedio igual o superior a 4, mostrando los resultados ordenados según la calificación promedio de forma descendente*".

Solución propuesta:

```
SELECT v.idV, f.marca, AVG(p.calificacion) AS
   puntuacion_media
FROM FABRICANTE f
  INNER JOIN VEHICULO v
    ON v.idF = f.idF
  INNER JOIN PUNTUACION p
    ON v.idV = p.idV
WHERE v.año = 2023
GROUP BY v.idV, f.marca
HAVING puntuacion_media >= 4
ORDER BY puntuacion_media DESC;
```

[2]Tenga en cuenta que las claves primarias aparecen subrayadas mientras que las claves foráneas aparece en *cursiva* junto con el superíndice ^{FK} y comparten nombre con la clave primaria a la que referencian.

(c) (½ Punto) Resuelva mediante SQL la siguiente consulta: "*obtener el identificador de la fábrica, su número, y localización de aquellas plantas de producción en las que se ha fabricado algún vehículo al que se le ha realizado todas las pruebas de tipo 'choque lateral'* ".

Solución propuesta:

```
SELECT pp.idF, pp.numero, pp.localizacion
FROM PLANTA_PRODUCCION pp
  INNER JOIN VEHICULO v
    ON v.idF = pp.idF AND v.numero = pp.numero
WHERE v.idV IN (SELECT v.idV
                FROM VEHICULO v
                   INNER JOIN PUNTUACION P
                        ON p.idV = v.idV
                   INNER JOIN PRUEBA pru
                        ON pru.idP = p.idP
                WHERE pru.tipo = 'choque lateral'
                GROUP BY v.idV
                HAVING COUNT(DISTINCT p.idP) = (SELECT COUNT
                    (*)
                                                FROM PRUEBA
                                                WHERE tipo =
                                                    'choque
                                                    lateral'
                                                )
                );
```

(d) (½ Punto) Escriba una sentencia en SQL que cree la tabla VEHICULO tal y como está definida en el enunciado.

Solución propuesta:

```
CREATE TABLE VEHICULO (
    idV INTEGER NOT NULL,
    modelo VARCHAR(128) NOT NULL,
    año INTEGER NOT NULL,
    idF INTEGER NOT NULL,
    numero INTEGER NOT NULL,
    PRIMARY KEY(idV),
    FOREIGN KEY(idF, numero) REFERENCES PLANTA_PRODUCCION(idF
        , numero)
)
```

(e) (½ Punto) Escribir una **FUNCIÓN** denominada `contar_vehiculos` que dados como parámetros el identificador de un fabricante (`idFabricante`) de tipo `VARCHAR(3)` y el año de fabricación, devuelva en un `INTEGER` el número de coches fabricados por el fabricante en el año indicado.

Solución propuesta:

```
DELIMITER $$
CREATE FUNCTION contar_vehiculos (idFabricante VARCHAR(3), añ
   oFabricacion INTEGER)
RETURNS INTEGER
DETERMINISTIC
BEGIN
    DECLARE conteo INTEGER;

    SELECT COUNT(*) INTO conteo
    FROM vehiculo
    WHERE idF = idFabricante AND año = añoFabricacion;

    RETURN conteo;
END$$
DELIMITER ;
```

(f) (1 Punto) Escribe un **PROCEDIMIENTO** denominado `listar_ventas` que liste el desglose anual de vehículos fabricados por los fabricantes de un país específico durante un año determinado. Tanto el país como el año deben proporcionarse como parámetros de entrada de tipos `VARCHAR(50)` e `INTEGER` respectivamente. El listado debe devolverse mediante un parámetro de salida de tipo `VARCHAR(1000)` en el que se indicará, para cada fabricante del país, el nombre del fabricante (`marca`) y el número total de vehículos producidos en ese año. El formato del listado debe ser el siguiente: `MarcaA:23|MarcaB:8|...|MarcaZ:42`. Los fabricantes de este listado deben aparecer ordenados alfabéticamente. Será obligatorio emplear la función `contar_vehiculos` del ejercicio anterior así como el uso de un `CURSOR` para implementar este procedimiento. Adicionalmente, se puede utilizar la función `CONCAT`[3] para concatenar las cadenas que formen el listado de salida.

Solución propuesta:

```
DELIMITER $$
CREATE PROCEDURE listar_ventas (IN pais_fabricacion VARCHAR
    (50), IN año INTEGER, OUT cadena VARCHAR(1000))
BEGIN
    DECLARE marcaFabricante VARCHAR(50);
    DECLARE cantidad INTEGER;
    DECLARE primero INT DEFAULT TRUE;
    DECLARE hecho INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT marca, contar_vehiculos(idF
        , año)
                           FROM vehiculo
                               NATURAL JOIN fabricante
                           WHERE pais = pais_fabricacion
                           ORDER BY marca ASC;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET hecho = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO marcaFabricante, cantidad;
        IF hecho THEN
            LEAVE read_loop;
        END IF;

        IF primero THEN
            SET cadena = CONCAT(marcaFabricante,':',cantidad)
                ;
            primero = FALSE;
        ELSE
            SET cadena = CONCAT(cadena, '|', marcaFabricante,
                ':',cantidad);
        END IF;
    END LOOP;
    CLOSE cur;
END$$
```

[3]La función `CONCAT` permite concatenar varias cadenas de texto. Por ejemplo, `CONCAT('a','b','c')` devuelve la cadena `'abc'`.

```
DELIMITER ;
```

(g) (½ Punto) Se quiere evitar, mediante un **TRIGGER**, la fabricación de nuevos vehículos del fabricante con identificador `F16` en las plantas ubicadas en en Estados Unidos (`localizacion = 'Estados Unidos'`). Codifica dicho TRIGGER.

Solución propuesta:

```
DELIMITER $$
CREATE TRIGGER evitarFabricacionEstadosUnidos BEFORE INSERT
   ON vehiculo
FOR EACH ROW
BEGIN
    DECLARE localizacion_planta VARCHAR(50);

    SELECT localizacion INTO localizacion_planta
    FROM planta_produccion
    WHERE idF = NEW.idF AND numero = NEW.numero;

    IF NEW.idF = 'F16' AND localizacion_planta = 'Estados
       Unidos' THEN
            SIGNAL SQLSTATE '04000'
            SET MESSAGE_TEXT = 'No se permite la fabricacion
               de nuevos vehiculos del fabricante F16 en
               Estados Unidos';
    END IF;
END$$
DELIMITER ;
```

EJERCICIO 4. (2 Puntos) Una clínica veterinaria *SOS Bigotes Urgentes*, con varios centros en España, necesita migrar su sistema de gestión de datos desde un modelo relacional (SQL) a una base de datos NoSQL de documentos (json). Su modelo de datos actual es el que muestra el siguiente modelo relacional[4]:

CLINICA (idClinica, nombre, direccion, telefono)

VETERINARIO (idVeterinario, nombre, especialidad, $idClinica^{FK}$)

MASCOTA (idMascota, nombre, especie, edad, $idCliente^{FK}$)

CLIENTE (idCliente, nombre, telefono, $idClinica^{FK}$)

En una primera migración usa un **modelo embebido** con una única colección con esta estructura:

```
clinicas: [
   {
     "_id": "clin001",
     "nombre": "Clinica Patas Felices",
     "direccion": "Calle del Hueso 42",
     "telefono": "555-123456",
     "veterinarios": [
       {
         "idVeterinario": "vet001", "nombre": "Dr. Lamepatas", "especialidad": "
             Cardiologia"
       },
       {
         "idVeterinario": "vet002", "nombre": "Dra. Ronroneo", "especialidad": "
             Dermatologia Felina"
       }
     ],
     "clientes": [
       {
         "idCliente": "cli001",
         "nombre": "Angeles Mai",
         "telefono": "600-111222",
         "mascotas": [
           {
             "idMascota": "m001", "nombre": "Baghera", "especie": "Gato", "edad": 5
           },
           {
             "idMascota": "m002", "nombre": "PdCar", "especie": "Gato", "edad": 3
           }
         ]
       },
       {
         "idCliente": "cli002",
         "nombre": "Carlos Ruiz",
         "telefono": "600-333444",
         "mascotas": [
           {
             "idMascota": "m003", "nombre": "Blu", "especie": "Perro", "edad": 2
           },
           {
             "idMascota": "m004", "nombre": "Bay", "especie": "Perro", "edad": 4
           }
         ]
       }
     ]
   }
]
```

[4]Tenga en cuenta que las claves primarias aparecen subrayadas mientras que las claves foráneas aparece en *cursiva* junto con el superíndice ^{FK} y comparten nombre con la clave primaria a la que referencian.

(a) (1 Punto) Se pide diseñar la base de datos utilizando un modelo referenciado.

Solución propuesta:

```
// Clinicas
[
    {
      "idClinica": "clin001",
      "nombre": "Clinica Patas Felices",
      "direccion": "Calle del Hueso 42",
      "telefono": "555-123456"
    }
]

// Veterinarios
[
  {
    "idVeterinario": "vet001",
    "nombre": "Dr. Lamepatas",
    "especialidad": "Cardiologia",
    "idClinica": "clin001"
  },
  {
    "idVeterinario": "vet002",
    "nombre": "Dra. Ronroneo",
    "especialidad": "Dermatologia Felina",
    "idClinica": "clin001"
  }
]

// Clientes
[
    {
      "idCliente": "cli001",
      "nombre": "Angeles Mai",
      "telefono": "600-111222",
      "idClinica": "clin001"
    },
    {
      "idCliente": "cli002",
      "nombre": "Carlos Ruiz",
      "telefono": "600-333444",
      "idClinica": "clin001"
    }
]

// Mascostas
[
  {
    "idMascota": "m001",
    "nombre": "Baghera",
    "especie": "Gato",
    "edad": 5,
    "idCliente": "cli001"
  },
```

```
  {
    "idMascota": "m002",
    "nombre": "PdCar",
    "especie": "Gato",
    "edad": 3,
    "idCliente": "cli001"
  },
  {
    "idMascota": "m003",
    "nombre": "Blu",
    "especie": "Perro",
    "edad": 2,
    "idCliente": "cli002"
  },
  {
    "idMascota": "m004",
    "nombre": "Bay",
    "especie": "Perro",
    "edad": 4,
    "idCliente": "cli002"
  }
]
```

(b) (1 Punto) Contestar a las siguientes preguntas, indicando embebido o referenciado, justificando la respuesta:

I. Si la aplicación requiere consultas frecuentes que recuperan toda la información de una clínica (incluyendo veterinarios, clientes y mascotas) en una sola operación, ¿qué modelo es más eficiente?

Solución propuesta:
Embebido. Consultas únicas sin joins.

II. Si se espera que la clínica tenga miles de clientes y mascotas, ¿qué diseño evitaría problemas de tamaño de documento y permitiría un crecimiento sostenible?

Solución propuesta:
Referenciado, escala mejor con datos masivos.

III. Si la especialidad de un veterinario cambia frecuentemente y hay que actualizarla en todos los registros asociados, ¿qué modelo simplificaría este proceso?

Solución propuesta:
Referenciado: Actualización en un solo lugar.

IV. Si un cliente puede llevar sus mascotas a múltiples clínicas, ¿qué modelo manejaría mejor esta relación sin duplicar datos?

Solución propuesta:
Referenciado: Usar referencias evita redundancia.