# UD2 – Examen (Modelo A)

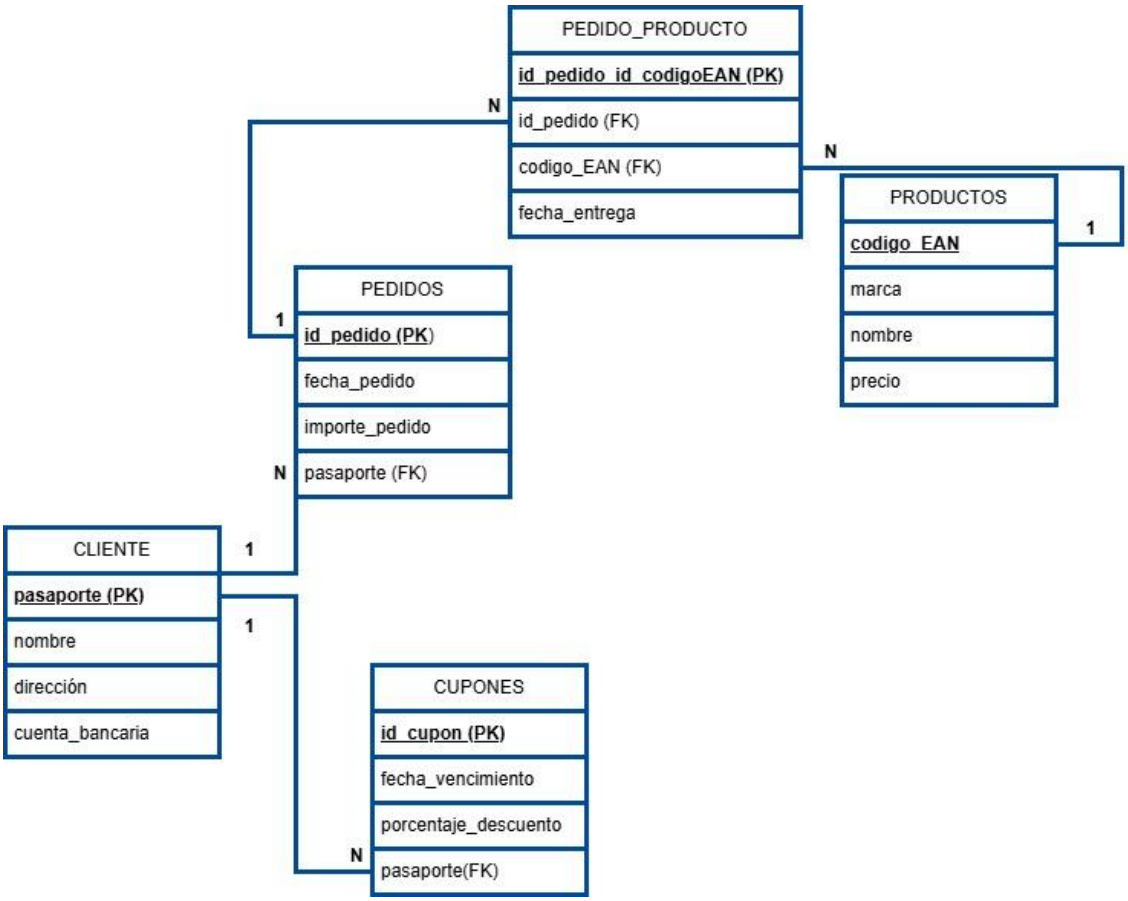
**LEE LAS INSTRUCCIONES ATENTAMENTE. DEBERÁS ENTREGAR ESTE MISMO DOCUMENTO EN FORMATO PDF (CUALQUIER OTRO FORMATO DISTINTO DE PDF NO SERÁ CORREGIDO). SE ESPERA QUE PUEDAS RESOLVER ESTAS ACTIVIDADES EMPLEANDO EXCLUSIVAMENTE TUS CONOCIMIENTOS Y LOS MATERIALES QUE HAYAS DESCARGADO PREVIAMENTE. EN CASO DE CUALQUIER EVIDENCIA DE COPIA, EL EXAMEN SE EVALUARÁ COMO SUSPENSO (0).**

**Pregunta 1.** Considerando las siguientes especificaciones:

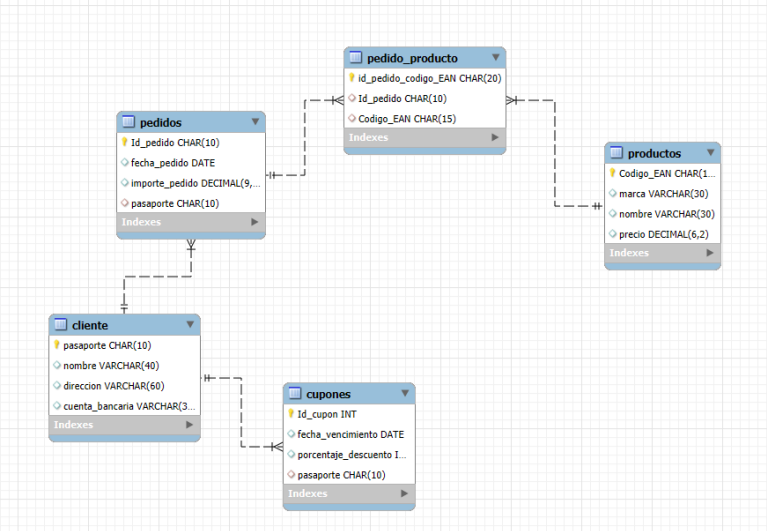
- Los clientes se identifican por su número de pasaporte, nombre, dirección y número de cuenta bancaria.
- Los clientes hacen pedidos. Dado que un cliente puede realizar varios pedidos, es importante conocer la fecha del pedido y el importe total. Un pedido pertenece a un único cliente.
- Cada producto se identifica por su código EAN, marca, nombre y precio. Un pedido puede incluir varios productos diferentes y un producto puede formar parte de varios pedidos distintos.
- El sistema puede crear cupones para los clientes, de modo que cada cliente puede recibir varios cupones. Cada cupón pertenece a un cliente y debe incluir una fecha de vencimiento y un porcentaje de descuento.

Se pide:

1. Un diagrama relacional adecuadamente presentado (no hace falta decir que no olvides incluir todas las tablas, columnas, relaciones, cardinalidades y tipos de datos apropiados). (2 puntos)




2. Tu implementación SQL (que debe ajustarse a tu diagrama, para lo que se pide que muestres el diagrama de ingeniería inversa debidamente ordenado). (2 puntos)



3. Realiza las siguientes modificaciones: (1 punto)
    a. Añade, usando la instrucción ALTER, una columna que distinga a los clientes premium de los clientes normales. (0,25 puntos)

```
CREATE TABLE Cliente (
        pasaporte CHAR(10) PRIMARY KEY,
  nombre VARCHAR (40),
  direccion VARCHAR (60),
  cuenta_bancaria VARCHAR (30)
);
ALTER TABLE Cliente ADD Tipo  ENUM('Premium','Regular');
```

    b. Cambia, utilizando la instrucción ALTER, el tipo de dato del EAN por otro que mantenga la funcionalidad. (0,25 puntos)

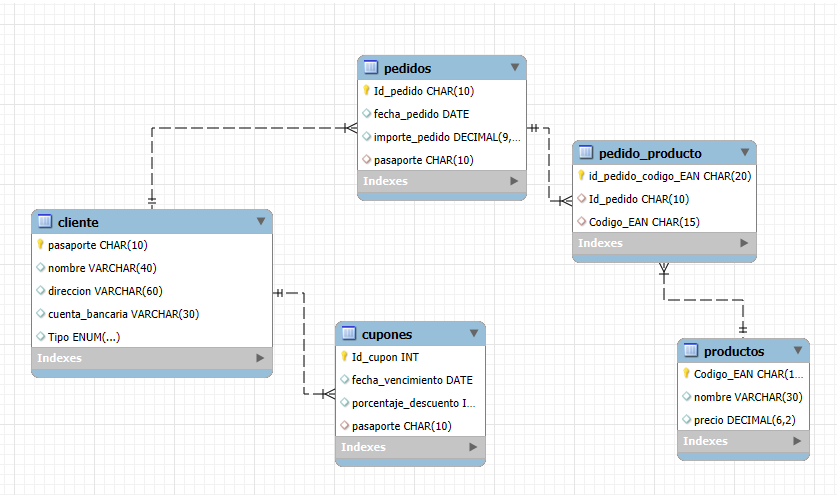
**No recuerdo.**

    c. Realiza las modificaciones oportunas para que cada cliente sólo pueda recibir un único cupón. (0,25 puntos)

**Realmente no sé a qué se refiere. A lo mejor en id_Cupon hacerlo, INT AUTO INCREMENT no se para el código de cupón, de manera que salga solo 1 código único cada vez.**

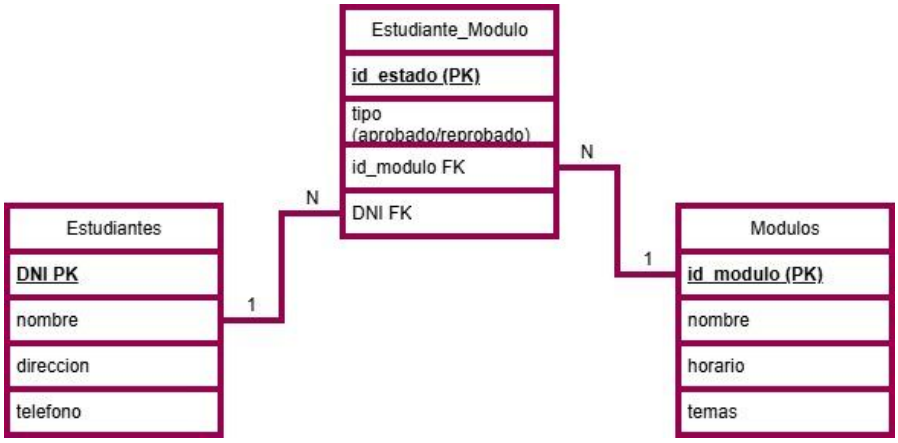
    d. Borra, utilizando la instrucción ALTER, la marca del producto. (0,25 puntos)

```
CREATE TABLE Productos (
Codigo_EAN CHAR(15) PRIMARY KEY,
marca VARCHAR (30),
nombre VARCHAR (30),
precio DECIMAL (6,2)
);
ALTER TABLE Productos DROP COLUMN marca;
```



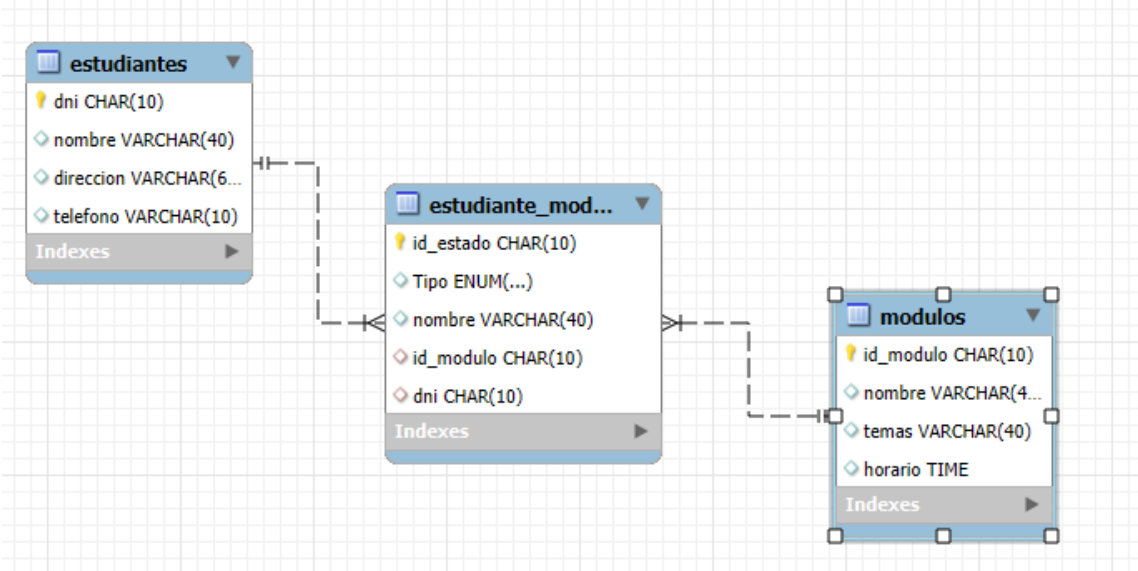
**Pregunta 2.** Considera la relación que existe en Decroly entre los módulos y los estudiantes. Se pide:

1. Una breve implementación en SQL (no menos de 4 columnas para cada tabla) y una captura de pantalla de la ingeniería inversa. (1,5 puntos)




```
DROP DATABASE IF EXISTS decroly;
CREATE DATABASE IF NOT EXISTS decroly;
USE decroly;
CREATE TABLE Estudiantes (
        dni CHAR(10) PRIMARY KEY,
  nombre VARCHAR (40),
  direccion VARCHAR (60),
  telefono VARCHAR (10)
);
CREATE TABLE Modulos (
        id_modulo CHAR(10) PRIMARY KEY,
  nombre VARCHAR (40),
  temas VARCHAR (40),
  horario TIME
);
CREATE TABLE Estudiante_Modulo (
        id_estado CHAR(10) PRIMARY KEY,
  Tipo ENUM('Aprobado','Reprobado'),
  nombre VARCHAR (40),
        id_modulo CHAR(10),
```

```
    dni CHAR(10),
    FOREIGN KEY (id_modulo) REFERENCES Modulos(id_modulo),
    FOREIGN KEY (dni) REFERENCES Estudiantes(dni)
);
```




2. Considera ahora la relación entre las televisiones y las aulas. Proporciona una breve implementación en SQL para la relación (no menos de 4 columnas para cada tabla) y una captura de pantalla de la ingeniería inversa. (1,5 puntos)

**NO ME DIO TIEMPO….**

**Pregunta 3.** Lee atentamente el siguiente código SQL. Hay 4 errores diferentes, resáltalos y explica detalladamente cómo puedes solucionar cada uno de ellos. (2 puntos)

```
DROP DATABASE IF EXISTS examA;
CREATE DATABASE examA;
USE examA;

CREATE TABLE Customers (
   CustomerID INT,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(100),/Esto es un varchar medio exagerado para un email.
```

```
    PhoneNumber VARCHAR(15),
    CONSTRAINT PK_Customers PRIMARY KEY (CustomerID),/En el Primary Key no es necesario poner esto así, con poner Id_Customers CHAR(15) PRIMARY KEY, sería suficiente.
);

CREATE TABLE Products (
    ProductID INT,/ Esto depende, me parece un INT queda muy cerrado, a solo números, cuando a veces un producto puede tener letras dentro de su  ID.
    ProductName VARCHAR(100),
    Price BOOLEAN, Los precios no deberían ser booleanos, con un DECIMAL (8,2) sería mejor.
    Stock INT,
    CONSTRAINT PK_Products PRIMARY KEY (ProductID)
);



CREATE TABLE OrderDetails (
    OrderDetailID INT,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    Price DECIMAL(10,2),
    CONSTRAINT PK_OrderDetails PRIMARY KEY (OrderDetailID),/ Esto no debería declararse así. con poner OrderDetailID CHAR(15) PRIMARY KEY, sería suficiente,
    CONSTRAINT FK_OD_Orders
        FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    CONSTRAINT FK_OD_Products
        FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
CREATE TABLE Orders (
    OrderID INT,
    CustomerID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10,2),
    CONSTRAINT PK_Orders PRIMARY KEY (OrderID),
```

```
    CONSTRAINT FK_Orders_Customers
        FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
CREATE TABLE Payments (
    PaymentID INT,
    OrderID INT,/ esto no se pero que este declarada nuevamente en esta tabla, me parece dará error, esta variable con este tipo ya está declarada en Orders, con lo cual aquí solo podría estar como FK.
    PaymentDate DATE,
    Amount DECIMAL(10,2),
    PaymentMethod VARCHAR(30),
    CONSTRAINT PK_Payments PRIMARY KEY (PaymentID),
    CONSTRAINT FK_Payments_Orders
        FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

CREATE TABLE Shipments (
    ShipmentID INT,
    OrderID INT,
    ShipmentDate DATE,
    Carrier VARCHAR(50),
    TrackingNumber VARCHAR(50),
    CONSTRAINT PK_Shipments PRIMARY KEY (ShipmentID),
    CONSTRAINT FK_Shipments_Orders
        FOREIGN KEY (OrderID) REFERENCES Orders(PaymentID)/ aquí ORDERS ni siquiera tiene dentro un PaymentID.
);
```

Y en general las FOREIGN KEYS tienen más trabajo del que deberían, con algo sencillo como: “FOREIGN KEY (PaymentID) REFERENCES Payments(PaymentID)”, sería suficiente