UD2 – Examen (Modelo A)

ALEJANDRO SAINZ SAINZ

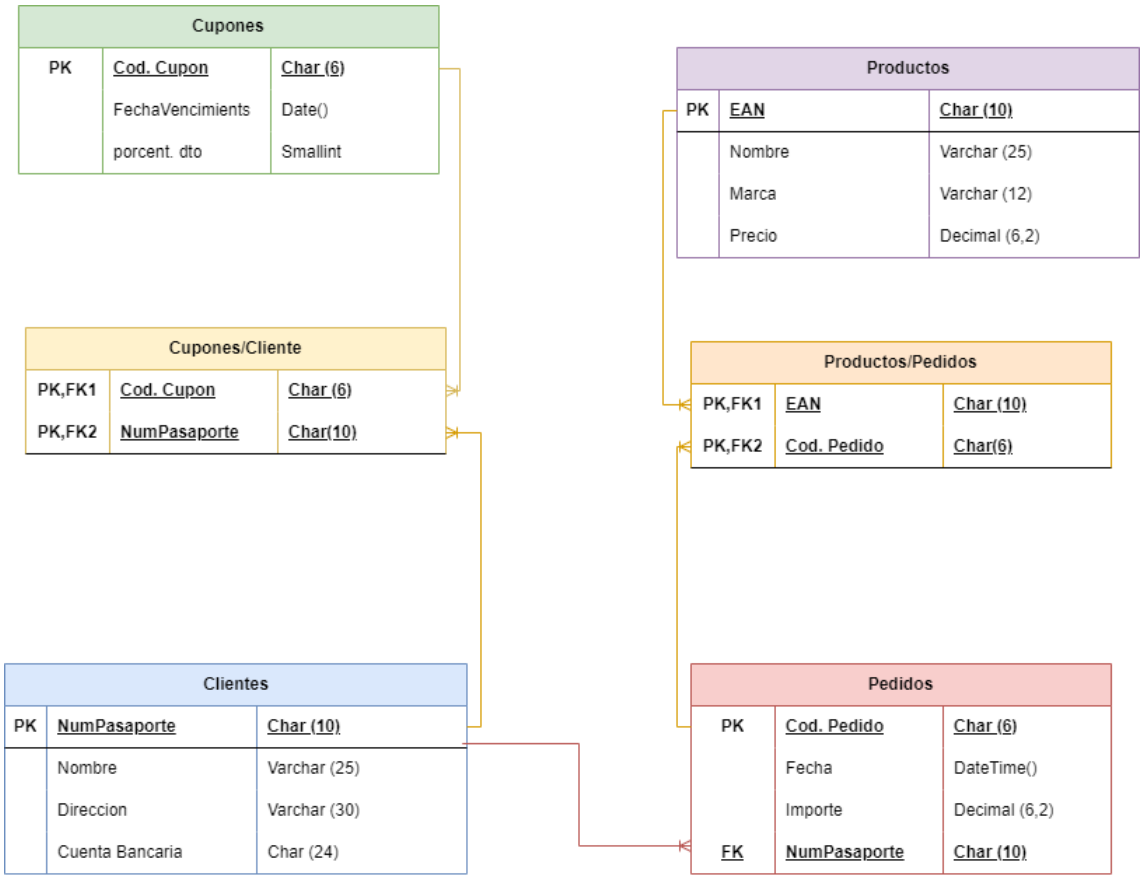
LEE LAS INSTRUCCIONES ATENTAMENTE. DEBERÁS ENTREGAR ESTE MISMO DOCUMENTO EN FORMATO PDF (CUALQUIER OTRO FORMATO DISTINTO DE PDF NO SERÁ CORREGIDO). SE ESPERA QUE PUEDAS RESOLVER ESTAS ACTIVIDADES EMPLEANDO EXCLUSIVAMENTE TUS CONOCIMIENTOS Y LOS MATERIALES QUE HAYAS DESCARGADO PREVIAMENTE. EN CASO DE 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 y dirección, así como por su 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.
- Cada producto se identifica por su código EAN, marca, nombre y precio. Un pedido puede incluir varios productos diferentes.
- El sistema puede crear cupones para los clientes, de modo que cada cliente puede recibir varios cupones. Cada cupón está relacionado con un cliente y debe incluir una fecha de vencimiento y un porcentaje de descuento.

Se pide:

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



Para aclarar algunos de los valores de los campos:

No recuerdo la cantidad de números que tiene un pasaporte pero siempre es fija, por eso la opción de char.

EAN se lo que es, pero no su formato, también he elegido char, pero con una cantidad indicativa.

La mayoría de los códigos, las PK, son char con una longitud determinada, ya que me imagino en este caso que siempre tienen el mismo formato y longitud, sólo que, modificando los valores de números y letras.

En el porcentaje de descuento, pensé en un decimal, pero no se dan este tipo de descuentos, siempre suelen ser números enteros, por eso el Smallint.

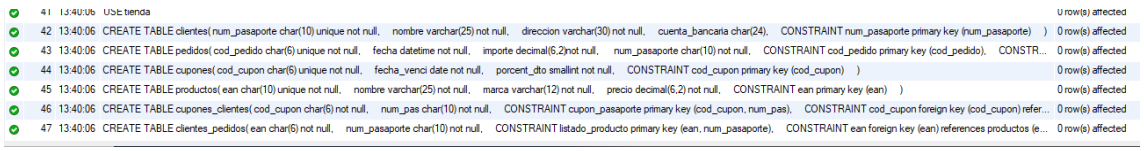
Por lo demás todo es bastante estándar.

2. Tu implementación SQL (que debe ajustarse a tu diagrama relacional). (2 puntos)

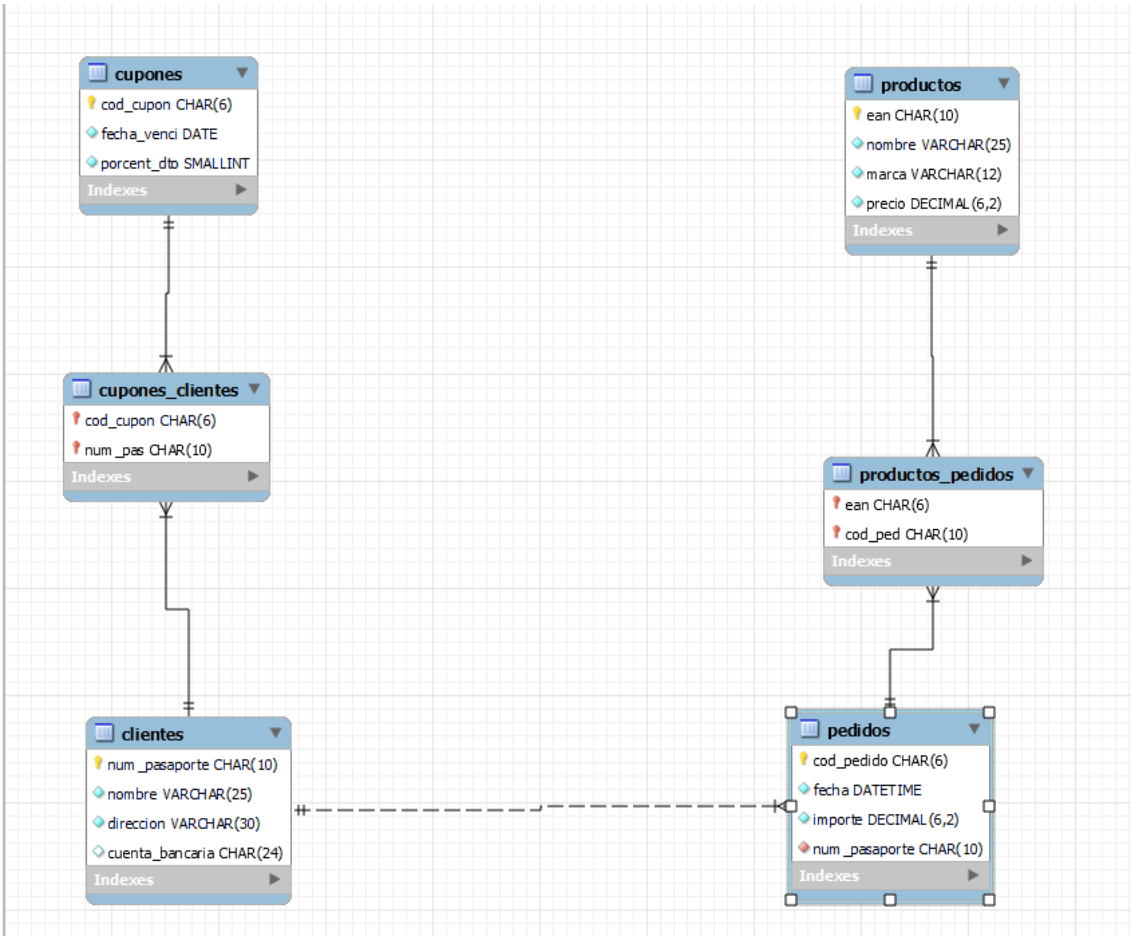
```
DROP DATABASE IF EXISTS tienda;
CREATE DATABASE tienda;
USE tienda;

CREATE TABLE clientes(
        num_pasaporte char(10) unique not null,
   nombre varchar(25) not null,
   direccion varchar(30) not null,
   cuenta_bancaria char(24),
   CONSTRAINT num_pasaporte primary key (num_pasaporte)
   );

CREATE TABLE pedidos(
        cod_pedido char(6) unique not null,
   fecha datetime not null,
   importe decimal(6,2)not null,
   num_pasaporte char(10) not null,
   CONSTRAINT cod_pedido primary key (cod_pedido),
   CONSTRAINT num_pasaporte foreign key (num_pasaporte) references clientes
(num_pasaporte)
   );

CREATE TABLE cupones(
        cod_cupon char(6) unique not null,
   fecha_venci date not null,
   porcent_dto smallint not null,
   CONSTRAINT cod_cupon primary key (cod_cupon)
   );

CREATE TABLE productos(
        ean char(10) unique not null,
   nombre varchar(25) not null,
   marca varchar(12) not null,
```

```
    precio decimal(6,2) not null,
    CONSTRAINT ean primary key (ean)
    );

CREATE TABLE cupones_clientes(
            cod_cupon char(6) not null,
    num_pas char(10) not null,
    CONSTRAINT cupon_pasaporte primary key (cod_cupon, num_pas),
    CONSTRAINT cod_cupon foreign key (cod_cupon) references cupones (cod_cupon),
    CONSTRAINT num_pas foreign key (num_pas) references clientes (num_pasaporte)
    );

CREATE TABLE productos_pedidos(
            ean char(6) not null,
    cod_ped char(10) not null,
    CONSTRAINT listado_producto primary key (ean, cod_ped),
    CONSTRAINT ean foreign key (ean) references productos (ean),
    CONSTRAINT cod_ped foreign key (cod_ped) references pedidos (cod_pedido)
    );
```

Este es todo mi código sql. Incluyo una captura de pantalla de la ejecución del script.



3. Tu diagrama de ingeniería inversa (no olvides que debe coincidir con tu diagrama relacional). (1 punto)



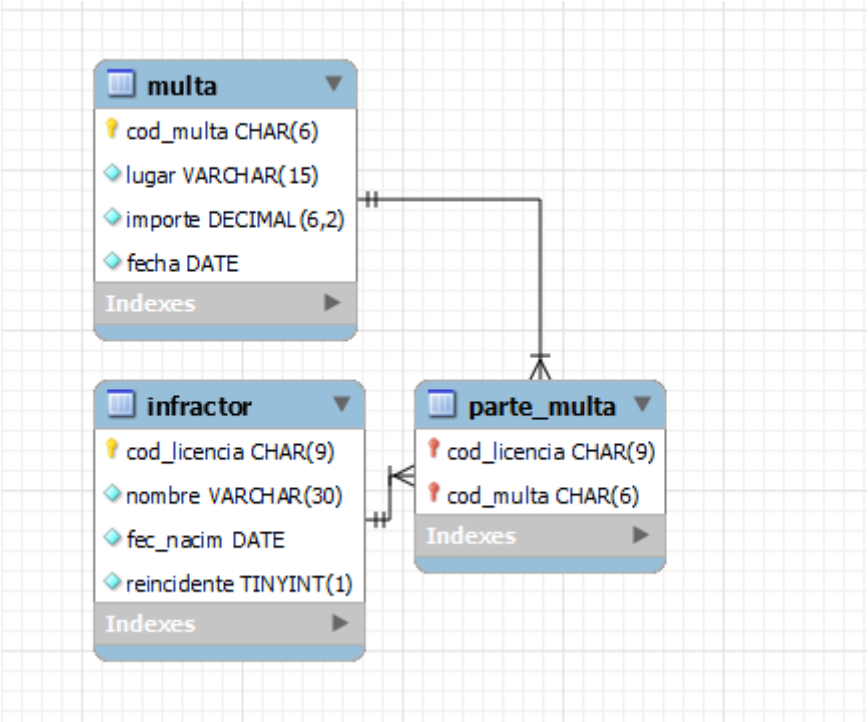
Este es el diagrama que resulta de la ingeniería inversa en el workbench.

Pregunta 2. Piensa en una relación N:N (ejemplo: módulos y estudiantes) y una relación 1:1 (ejemplo: mensajero y furgoneta). Se pide:

1. Una breve implementación de SQL para la relación N:N (no más de 4 columnas para cada tabla) y una captura de pantalla de la ingeniería inversa. (2 puntos)

El ejemplo que voy a usar sería de, gente perteneciente a un gremio o grupo, que reciben una multa por una infracción, un infractor puede tener varias multa, y en una multa puede haber implicado más de un infractor.

```
DROP DATABASE IF EXISTS multas;
CREATE DATABASE multas;
use multas;

CREATE TABLE infractor(
        cod_licencia CHAR(9) unique not null,
  nombre varchar(30) not null,
  fec_nacim date not null,
  reincidente boolean not null,
  CONSTRAINT cod_licencia primary key (cod_licencia)
  );

CREATE TABLE multa(
        cod_multa char(6) unique not null,
  lugar varchar(15) not null,
  importe decimal(6,2) not null,
  fecha date not null,
  CONSTRAINT cod_multa primary key (cod_multa)
  );

CREATE TABLE parte_multa(
        cod_licencia char(9) not null,
  cod_multa char(6) not null,
  CONSTRAINT implicados_multa primary key(cod_licencia,cod_multa),
  CONSTRAINT cod_licencia foreign key (cod_licencia) references infractor(cod_licencia),
  CONSTRAINT cod_multa foreign key (cod_multa) references multa(cod_multa)
        );
```

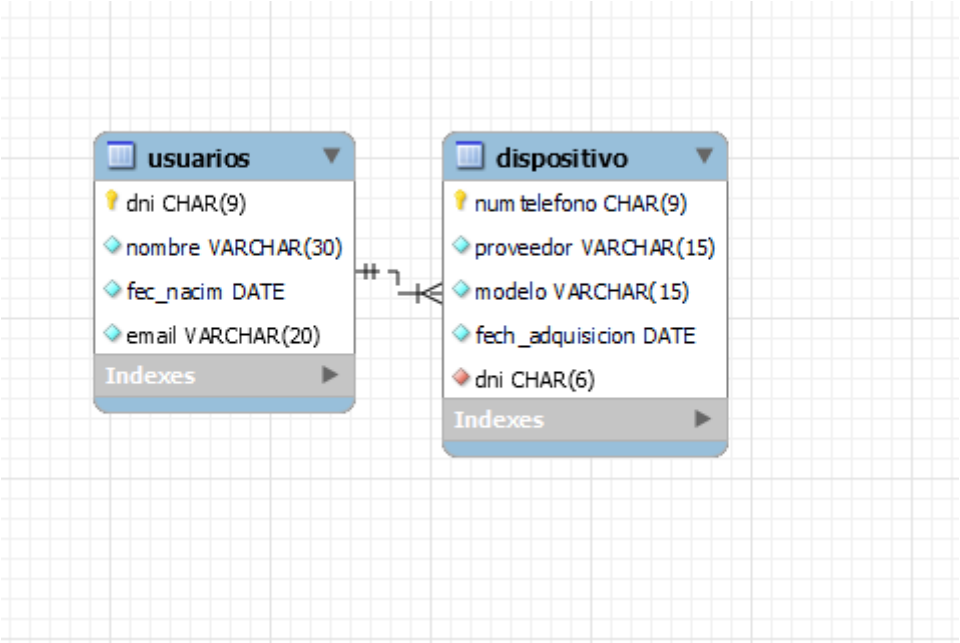


Este sería el diagrama que muestra la ingeniería inversa.

2. Una breve implementación de SQL para la relación 1:1 (no más de 4 columnas para cada tabla) y una captura de pantalla de la ingeniería inversa. (1 punto)

Como ejemplo voy a utilizar una línea de móvil y un usuario

```
DROP DATABASE IF EXISTS lineasmoviles;
CREATE DATABASE lineasmoviles;
use lineasmoviles;

CREATE TABLE usuarios(
        dni CHAR(9) unique not null,
   nombre varchar(30) not null,
   fec_nacim date not null,
   email varchar(20) not null,
   CONSTRAINT dni primary key (dni)
   );

CREATE TABLE dispositivo(
        numtelefono char(9) unique not null,
   proveedor varchar(15) not null,
   modelo varchar(15) not null,
   fech_adquisicion date not null,
   dni CHAR(6) not null,
   CONSTRAINT numtelefono primary key (numtelefono),
   CONSTRAINT dni foreign key(dni) references usuarios(dni)
   );
```



NO PUEDES UTILIZAR LOS EJEMPLOS. EL EJERCICIO NO SERÁ EVALUADO SI LO HACES

Pregunta 3. Lee atentamente el siguiente código SQL. Hay 4 errores diferentes, resáltalos y explica cómo puedes solucionarlos. (2 puntos)

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

//El primer error es la creación de la table orders, ya que necesita una foreign key, y la table a la que hace referencia no esta creada todavia. Esta table debe de ser definida después de crear aquella table de la que es dependiente.

CREATE TABLE Orders (
        OrderID INT,
        CustomerID INT,
        OrderDate DATE,
        TotalAmount DECIMAL(10, 2),
        CONSTRAINT OrderID PRIMARY KEY (OrderID),
        CONSTRAINT CustFK FOREIGN KEY (CustomerID) REFERENCES OrderDetails(CustomerID)
);
```

```
CREATE TABLE Customers (
        CustomerID INT,
        FirstName VARCHAR(50),
        LastName VARCHAR(50),
        Email VARCHAR(100),
        PhoneNumber VARCHAR(15),
        CONSTRAINT CustomerID PRIMARY KEY (CustomerID), //Esa coma no va ahí, dará error. Me ha pasado en este mismo examen, error de sintaxis, no se va a ejecutar bién el script.
);

CREATE TABLE OrderDetails (
        OrderDetailID INT,
        OrderID INT,
        Quantity BOOLEAN,     //No creo que la cantidad deba de ser expresada como un Boolean, debería de ser un int o smallint. No estoy seguro de que cuente como error, el script lo va a generar igual, pero es el que he encontrado por ahora.

        Price DECIMAL(10, 2),
        CONSTRAINT OrderDetailID PRIMARY KEY (OrderDetailID),
        CONSTRAINT OrderID FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
        //Esta última linea no se va a ejecutar, hasta que no arreglemos el primer error. Si la primera table no se crea porque no está definido en el orden correcto, esto no se va a ejecutar.
);
```