

Challenge Engineer - Primera Parte – SQL

Para este challenge diseñé un DER (Diagrama de Entidad – Relación) que responde al modelo del negocio ecommerce solicitado.

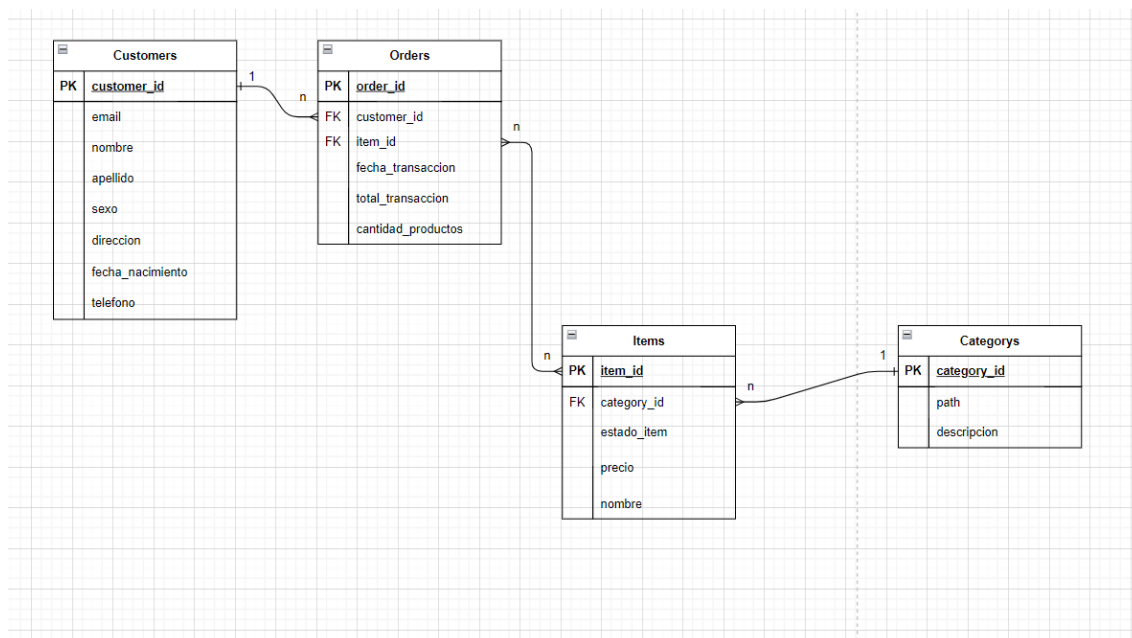
Customers: Representa a los usuarios del sitio de ecommerce. Contiene información personal como nombre, apellido, email, dirección, etc. Cada Customer puede realizar múltiples Orders.

Orders: Refleja las transacciones generadas dentro del sitio. Cada compra realizada por un Customer se registra como una Order. Cada Order contiene información sobre la fecha de la transacción, el Customer que realizó la compra y los Items comprados.

Items: Contiene información sobre los productos publicados en el marketplace. Cada Item está asociado a una Category y puede tener múltiples Orders relacionadas si ha sido comprado por los Customers.

Categorys: Describe las categorías de los Items. Cada Item está asociado a una Category.

El diagrama está realizado en Draw.io



Customers cuenta con una PK que se relaciona con la FK customer_id de Orders.

Items posee una PK que se relaciona con la FK item_id de Orders.

Orders cuenta con su propia PK.

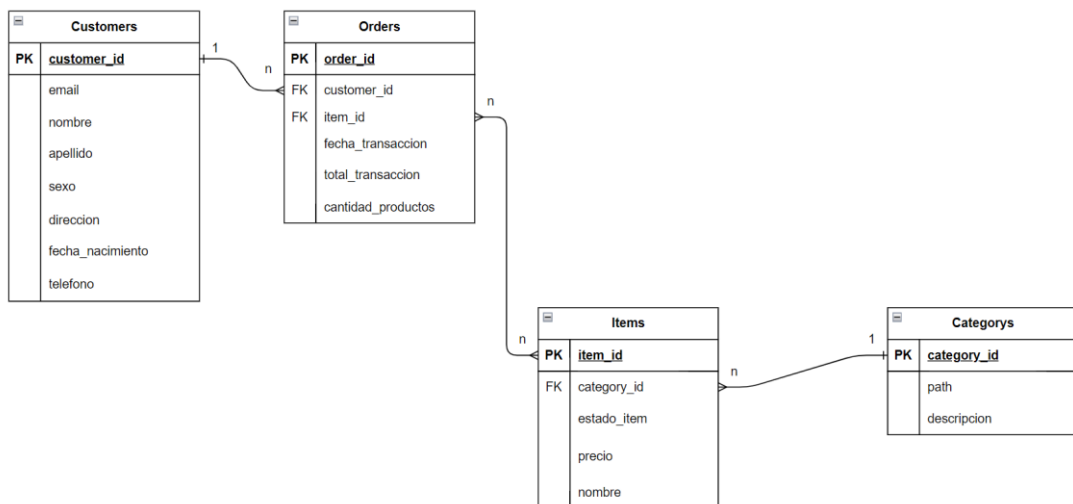
Categorys cuenta con una PK que se relaciona con la FK category_id de Items.

En cuanto a la relación y la cardinalidad de estas 4 entidades destaco lo siguiente.

Customers tiene una relación de 1 a muchos con Orders, ya que un comprador puede hacer múltiples ordenes, pero una orden pertenece a un solo comprador.

Orders e Items tienen una relación de mucho a muchos ya que una orden puede tener varios ítems, y un ítem puede estar en múltiples órdenes.

Item y Category tienen una relación de muchos a 1, ya que un ítem puede tener una sola categoría, pero una categoría puede tener diferentes ítems.



Enlace del drwai.io

X9QKN8%2Fc%2F7CuykG%2Fb38zbQDIPf7MzcEGQIPagfeJx893XZAYymh0XbAr5BZ%2FX
agUJBfXJWsaAfOJ111OKAOAnjPme13qpVIXtG%2FxbGMPssvYt47Xyt6vf67M0bYH9iLsTt
8YSTNZhXAWsuU%2FEKFYUTJyKKn068%2Fsqe%2BX%2Bw1qF36uoPAP4wYUnva5phj6B
a8KI9J78wc%2BVuojDSHUwit2MxWdWcOtbECxphjpHQZO%2FcOdfxjjjmcQs%2B0LNy0Z
XOURBkaYw6vyl7mvIMfUkLYbkeMwsX8gYRI5Pg

Generación del script DDL

Generación del script DDL para la creación de cada una de las tablas representadas en el DER, previamente cree una base de datos ecommerce y la utilicé para realizar ahí la creación de las tablas.

-- SCRIPT DDL para la creación de mi base datos ecommerce en donde voy a trabajar

CREATE DATABASE ecommerce;

-- SCRIPT DDL para usar la base de datos

USE ecommerce;

-- Script DDL para la creación de la tabla Customers

```
CREATE TABLE Customers (  
    customer_id INT PRIMARY KEY,  
    email VARCHAR(100),  
    nombre VARCHAR(50),  
    apellido VARCHAR(50),  
    sexo CHAR(1),  
    direccion VARCHAR(100),  
    fecha_nacimiento DATE,  
    telefono VARCHAR(20)  
);
```

-- Script DDL para la creación de la tabla Categorys

```
CREATE TABLE Categorys (  
    category_id INT PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion VARCHAR(255)  
);
```

-- Script DDL para la creación de la tabla Items

```
CREATE TABLE Items (  
    item_id INT PRIMARY KEY,  
    category_id INT,  
    nombre VARCHAR(100),  
    precio DECIMAL(10, 2),  
    estado VARCHAR(20),  
    FOREIGN KEY (category_id) REFERENCES Categorys(category_id)  
);
```

-- Script DDL para la creación de la tabla Orders

```
CREATE TABLE Orders (  
    order_id INT PRIMARY KEY,  
    customer_id INT,  
    item_id INT,  
    fecha_transaccion DATE,  
    total_transaccion DECIMAL(10, 2),  
    cantidad_productos INT,  
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),  
    FOREIGN KEY (item_id) REFERENCES Items(item_id)  
);
```

Dato importante: Es importante el orden en que creo las entidades y tablas, ya que si ejecutara primero la creación de la tabla Orders por ejemplo, antes de crear la de Customer o Items, marcaría un error, dado a que necesito las otras tablas como referencias para las Foreign Key.

Datos complementarios:

El primer script de create, crea la tabla Customers con las siguientes columnas:

- customer_id: Identificador único entero para cada cliente (clave primaria).
- email: Dirección de correo electrónico del cliente.
- nombre: Nombre del cliente.
- apellido: Apellido del cliente.
- sexo: Sexo del cliente ('M' para masculino, 'F' para femenino).
- direccion: Dirección postal del cliente.
- fecha_nacimiento: Fecha de nacimiento del cliente.
- telefono: Número de teléfono del cliente.

El segundo script de create, crea la tabla Categorys con las siguientes columnas:

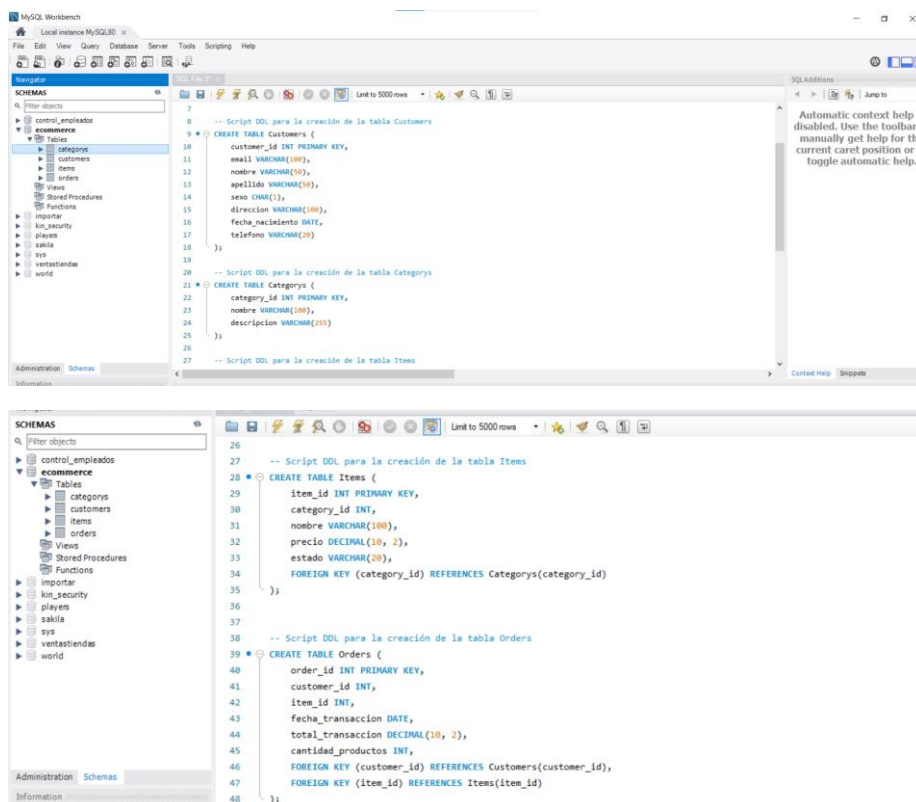
- category_id: Identificador único entero para cada categoría (clave primaria).
- nombre: Nombre de la categoría.
- descripcion: Descripción breve de la categoría.

El tercer script de create, crea la tabla Items con las siguientes columnas:

- item_id: Identificador único entero para cada item (clave primaria).
- category_id: Identificador de la categoría a la que pertenece el item (clave foránea que referencia a la tabla Categorys).
- nombre: Nombre del item.
- precio: Precio del item (en formato decimal con dos decimales).
- estado: Estado del item (por ejemplo, "activo", "inactivo", "descontinuado").
- La restricción de clave foránea FOREIGN KEY (category_id) REFERENCES Categorys(category_id) asegura que la categoría a la que se asocia un item debe existir en la tabla Categorys.

El cuarto script de create, crea la tabla Orders con las siguientes columnas:

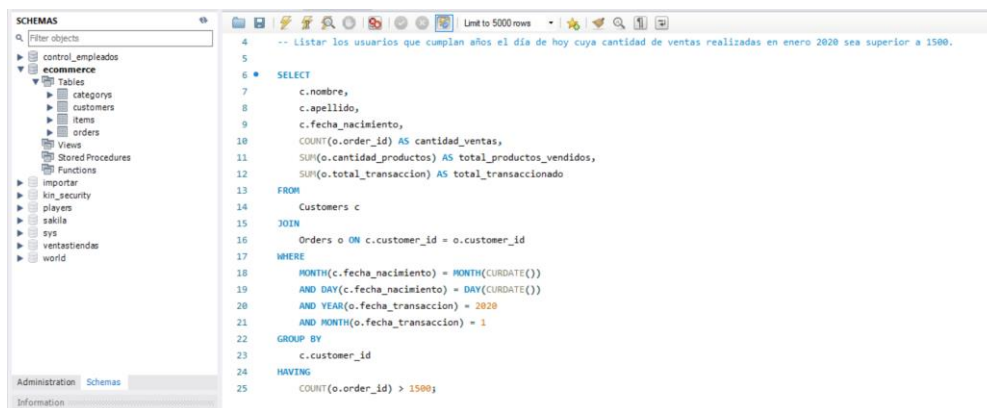
- order_id: Identificador único entero para cada pedido (clave primaria).
- customer_id: Identificador del cliente que realizó el pedido (clave foránea que referencia a la tabla Customer).
- item_id : Identificador del item que contiene la orden (clave foránea que referencia a la tabla Items)
- fecha_transaccion: Fecha de transacción de la orden.
- total_transaccion: Total de la transacción de la orden.
- cantidad_productos: Numero entero que muestra la cantidad de productos que contiene la orden, este atributo lo terminé agregando mas tarde luego de hacer la query para el caso2 ya que era mas eficiente para la consulta que la tabla orders presente la cantidad de productos que contiene una orden.
- La restricción de clave foránea FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) asegura que el comprador al que se asocia una orden debe existir en la tabla Customers.
- La restricción de clave foránea FOREIGN KEY (item_id) REFERENCES Items(item_id) asegura que el item al que se asocia una orden debe existir en la tabla Items.



Casos que resolver

1. Listar los usuarios que cumplan años el día de hoy cuya cantidad de ventas realizadas en enero 2020 sea superior a 1500.
2. Por cada mes del 2020, se solicita el top 5 de usuarios que más vendieron (\$) en la categoría Celulares. Se requiere el mes y año de análisis, nombre y apellido del vendedor, cantidad de ventas realizadas, cantidad de productos vendidos y el monto total transaccionado.
3. Se solicita poblar una nueva tabla con el precio y estado de los ítems a fin del día. Tener en cuenta que debe ser reprocesable. Vale resaltar que en la tabla Item, vamos a tener únicamente el último estado informado por la PK definida. *(Se puede resolver a través de StoredProcedure)*

Caso 1.



Selecciono las columnas que se mostrarán en los resultados. En este caso, selecciono el nombre, apellido y fecha de nacimiento del cliente con tres agregaciones, que son la cantidad de ventas realizadas por cada cliente (COUNT(o.order_id)), la cantidad total de productos vendidos (SUM(o.cantidad_productos)) y el monto total transaccionado por cada cliente de los productos vendidos en cada orden (SUM(o.total_transaccion)).

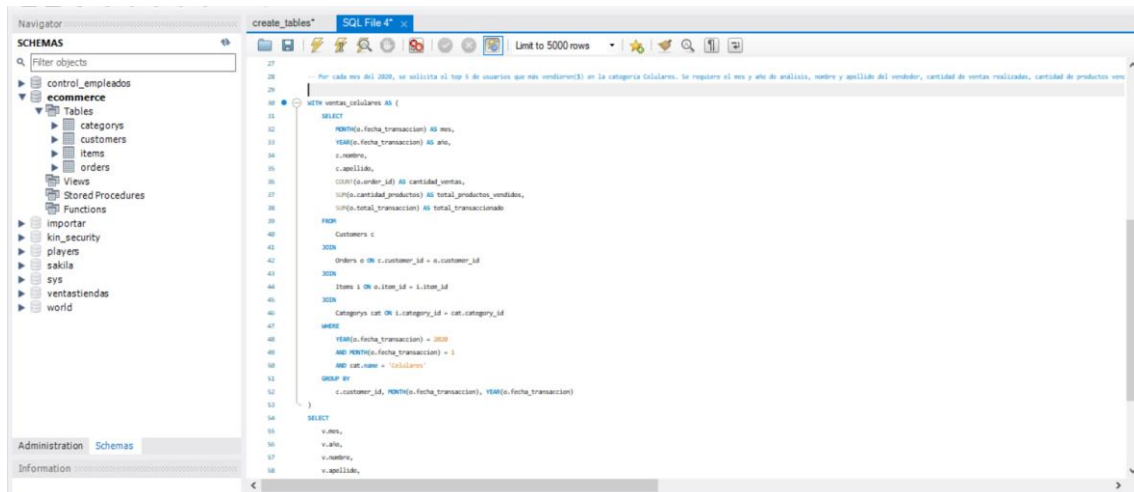
Especifico las tablas que se utilizarán en la consulta y pongo las condiciones de unión (joins). En este caso, utilizo las tablas Customers y Orders. Realizo la unión de las tablas según las claves primarias y foráneas.

Filtro los registros usando la condición WHERE para encontrar clientes cuya fecha de nacimiento coincida con el día y el mes actuales, también el año coincida con el 2020, por eso uso YEAR, DAY y MONTH para que coincidan con los datos que quiero comparar. También uso la función CURDATE para representar la fecha actual del sistema.

Utilizo GROUP BY para agrupar los resultados por el identificador único del cliente.

Utilizo HAVING COUNT(o.order_id) > 1500 para aplicar un filtro adicional y seleccionar solo a los clientes que hicieron más de 1500 órdenes.

Caso 2.



Utilizo la cláusula WITH para definir una subconsulta llamada ventas_celulares.

Dentro de esta subconsulta, selecciono el mes de la fecha de transacción (utilizando la función MONTH), el año de la fecha de transacción (utilizando la función YEAR), el nombre y apellido del cliente, la cantidad de órdenes realizadas por el cliente (utilizando la función COUNT), la suma de la cantidad de productos vendidos en todas las órdenes (utilizando la función SUM), la suma del monto total de todas las órdenes (utilizando la función SUM).

Hago un JOIN entre las tablas Customers, Orders, Items y Categorys para obtener información sobre las órdenes relacionadas con la categoría "Celulares".

Coloco el WHERE para filtrar las órdenes realizadas con las condiciones requeridas como el año 2020 y en el mes 1 (enero).

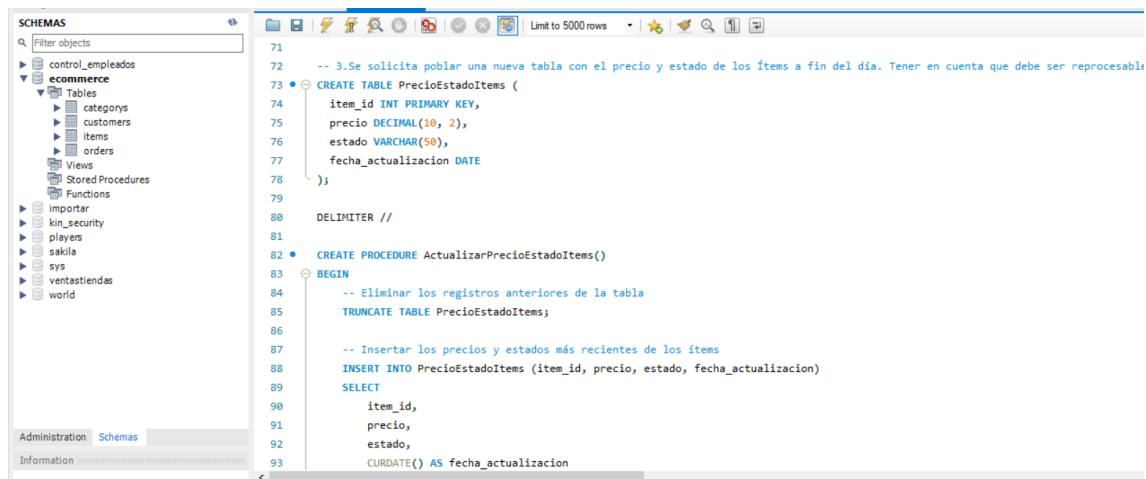
Agrupo los resultados por customer_id, mes y año.

Después realizo la consulta principal, en donde selecciono todas las columnas de la subconsulta ventas_celulares.

Ordeno los resultados por año, mes y total transaccionado en orden descendente.

Finalmente limito los resultados a los primeros 5 registros.

Caso 3.



The screenshot shows a MySQL IDE window with a SQL script. On the left, the 'SCHEMAS' panel shows a tree view with 'ecommerce' selected. The main editor displays the following SQL code:

```
71
72 -- 3.Se solicita poblar una nueva tabla con el precio y estado de los ítems a fin del día. Tener en cuenta que debe ser reprocesable
73 CREATE TABLE PrecioEstadoItems (
74     item_id INT PRIMARY KEY,
75     precio DECIMAL(10, 2),
76     estado VARCHAR(50),
77     fecha_actualizacion DATE
78 );
79
80 DELIMITER //
81
82 CREATE PROCEDURE ActualizarPrecioEstadoItems()
83 BEGIN
84     -- Eliminar los registros anteriores de la tabla
85     TRUNCATE TABLE PrecioEstadoItems;
86
87     -- Insertar los precios y estados más recientes de los ítems
88     INSERT INTO PrecioEstadoItems (item_id, precio, estado, fecha_actualizacion)
89     SELECT
90         item_id,
91         precio,
92         estado,
93         CURDATE() AS fecha_actualizacion
```

Con esta query creo una tabla llamada PrecioEstadoItems para almacenar los precios y estados más recientes de los ítems, y define un procedimiento almacenado llamado ActualizarPrecioEstadoItems que actualiza esta tabla con los datos más recientes de la tabla Items.

Utilizo el comando CREATE TABLE para crear una nueva tabla llamada PrecioEstadoItems.

La tabla tiene cuatro columnas: item_id, precio, estado y fecha_actualizacion.

La columna item_id se define como la clave primaria de la tabla utilizando PRIMARY KEY.

Las columnas precio y estado son del tipo DECIMAL (10, 2) y VARCHAR(50) respectivamente.

La columna fecha_actualizacion es del tipo DATE.

Previo a crear el procedure ActualizarPrecioEstadoItems, utilizo DELIMITER para cambiar el delimitador de comandos de ";" a "//". Esto lo vi necesario colocar porque MySQL interpreta cada ; como el final del procedimiento almacenado o la función, en lugar de como el final de una sentencia SQL individual dentro del procedimiento, entonces generaba algunos errores si no cambiaba el delimitador.

Defino el procedimiento almacenado ActualizarPrecioEstadoItems utilizando CREATE PROCEDURE.

Dentro del procedimiento almacenado, ejecuto las siguientes acciones:

Utilizo TRUNCATE TABLE para eliminar todos los registros existentes de la tabla PrecioEstadoItems.

Luego, realizo una inserción en la tabla PrecioEstadoItems utilizando INSERT INTO.

Los valores por insertar se obtienen de la tabla Items mediante una consulta SELECT.

La subconsulta SELECT busca los precios y estados más recientes de cada ítem.

Utilizo CURDATE () para obtener la fecha actual como fecha_actualizacion.

La condición (item_id, fecha_actualizacion) IN la utilizo para seleccionar solo los registros más recientes de cada ítem.

Finalmente restauro el delimitador original utilizando DELIMITER;