UNIVERSIDAD AUTONOMA METROPOLITANA

UNIDAD CUAJIMALPA

Base de Datos

Practica Final

Tienda\_online



**DR. GUILLERMO MONROY RODRIGUEZ**

*EMILIANO YEI CUICAME OLMEDO ANGELES*



Introducción

Se desarrolla una base de datos para una tienda online que está estructurada por Clientes, Productos, Pedidos, Categorías, Detalles del pedido y Reseñas. Cumpliendo con todos los requisitos que se necesitan para crearla y además restricciones para mejorar la eficiencia, como el máximo de 5 pedidos por cliente o reseñas de clientes que hayan comprado el producto.

Objetivos

* Implementar una base de datos funcional y congruente con pruebas realistas
* Implementar índices
* Implementar consultas y procedimientos almacenados para una mejor eficiencia en la base de datos
* Pruebas de funcionamiento

Diagrama

El contenido generado por IA puede ser incorrecto.

Justificación de la normalización

Primera forma normal o 1NF

* Nada se repite
* Cada tabla cuenta con primary key

Segunda forma normal o 2NF

* Cumple con la primera forma normal
* Las columnas dependen de una clave primaria

Tercera forma normal o 3NF

* Cumple con la segunda formal normal cualquier atributo describe
* Con todos sus atributos describe la clave primaria de su tabla

Identificación de claves Primarias, Foraneas y candidatas

1. Tabla clientes

Clave primaria: id\_cliente

Clave candidata : Correo (identifica al comprador)

1. Tabla categorías

Clave primaria: id\_categoria

Sin foránea ni candidata

1. Tabla Productos

Clave primaria: id\_producto

Clave foránea: id\_categoria

1. Tabla Pedidos

Clave primaria: id\_pedido

Clave foranea: id\_cliente

1. Tabla Detalles\_pedido

Clave primaria: id\_detalle

Clave foranea: id\_producto

id\_pedido

Clave candidata: id\_pedido + id\_producto

1. Tabla resenias

Clave primaria: id\_resenia

Clave foranea: id\_cliente

id\_producto

Clave candidata: id\_cliente + id\_producto

Scrips SQL

CREATE DATABASE Tienda\_online;

USE Tienda\_online;

CREATE TABLE Clientes (

id\_cliente INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

nombre VARCHAR(30),

correo VARCHAR(30) UNIQUE NOT NULL,

telefono VARCHAR (15),

direccion VARCHAR(100)

);

CREATE TABLE Categorias (

id\_categoria INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

nombre VARCHAR(30),

descripcion VARCHAR(100)

);

CREATE TABLE Productos (

id\_producto INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

nombre VARCHAR(30),

descripcion VARCHAR(100),

precio FLOAT NOT NULL DEFAULT 0,

stock INT NOT NULL CHECK (stock >= 0),

id\_categoria INT NOT NULL,

FOREIGN KEY (id\_categoria) REFERENCES Categorias(id\_categoria)

);

CREATE TABLE Pedidos (

id\_pedido INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

fecha\_pedido DATE,

estado VARCHAR(50),

id\_cliente INT NOT NULL,

FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente)

);

CREATE TABLE Detalles\_pedido (

id\_detalle INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

cantidad INT NOT NULL CHECK (cantidad >= 0),

precio\_unitario FLOAT,

id\_pedido INT NOT NULL,

id\_producto INT NOT NULL,

FOREIGN KEY (id\_pedido) REFERENCES Pedidos(id\_pedido),

FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto)

);

CREATE TABLE Resenias (

id\_resenia INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

calificacion INT CHECK (calificacion BETWEEN 1 AND 5),

comentario VARCHAR (1000),

fecha DATE,

id\_cliente INT NOT NULL,

id\_producto INT NOT NULL,

FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente),

FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto)

);

INDICES

USE Tienda\_online;

CREATE INDEX idx\_producto\_nombre ON productos(nombre);

CREATE INDEX idx\_producto\_categoria ON productos(id\_categoria);

CREATE INDEX idx\_pedidos\_cliente ON pedidos(id\_cliente);

CREATE INDEX idx\_resenias\_id\_producto ON Resenias(id\_producto);

Consultas

USE Tienda\_online;

-- Lista de productos disponibles por categoria y ordenados por precio

SELECT c.nombre AS categoria, -- toma el nombre de la categoria desde la tabla categorias

p.nombre AS producto, -- toma el nombre del producto desde la tabla productos

p.precio, -- accede directamente al precio

p.stock -- acede directamente al stock

FROM Productos p -- tabla productos por referencia p

INNER JOIN Categorias c ON p.id\_categoria = c.id\_categoria -- une tablas producto con categorias

WHERE p.stock > 0 -- sentencia que muestra stock disponible osea mas de 0

ORDER BY c.nombre, p.precio ASC; -- ordena el nombre de la categoria y el precio del producto de menor a mayor

Consulta 2

USE Tienda\_online;

-- muestra clientes con productos pendientes y total de compras

SELECT c.nombre AS cliente, -- toma el nombre cliente desde la tabla clientes

c.correo, -- accede al correo

c.telefono, -- accede al telefono

sum(dp.cantidad \* dp.precio\_unitario) AS total\_pendiente -- suma el subtotal del cliente y al final se suma para que de el total

FROM Clientes c -- tabla clientes por referencia c

INNER JOIN Pedidos p ON c.id\_cliente = p.id\_cliente -- une las dos tablas para clientes con al menos un pedido

INNER JOIN Detalles\_pedido dp ON p.id\_pedido = dp.id\_pedido -- une las dos tablas que accede a la cantidad y precio del pedido

WHERE p.estado = 'pendiente' -- sentencia que solo tomara encuenta los que se encuentren pendiente

GROUP BY c.id\_cliente, c.nombre, c.correo, c.telefono -- agrupa los clientes

ORDER BY total\_pendiente DESC; -- ordena los pendientes y el total de mayor a menor

Consulta 3

USE Tienda\_online;

-- 5 productos con mejor calificacion y resenias

SELECT p.nombre AS producto, -- selecciona el nombre del producto de la tabla productos

AVG(r.calificacion) AS promedio\_calificacion, -- calcula la calificacion de las resenias y el promedio de los productos

COUNT(r.id\_resenia) AS total\_resenias -- muestra las resenias de cada producto

FROM Productos p -- toma productos con referencia p

INNER JOIN Resenias r ON p.id\_producto = r.id\_producto -- une las tablas del id producto y resenias

GROUP BY p.id\_producto, p.nombre -- agrupa los productos

ORDER BY promedio\_calificacion DESC -- ordena los productos de mejor al peor

LIMIT 5; -- muestra solo los primeros 5 productos segun el promedio

Procedimientos

USE Tienda\_online;

-- stored procedure 1

DELIMITER $$

CREATE PROCEDURE RegistraPedido(

IN cliente\_id INT,

IN producto\_id INT,

IN cantidad INT,

IN precio\_unitario FLOAT

)

BEGIN

DECLARE pendientes INT; -- declaro nueva variable para los pedidos pendientes

DECLARE stockDisponible INT; -- nueva variable para stock disponible

SELECT COUNT(\*) INTO pendientes -- cuenta pedidos pendientes del cliente

FROM Pedidos -- accede a tabla pedidos

WHERE id\_cliente = cliente\_id AND estado = 'pendiente'; -- guarda los pendientes del cliente

SELECT stock INTO stockDisponible -- selecciona el stock actual y lo guarda a stockDisponible

FROM Productos -- accede a Productos

WHERE id\_producto = producto\_id; -- actualiza el nuevo producto id

-- realice estas dos condiciones para que si el cliente tiene mas de 5 pendientes, mande error de que tiene mas

-- pero si tiene menos de 5 pendientes salta la primera condicion, y si no hay stock suficiente tambien manda

-- error de que ya no hay stock, finalmente

-- si cumple las 2 condiciones ya crea el nuevo pedido

IF

pendientes >= 5 THEN -- condicion para que tenga menos de 5 pendientes

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'Ya cuenta con 5 pedidos pendientes '; -- imprime el mensaje

ELSEIF

stockDisponible < cantidad THEN -- condicion para saber si ya no hay stock

SIGNAL SQLSTATE '45000'-- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'Stock agotado';

ELSE

INSERT INTO Pedidos (fecha\_pedido, estado, id\_cliente) -- registra el pedido

VALUES (CURDATE(), 'pendiente', cliente\_id); -- devuelve la fecha

SET @nuevoPedido\_id = last\_insert\_id(); -- genera un nuevo id

INSERT INTO Detalles\_pedido (cantidad, precio\_unitario, id\_pedido, id\_producto) -- inserta el detalle del pedido

VALUES (cantidad, precio\_unitario, @nuevoPedido\_id, producto\_id); -- agrega el nuevo id

UPDATE Productos

SET stock = stock - cantidad -- actualiza el nuevo stock

WHERE id\_producto = producto\_id; -- se establece la nueva cantidad

END IF;

END $$

Procedimiento 2

USE Tienda\_online;

-- stored procedure 2

DELIMITER $$

CREATE PROCEDURE RegistrarResenia(

IN cliente\_id INT,

IN producto\_id INT,

IN calificacion INT,

IN comentario VARCHAR(1000)

)

BEGIN

DECLARE compras INT; -- variable local para guardar las compras del cliente

SELECT COUNT(\*) INTO compras -- cuenta los pedidos del cliente

FROM Pedidos P -- accede a pedidos con referencia p

INNER JOIN Detalles\_pedido DP ON P.id\_pedido = DP.id\_pedido -- une las tablas id del pedido con detalles del pedido

WHERE P.id\_cliente = cliente\_id AND DP.id\_producto = producto\_id; -- verifica que se haya comprado el producto

IF

compras = 0 THEN -- si no hay compras, manda el error

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'No puede dar ninuguna resenia, debido a que no ha realizado su compra'; -- imprime el texto

ELSE

INSERT INTO Resenias (calificacion, comentario, fecha, id\_cliente, id\_producto) -- si tiene compras, puede dejar su resenia

VALUES (calificacion, comentario, CURDATE(), cliente\_id, producto\_id); -- se usa curdate para guardar la nueva fecha

END IF;

END $$

Procedimiento 3

USE Tienda\_online;

-- stored procedure 3

DELIMITER $$

CREATE PROCEDURE ActualizarStock(

IN producto\_id INT,

IN cantidad INT

)

BEGIN

DECLARE stock\_actual INT; -- variable local que guarda el nuevo stock

SELECT stock INTO stock\_actual -- consulta el stock y se guarda

FROM Productos -- accede a productos

WHERE id\_producto = producto\_id; -- actualiza el id

IF

stock\_actual < cantidad THEN -- disminuimos la cantidad de stock que se quiere tomar, y si no es suficiente manda el error

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'Stock agotado';

ELSE

UPDATE Productos -- si todavia hay stock, se actualiza restando la cantidad vendida

SET stock = stock - cantidad

WHERE id\_producto = producto\_id; -- el producto tiene que coincidir con el id

END IF;

END $$

Procedimiento 4

USE Tienda\_online;

-- stored procedure 4

DELIMITER $$

CREATE PROCEDURE CambiarEstado(

IN pedido\_id INT,

IN nuevo\_estado VARCHAR(50)

)

BEGIN

DECLARE existe INT; -- variable local para guardar la existencia

SELECT COUNT(\*) INTO existe -- cuenta cuantos pedidos existen con este id

FROM Pedidos

WHERE id\_pedido = pedido\_id;

IF

existe = 0 THEN -- si es cero, no hay ningun pedido con este id

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'El pedido no existe';

ELSE

UPDATE Pedidos -- si es mayor a cero, significa que existe

SET estado = nuevo\_estado -- actualiza el valor del pedido

WHERE id\_pedido = pedido\_id;

END IF;

END $$

Procedimiento 5

SE Tienda\_online;

-- stored procedure 5

DELIMITER $$

CREATE PROCEDURE EliminaActualiza(

IN p\_id\_producto INT,

OUT promedio FLOAT

)

BEGIN

DELETE

FROM Resenias -- elimina resenias que tengan el mismo id

WHERE id\_producto = p\_id\_producto;

SELECT IFNULL(AVG(calificacion), 0) -- calcula el promedio de calificaciones de las resenias

INTO promedio -- si np hay resenias devuelve 0

FROM Resenias -- accede a la tabla resenias

WHERE id\_producto = p\_id\_producto;

END $$

Procedimiento 6

USE Tienda\_online;

-- stored procedure 6

DELIMITER $$

CREATE PROCEDURE AgregarProducto(

IN p\_nombre VARCHAR(30),

IN p\_descripcion VARCHAR(100),

IN p\_precio FLOAT,

IN p\_stock INT,

IN p\_id\_categoria INT

)

BEGIN

DECLARE existe INT; -- Variable local para saber si el producto existe

SELECT COUNT(\*) INTO existe -- busca si hay productos con el mismo nombre y categoria

FROM Productos -- accede a productos

WHERE nombre = p\_nombre AND id\_categoria = p\_id\_categoria;

IF

existe > 0 THEN -- si existe lanza un errror

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'El producto ya existe ';

ELSE

INSERT INTO Productos(nombre, descripcion, precio, stock, id\_categoria) -- inserta el nuevo producto

VALUES (p\_nombre, p\_descripcion, p\_precio, p\_stock, p\_id\_categoria); -- intercambia valores

END IF;

END $$

Procedimiento 7

USE Tienda\_online;

-- stored procedure 7

DELIMITER $$

CREATE PROCEDURE ActualizarCliente(

IN p\_id\_cliente INT,

IN p\_nuevo\_telefono INT,

IN p\_nueva\_direccion VARCHAR(100)

)

BEGIN

DECLARE existe INT; -- variable para guardar si el cliente existe

SELECT COUNT(\*) INTO existe -- hace un conteo en clientes y si existe lo guarda

FROM Clientes -- aaccede a tabla clientes

WHERE id\_cliente = p\_id\_cliente;

IF

existe = 0 THEN -- si no existe manda el error

SIGNAL SQLSTATE '45000' -- genera el estado de error, el numero es necesario que el cliente sepa el error

SET MESSAGE\_TEXT = 'el cliente que busca, no existe';

ELSE

IF p\_nuevo\_telefono IS NOT NULL THEN -- si el telefono no es null se actualiza

UPDATE Clientes

SET telefono = p\_nuevo\_telefono -- se actualiza el nuevo telefono

WHERE id\_cliente = p\_id\_cliente;

END IF;

IF p\_nueva\_direccion IS NOT NULL THEN -- si la direccion no es null se actualiza

UPDATE Clientes

SET direccion = p\_nueva\_direccion -- la dirrecion se actualiza

WHERE id\_cliente = p\_id\_cliente;

END IF;

END IF;

END $$

Procedimiento 8

USE Tienda\_online;

-- stored procedure 8

DELIMITER $$

CREATE PROCEDURE ReporteBajo()

BEGIN

SELECT -- lista del producto

p.id\_producto,

p.nombre AS nombre\_producto,

p.descripcion,

p.stock,

c.nombre AS categoria -- nombre de la categoria desde la categoria

FROM Productos p -- accede a productos con representacion p

INNER JOIN Categorias c ON p.id\_categoria = c.id\_categoria -- une tablas con categoria para que de el nombre

WHERE p.stock < 5 -- sentencia con menos de 5 productos en stock

ORDER BY p.stock ASC; -- ordena productos de menor a mayor stock

END $$

Conclusiones

La base de datos se implemento como se esperaba cumpliendo con todos los requisitos y restricciones dadas, se aprendió a organizar y diseñar la base de datos por las formas normales para evitar que se repitieran datos o que todas las tablas dependieran de una clave primaria.

El mayor reto fue implementar los Stored Procedures, ya que era ejecutar a base de prueba y error, sin embargo, al final se implementaron las 8 necesarias, de lo cual agiliza la consulta por medio de los índices.