Universidad Tecnológica De Panamá
Facultad de Ingeniería en Sistemas Computacionales
2do Semestre

Profesor: Ronald Ponce

Integrantes: Jean Meléndez 8-985-955

Kevin Valdés 8-1021-301

Daniel Gonzales 8-1022-1099

Martin Liao 1-757-1706

Ricardo Rose

Asignatura: Base de datos

Laboratorio: #1

Año Lectivo: 2024

Introducción:

En la era digital actual, el comercio electrónico ha revolucionado la forma en que los consumidores adquieren productos y servicios. Para gestionar de manera efectiva las operaciones de venta en línea, es esencial contar con un sistema de base de datos bien estructurado que no solo facilite el almacenamiento de información, sino que también garantice la integridad y la accesibilidad de los datos. Este trabajo se centra en el diseño y la implementación de una base de datos para un sistema de ventas en línea, donde se aborda la creación de diversas tablas que representan entidades clave, como productos, usuarios, órdenes y proveedores.

A lo largo de este documento, se explorará la justificación de la existencia de cada tabla y su relevancia dentro del ecosistema de ventas. También se examinarán las relaciones entre las diferentes entidades y cómo estas interacciones contribuyen a la eficiencia operativa del sistema. Además, se discutirá la elección de tipos de datos adecuados para garantizar la precisión y la integridad de la información almacenada, así como la lógica detrás de las inserciones de datos en las tablas.

Este estudio no solo proporciona un marco teórico sobre el diseño de bases de datos, sino que también ofrece una visión práctica de cómo un sistema bien diseñado puede optimizar la experiencia del usuario, mejorar la gestión de inventarios y facilitar el análisis de datos, lo que a su vez impulsa la toma de decisiones estratégicas en un entorno empresarial competitivo.

Al final, se presentarán conclusiones sobre la importancia de una base de datos robusta en el éxito de un negocio de ventas en línea y su papel en la transformación digital del comercio.

Sustentación de las tablas creadas en este laboratorio:

1. Tabla Categorías

Objetivo

Organizar los productos en grupos temáticos para facilitar la navegación y búsqueda.

Justificación

La clasificación de productos en categorías permite a los usuarios encontrar rápidamente lo que buscan y también ayuda a los administradores a gestionar el inventario de manera más eficiente.

2. Tabla Productos

Objetivo

Almacenar la información esencial sobre los productos que se ofrecen en la tienda.

Justificación

Cada producto necesita un registro que incluya su nombre, descripción, precio y stock. La relación con las categorías permite clasificar productos y mejorar la experiencia del usuario.

3. Tabla Usuarios

Objetivo

Registrar la información de los clientes que interactúan con el sistema.

Justificación

Mantener un registro único de cada usuario es fundamental para la gestión de cuentas, pedidos y soporte al cliente. La información almacenada permite la autenticación y personalización de la experiencia del usuario.

4. Tabla Métodos de Pago

Objetivo

Definir los diferentes métodos de pago que los usuarios pueden utilizar.

Justificación

Ofrecer múltiples opciones de pago es crucial para satisfacer las preferencias de los clientes y facilitar la finalización de compras.

5. Tabla Órdenes

Objetivo

Registrar las órdenes realizadas por los usuarios.

Justificación

Las órdenes son la base del negocio; es necesario tener un seguimiento claro de cada transacción, incluyendo el usuario, total y método de pago. Esto facilita la gestión de pedidos y el análisis de ventas.

6. Tabla Detalles de Órdenes

Objetivo

Almacenar información específica sobre los productos incluidos en cada orden.

Justificación

Permite desglosar cada orden en sus componentes, lo cual es esencial para la gestión del inventario y la atención al cliente. Facilita también la generación de informes de ventas por producto.

7. Tabla Proveedores

Objetivo

Mantener un registro de los proveedores de productos.

Justificación

Es fundamental conocer las fuentes de los productos para gestionar el inventario, establecer relaciones comerciales y optimizar costos de adquisición.

8. Tabla Productos_Proveedores

Objetivo

Definir la relación entre productos y proveedores en un esquema de muchos a muchos.

Justificación

Muchos productos pueden ser suministrados por diferentes proveedores, y viceversa. Esta tabla permite gestionar estas relaciones de manera efectiva.

9. Tabla Reseñas

Objetivo

Almacenar las opiniones y calificaciones de los usuarios sobre los productos.

Justificación

Las reseñas son importantes para fomentar la confianza del cliente y proporcionar retroalimentación sobre los productos. También ayudan a otros usuarios a tomar decisiones de compra.

10. Tabla Direcciones de Envío

Objetivo

Registrar las direcciones a las que se enviarán los productos.

Justificación

Mantener información sobre direcciones de envío es crucial para procesar pedidos y garantizar entregas correctas y eficientes.

11. Tabla Carritos

Objetivo

Almacenar los carritos de compras de los usuarios.

Justificación

Permite a los usuarios agregar productos y revisar sus selecciones antes de completar la compra. Esto mejora la experiencia de compra y reduce el abandono del carrito.

12. Tabla Items en Carrito

Objetivo

Registrar los productos que cada usuario ha añadido a su carrito.

Justificación

Mantener un seguimiento de los artículos en el carrito permite a los usuarios revisar sus selecciones antes de finalizar la compra y es esencial para gestionar el proceso de checkout.

13. Tabla Cupones

Objetivo

Almacenar información sobre cupones de descuento.

Justificación

Los cupones son herramientas efectivas para incentivar compras y fomentar la lealtad del cliente. Es fundamental tener un registro claro de las promociones activas.

14. Tabla Órdenes_Cupones

Objetivo

Establecer una relación entre las órdenes y los cupones utilizados.

Justificación

Permite realizar un seguimiento de las promociones aplicadas a cada orden, lo que es crucial para la gestión de descuentos y la contabilidad.

15. Tabla Transacciones

Objetivo

Registrar las transacciones financieras relacionadas con las órdenes.

Justificación

Mantener un registro de las transacciones es esencial para la contabilidad y el seguimiento del estado de cada pedido. Permite gestionar mejor las finanzas del negocio

Código de Creación de tablas de este sistema de ventas online:

```
USE BD_VENTAS_ONLINE_G1;
GO

-- Tabla de Categorías
CREATE TABLE Categorías (
    CategoriaID INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(100) NOT NULL UNIQUE,
    Descripción NVARCHAR(255)
);
GO

-- Tabla de Productos
CREATE TABLE Productos (
    ProductoID INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(100) NOT NULL,
    Descripción NVARCHAR(255),
    Precio DECIMAL(10, 2) NOT NULL CHECK (Precio > 0),
    Stock INT NOT NULL CHECK (Stock >= 0),
    CategoriaID INT,
    CONSTRAINT FK_Producto_Categoria FOREIGN KEY (CategoriaID) REFERENCES
Categorías(CategoriaID)
        ON DELETE SET NULL ON UPDATE CASCADE
);
GO

-- Tabla de Usuarios
CREATE TABLE Usuarios (
```

```
    UsuarioID INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(100) NOT NULL,
    Correo NVARCHAR(100) NOT NULL UNIQUE,
    Contraseña NVARCHAR(255) NOT NULL,
    Dirección NVARCHAR(255),
    Teléfono NVARCHAR(20)
);
GO

-- Tabla de Métodos de Pago
CREATE TABLE MétodosPago (
    MétodoPagoID INT PRIMARY KEY IDENTITY(1,1),
    Tipo NVARCHAR(50) NOT NULL,
    Detalles NVARCHAR(255)
);
GO

-- Tabla de Órdenes
CREATE TABLE Órdenes (
    OrdenID INT PRIMARY KEY IDENTITY(1,1),
    UsuarioID INT NOT NULL,
    Fecha DATETIME DEFAULT GETDATE(),
    Total DECIMAL(10, 2) NOT NULL CHECK (Total >= 0),
    MétodoPagoID INT,
    CONSTRAINT FK_Orden_Usuario FOREIGN KEY (UsuarioID) REFERENCES
Usuarios(UsuarioID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Orden_MétodoPago FOREIGN KEY (MétodoPagoID)
REFERENCES MétodosPago(MétodoPagoID)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

```
GO

-- Tabla de Detalles de Órdenes
CREATE TABLE DetallesOrden (
    DetalleID INT PRIMARY KEY IDENTITY(1,1),
    OrdenID INT NOT NULL,
    ProductoID INT NOT NULL,
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    PrecioUnitario DECIMAL(10, 2) NOT NULL CHECK (PrecioUnitario > 0),
    CONSTRAINT FK_DetalleOrden_Orden FOREIGN KEY (OrdenID) REFERENCES Órdenes(OrdenID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_DetalleOrden_Producto FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
        ON DELETE CASCADE ON UPDATE CASCADE
);
GO

-- Tabla de Proveedores
CREATE TABLE Proveedores (
    ProveedorID INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(100) NOT NULL UNIQUE,
    Contacto NVARCHAR(100),
    Teléfono NVARCHAR(20),
    Dirección NVARCHAR(255)
);
GO

-- Tabla de Productos_Proveedores (relación muchos a muchos)
CREATE TABLE Productos_Proveedores (
    ProductoID INT NOT NULL,
```

```
    ProveedorID INT NOT NULL,
    PrecioCompra DECIMAL(10, 2) NOT NULL CHECK (PrecioCompra > 0),
    PRIMARY KEY (ProductoID, ProveedorID),
    CONSTRAINT FK_ProductoProveedor_Producto FOREIGN KEY (ProductoID)
REFERENCES Productos(ProductoID)
        ON DELETE CASCADE,
    CONSTRAINT FK_ProductoProveedor_Proveedor FOREIGN KEY (ProveedorID)
REFERENCES Proveedores(ProveedorID)
        ON DELETE CASCADE
);
GO

-- Tabla de Reseñas
CREATE TABLE Reseñas (
    ReseñaID INT PRIMARY KEY IDENTITY(1,1),
    ProductoID INT NOT NULL,
    UsuarioID INT NOT NULL,
    Calificación INT CHECK (Calificación BETWEEN 1 AND 5),
    Comentario NVARCHAR(255),
    Fecha DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Reseña_Producto FOREIGN KEY (ProductoID) REFERENCES
Productos(ProductoID)
        ON DELETE CASCADE,
    CONSTRAINT FK_Reseña_Usuario FOREIGN KEY (UsuarioID) REFERENCES
Usuarios(UsuarioID)
        ON DELETE CASCADE
);
GO

-- Tabla de Direcciones de Envío
CREATE TABLE DireccionesEnvio (
```

```
    DirecciónID INT PRIMARY KEY IDENTITY(1,1),
    UsuarioID INT NOT NULL,
    Dirección NVARCHAR(255) NOT NULL,
    Ciudad NVARCHAR(100) NOT NULL,
    Estado NVARCHAR(100) NOT NULL,
    CódigoPostal NVARCHAR(20) NOT NULL,
    CONSTRAINT FK_Direccion_Usuario FOREIGN KEY (UsuarioID) REFERENCES Usuarios(UsuarioID)
        ON DELETE CASCADE
);
GO

-- Tabla de Carritos
CREATE TABLE Carritos (
    CarritoID INT PRIMARY KEY IDENTITY(1,1),
    UsuarioID INT NOT NULL,
    FechaCreacion DATETIME DEFAULT GETDATE(),
    CONSTRAINT FK_Carrito_Usuario FOREIGN KEY (UsuarioID) REFERENCES Usuarios(UsuarioID)
        ON DELETE CASCADE
);
GO

-- Tabla de Items en Carrito
CREATE TABLE ItemsCarrito (
    ItemID INT PRIMARY KEY IDENTITY(1,1),
    CarritoID INT NOT NULL,
    ProductoID INT NOT NULL,
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    CONSTRAINT FK_ItemCarrito_Carrito FOREIGN KEY (CarritoID) REFERENCES Carritos(CarritoID)
```

```
        ON DELETE CASCADE,
    CONSTRAINT FK_ItemCarrito_Producto FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
        ON DELETE CASCADE
);
GO

-- Tabla de Cupones
CREATE TABLE Cupones (
    CuponID INT PRIMARY KEY IDENTITY(1,1),
    Codigo NVARCHAR(50) NOT NULL UNIQUE,
    Descuento DECIMAL(5, 2) CHECK (Descuento >= 0 AND Descuento <= 100),
    FechaExpiracion DATETIME NOT NULL
);
GO

-- Tabla de Órdenes_Cupones (relación muchos a muchos)
CREATE TABLE Órdenes_Cupones (
    OrdenID INT NOT NULL,
    CuponID INT NOT NULL,
    PRIMARY KEY (OrdenID, CuponID),
    CONSTRAINT FK_OrdenCupon_Orden FOREIGN KEY (OrdenID) REFERENCES Órdenes(OrdenID)
        ON DELETE CASCADE,
    CONSTRAINT FK_OrdenCupon_Cupon FOREIGN KEY (CuponID) REFERENCES Cupones(CuponID)
        ON DELETE CASCADE
);
GO

-- Tabla de Transacciones
```

```
CREATE TABLE Transacciones (
    TransaccionID INT PRIMARY KEY IDENTITY(1,1),
    OrdenID INT NOT NULL,
    Monto DECIMAL(10, 2) NOT NULL CHECK (Monto >= 0),
    Fecha DATETIME DEFAULT GETDATE(),
    Estado NVARCHAR(50) NOT NULL,
    CONSTRAINT FK_Transaccion_Orden FOREIGN KEY (OrdenID) REFERENCES Órdenes(OrdenID)
        ON DELETE CASCADE
);
GO

USE BD_VENTAS_ONLINE_G1;
GO
```

Sustentación de los tipos de datos utilizados (Creación de Tablas):

1.Restricciones y Justificación

Restricciones Primarias

- **PRIMARY KEY**: Cada tabla tiene una clave primaria que asegura la unicidad de cada registro. Esto es esencial para identificar de manera única cada entidad, como productos, usuarios y órdenes.

Restricciones de Integridad Referencial

- **FOREIGN KEY**: Las claves foráneas establecen relaciones entre las tablas, garantizando que las referencias sean válidas.
    - Ejemplo: FK_Producto_Categoria asegura que cada producto pertenezca a una categoría existente. Esto mantiene la integridad de la base de datos y previene registros huérfanos.

Restricciones de Unicidad

- **UNIQUE**: Se aplica a campos como Correo en la tabla de Usuarios y Codigo en la tabla de Cupones. Esto asegura que no haya duplicados, lo que es crucial para la autenticación de usuarios y la aplicación de cupones.

Restricciones de Chequeo

- **CHECK**: Se utilizan para validar valores en columnas específicas:
    - Precio > 0 y Stock >= 0: Asegura que no se registren precios negativos o cantidades de stock negativas, lo cual no tendría sentido en un contexto comercial.
    - Calificación BETWEEN 1 AND 5: Asegura que las calificaciones de las reseñas estén dentro de un rango válido, evitando datos erróneos.

Restricciones de Definición de Nulo

- Las columnas que no pueden contener valores nulos (como Nombre en Usuarios) son importantes para asegurar que se tenga información clave antes de procesar o almacenar un registro.

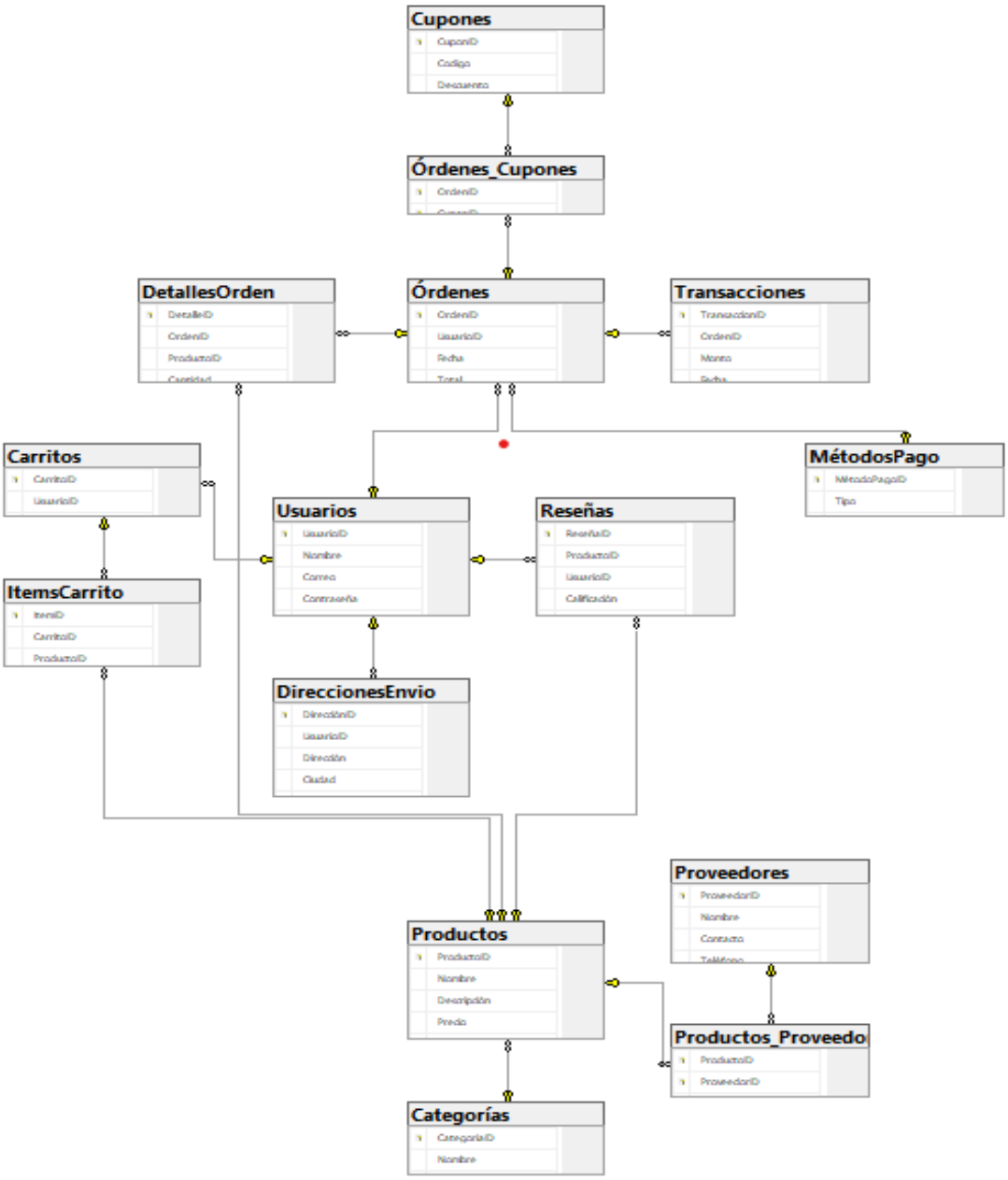
---

2. Tipos de Datos y Justificación

Tipos de Datos Utilizados

- **INT**: Se usa para identificadores y conteos, como ProductoID, UsuarioID y Cantidad. Es eficiente en términos de almacenamiento y permite realizar operaciones matemáticas rápidas.
- **NVARCHAR(n)**: Se utiliza para cadenas de texto, permitiendo almacenar nombres, descripciones y comentarios. El uso de NVARCHAR permite soportar caracteres de diferentes idiomas y evita problemas de codificación.
- **DECIMAL(p, s)**: Para precios y montos, se usa DECIMAL con precisión y escala definidas (p = total de dígitos, s = dígitos después del punto decimal). Esto es importante para evitar errores de redondeo en operaciones financieras.
- **DATETIME**: Se usa para campos de fecha y hora como Fecha y FechaExpiracion, lo que permite realizar un seguimiento preciso de transacciones y eventos.

Modelado de tablas (Modelo Relacional):



Sustentación de los inserts utilizados en este laboratorio:

1.Inserción en la Tabla de Categorías

```
IF NOT EXISTS (SELECT * FROM Categorías WHERE Nombre = 'Electrónica')
BEGIN
    INSERT INTO Categorías (Nombre, Descripción) VALUES
    ('Electrónica', 'Dispositivos y gadgets electrónicos.'),
    ('Ropa', 'Ropa y accesorios de moda.'),
    ('Hogar', 'Artículos para el hogar y la decoración.'),
    ('Juegos', 'Videojuegos y accesorios para jugar.');
END
```

Justificación: Se insertan categorías que son fundamentales para clasificar los productos. Se verifica si ya existen para evitar duplicados, lo que garantiza la integridad de los datos.

2. Inserción en la Tabla de Productos

```
IF NOT EXISTS (SELECT * FROM Productos WHERE Nombre = 'Smartphone')
BEGIN
    INSERT INTO Productos (Nombre, Descripción, Precio, Stock, CategoriaID) VALUES
    ('Smartphone', 'Teléfono inteligente de última generación.', 699.99, 50, 1),
    ('Laptop', 'Laptop ultradelgada y ligera.', 999.99, 30, 1),
    ('Camiseta', 'Camiseta de algodón de alta calidad.', 19.99, 100, 2),
    ('Sofá', 'Sofá de dos plazas, cómodo y elegante.', 499.99, 20, 3),
    ('Auriculares Inalámbricos', 'Auriculares con cancelación de ruido.', 199.99, 40, 1),
    ('Consola de Videojuegos', 'Consola de videojuegos de última generación.', 399.99, 15, 4);
END
```

Justificación: Se añaden productos representativos en distintas categorías. Al usar CategoriaID, se establece una relación con la tabla de categorías, asegurando que cada producto esté correctamente clasificado. Las condiciones de existencia previenen duplicados.

3. Inserción en la Tabla de Usuarios

IF NOT EXISTS (SELECT * FROM Usuarios WHERE Correo = ‘juan.perez@example.com’)

BEGIN

INSERT INTO Usuarios (Nombre, Correo, Contraseña, Dirección, Teléfono) VALUES

(‘Juan Pérez’, ‘juan.perez@example.com’, ‘contraseña123’, ‘Calle 123, Ciudad’, ‘555-1234’),

(‘María López’, ‘maria.lopez@example.com’, ‘password456’, ‘Avenida 456, Ciudad’, ‘555-5678’),

(‘Lucía Martínez’, ‘lucia.martinez@example.com’, ‘luciapassword’, ‘Calle Nueva, Ciudad’, ‘555-3456’);

END

Justificación: Se crean registros de usuarios, fundamentales para las transacciones. Cada usuario tiene un correo único, lo que es vital para el acceso y la seguridad. Las contraseñas están almacenadas en texto plano aquí por simplicidad, pero en un entorno real se deberían usar técnicas de hashing.

4. Inserción en la Tabla de Métodos de Pago

IF NOT EXISTS (SELECT * FROM MétodosPago WHERE Tipo = ‘Tarjeta de Crédito’)

BEGIN

INSERT INTO MétodosPago (Tipo, Detalles) VALUES

(‘Tarjeta de Crédito’, ‘Visa y MasterCard aceptadas.’),

(‘PayPal’, ‘Pago seguro a través de PayPal.’),

(‘Transferencia Bancaria’, ‘Transferencia directa a la cuenta del vendedor.’),

(‘Bitcoin’, ‘Pago mediante criptomonedas.’);

END

Justificación: Se definen métodos de pago disponibles para los usuarios. Esto es esencial para facilitar las transacciones y brindar opciones variadas a los clientes.

5. Inserción en la Tabla de Órdenes

```
IF NOT EXISTS (SELECT * FROM Órdenes WHERE UsuarioID = 1)
BEGIN
    INSERT INTO Órdenes (UsuarioID, Total, MétodoPagoID) VALUES
    (1, 719.98, 1),
    (2, 499.99, 2),
    (3, 399.99, 1);
END
```

Justificación: Se registran órdenes asociadas a usuarios, con un total que representa la suma de los productos comprados. Se asegura que cada orden esté vinculada a un usuario y un método de pago.

6. Inserción en la Tabla de Detalles de Órdenes

```
IF NOT EXISTS (SELECT * FROM DetallesOrden WHERE OrdenID = 1 AND ProductoID = 1)
BEGIN
    INSERT INTO DetallesOrden (OrdenID, ProductoID, Cantidad, PrecioUnitario) VALUES
    (1, 1, 1, 699.99),
    (1, 3, 1, 19.99),
    (2, 4, 1, 499.99),
    (3, 6, 1, 399.99);
END
```

Justificación: Los detalles de las órdenes especifican qué productos se compraron, su cantidad y precio unitario. Esto es fundamental para el historial de compras y para el cálculo de totales.

7. Inserción en la Tabla de Proveedores

IF NOT EXISTS (SELECT * FROM Proveedores WHERE Nombre = ‘Luis Fernández’)

BEGIN

INSERT INTO Proveedores (Nombre, Contacto, Teléfono, Dirección) VALUES

(‘Luis Fernández’, ‘Luis Fernández’, ‘555-0033’, ‘Calle Real 789’),

(‘Ana Gómez’, ‘Ana Gómez’, ‘555-4567’, ‘Calle Secundaria 456’);

END

Justificación: Se registran proveedores que suministran productos. Conocer a los proveedores es esencial para la gestión de inventarios y la logística.

8. Inserción en la Tabla Productos_Proveedores

IF NOT EXISTS (SELECT * FROM Productos_Proveedores WHERE ProductoID = 1 AND ProveedorID = 1)

BEGIN

INSERT INTO Productos_Proveedores (ProductoID, ProveedorID, PrecioCompra) VALUES

(1, 1, 650.00),

(3, 1, 15.00),

(4, 2, 450.00);

END

Justificación: Establece la relación entre productos y sus proveedores, así como el precio de compra. Esto permite un seguimiento claro del costo y la procedencia de los productos.

9. Inserción en la Tabla de Reseñas

IF NOT EXISTS (SELECT * FROM Reseñas WHERE ProductoID = 1 AND UsuarioID = 1)

BEGIN

INSERT INTO Reseñas (ProductoID, UsuarioID, Calificación, Comentario) VALUES

(1, 1, 5, ‘Excelente smartphone, funciona perfectamente.’),

(3, 2, 4, ‘Buena camiseta, aunque un poco pequeña.’),

(6, 3, 5, ‘La consola es increíble, muchos juegos para disfrutar.’);

END

Justificación: Se registran reseñas para productos, lo cual es vital para la retroalimentación del cliente y mejora continua. Cada reseña está asociada a un producto y un usuario, manteniendo la relación.

10. Inserción en la Tabla Direcciones de Envío

IF NOT EXISTS (SELECT * FROM DireccionesEnvio WHERE UsuarioID = 1)

BEGIN

INSERT INTO DireccionesEnvio (UsuarioID, Dirección, Ciudad, Estado, CódigoPostal) VALUES

(1, ‘Calle 123, Ciudad’, ‘Ciudad’, ‘Estado’, ‘11111’),

(2, ‘Avenida 456, Ciudad’, ‘Ciudad’, ‘Estado’, ‘22222’),

(3, ‘Calle Nueva, Ciudad’, ‘Ciudad’, ‘Estado’, ‘33333’);

END

Justificación: Se almacenan direcciones de envío para los usuarios, necesarias para el proceso de entrega. La verificación evita la duplicación.

11. Inserción en la Tabla de Carritos

IF NOT EXISTS (SELECT * FROM Carritos WHERE UsuarioID = 1)

BEGIN

INSERT INTO Carritos (UsuarioID) VALUES

(1),

(2),

(3);

END

Justificación: Se crean carritos para cada usuario. Los carritos permiten a los usuarios gestionar sus compras antes de realizar un pedido.

12. Inserción en la Tabla Items en Carrito

IF NOT EXISTS (SELECT * FROM ItemsCarrito WHERE CarritoID = 1 AND ProductoID = 1)

BEGIN

INSERT INTO ItemsCarrito (CarritoID, ProductoID, Cantidad) VALUES

(1, 1, 1),

(2, 3, 2),

(3, 6, 1);

END

Justificación: Los ítems en el carrito registran qué productos han sido seleccionados por los usuarios y en qué cantidades, lo cual es esencial para el proceso de compra.

13. Inserción en la Tabla de Cupones

sql

Copiar 22ódigo

IF NOT EXISTS (SELECT * FROM Cupones WHERE Codigo = ‘FREESHIP’)

BEGIN

INSERT INTO Cupones (Codigo, Descuento, FechaExpiracion) VALUES

(‘FREESHIP’, 0.00, ‘2024-05-15’),

(‘DESC10’, 10.00, ‘2024-06-30’);

END

Justificación: Se crean cupones para promociones y descuentos, incentivando las compras. La verificación evita duplicados.

14. Inserción en la Tabla de Órdenes_Cupones

IF NOT EXISTS (SELECT * FROM Órdenes_Cupones WHERE OrdenID = 1 AND CuponID = 1)

BEGIN

INSERT INTO Órdenes_Cupones (OrdenID, CuponID) VALUES

(1, 1),

(2, 2);

END

Justificación: Se registran los cupones utilizados en las órdenes, lo que permite hacer un seguimiento de las promociones aplicadas.

15. Inserción en la Tabla de Transacciones

IF NOT EXISTS (SELECT * FROM Transacciones WHERE OrdenID = 1)

BEGIN

INSERT INTO Transacciones (OrdenID, Monto, Estado) VALUES

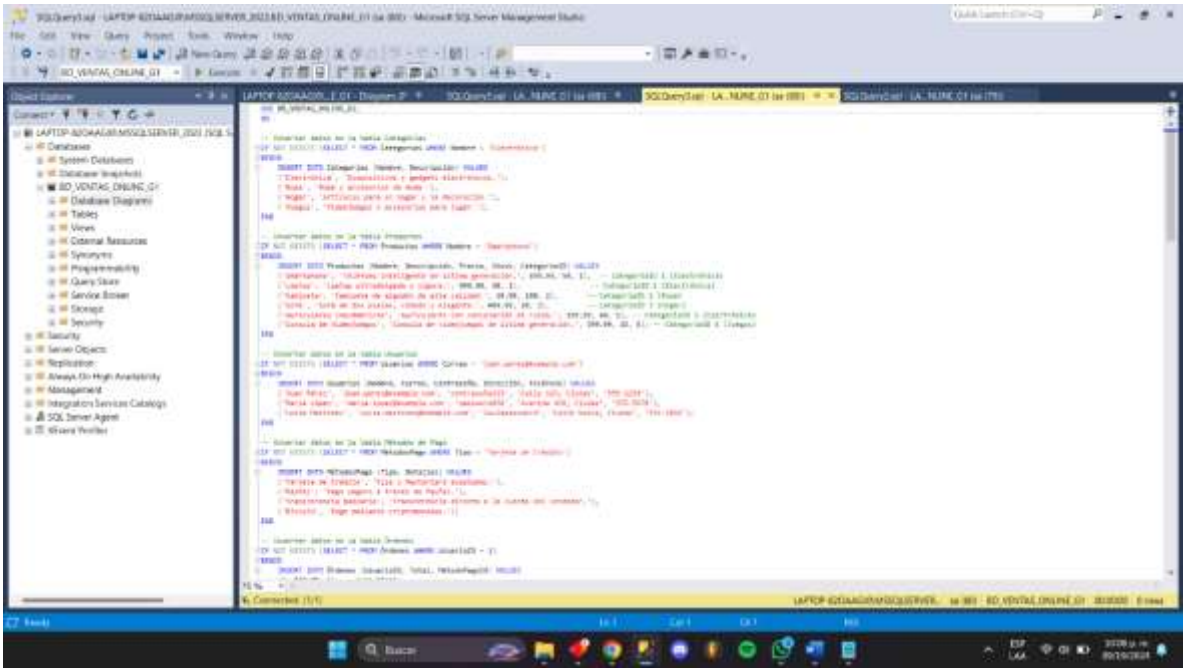
(1, 719.98, ‘Completado’),

(2, 499.99, ‘Completado’),

(3, 399.99, ‘Pendiente’);

END

Justificación: Se registran transacciones relacionadas con las órdenes. Cada transacción incluye el monto y el estado, lo que es crucial para el manejo de la contabilidad y el seguimiento de los pagos.



Tipos de datos utilizados en los inserts:

1.Tabla Categorías

Nombre NVARCHAR(100): Se utiliza NVARCHAR para permitir caracteres especiales y acentos, común en nombres de categorías. El tamaño de 100 caracteres es suficiente para la mayoría de los nombres.

Descripción NVARCHAR(255): También se usa NVARCHAR para la descripción, permitiendo suficiente longitud para describir la categoría.

2. Tabla Productos

Nombre NVARCHAR(100): Al igual que en la tabla de categorías, se permite el uso de caracteres especiales.

Descripción NVARCHAR(255): Se justifica por la necesidad de descripciones detalladas de los productos.

Precio DECIMAL(10, 2): Se usa DECIMAL para manejar valores monetarios con precisión, permitiendo hasta 10 dígitos en total, de los cuales 2 son decimales, adecuado para precios.

Stock INT: Un número entero es adecuado para representar cantidades de productos disponibles, que no pueden ser negativas.

CategoriaID INT: Se usa un entero para referenciar la categoría del producto, relacionándose con la tabla de categorías.

3. Tabla Usuarios

Nombre NVARCHAR(100): Similar a las categorías y productos, permite caracteres especiales en nombres.

Correo NVARCHAR(100): Un tamaño de 100 caracteres es adecuado para direcciones de correo electrónico.

Contraseña NVARCHAR(255): Se utiliza un tamaño mayor para almacenar contraseñas de forma segura (idealmente, en texto cifrado).

Dirección NVARCHAR(255): Se permite suficiente longitud para incluir detalles de la dirección.

Teléfono NVARCHAR(20): Se usa NVARCHAR para permitir caracteres como guiones o paréntesis, y 20 caracteres son suficientes para la mayoría de los números de teléfono.

4. Tabla Métodos de Pago

Tipo NVARCHAR(50): Se utiliza NVARCHAR para permitir variaciones en los nombres de los métodos de pago.

Detalles NVARCHAR(255): Permite suficiente longitud para explicar el método de pago.

5. Tabla Órdenes

UsuarioID INT: Se utiliza un entero para hacer referencia a la tabla de usuarios.

Fecha DATETIME: Se usa DATETIME para registrar la fecha y hora de la orden.

Total DECIMAL(10, 2): Se utiliza DECIMAL para reflejar el total de la orden con precisión.

MétodoPagoID INT: Se usa un entero para referenciar el método de pago.

6. Tabla Detalles de Órdenes

OrdenID INT: Se usa un entero para hacer referencia a la tabla de órdenes.

ProductoID INT: Se utiliza un entero para referenciar el producto comprado.

Cantidad INT: Un entero es adecuado para representar la cantidad de productos.

PrecioUnitario DECIMAL(10, 2): Se usa DECIMAL para almacenar el precio unitario de los productos con precisión.

7. Tabla Proveedores

Nombre NVARCHAR(100): Se usa NVARCHAR para permitir caracteres especiales en nombres.

Contacto NVARCHAR(100): Se permite suficiente longitud para incluir el nombre del contacto.

Teléfono NVARCHAR(20): Similar a la tabla de usuarios, para permitir caracteres como guiones.

Dirección NVARCHAR(255): Se permite suficiente longitud para detalles de la dirección.

8. Tabla Productos_Proveedores

ProductoID INT: Se utiliza un entero para referenciar el producto.

ProveedorID INT: Se utiliza un entero para referenciar el proveedor.

PrecioCompra DECIMAL(10, 2): Se usa DECIMAL para reflejar el precio de compra con precisión.

9. Tabla Reseñas

ProductoID INT: Se utiliza un entero para referenciar el producto que está siendo reseñado.

UsuarioID INT: Se utiliza un entero para referenciar al usuario que deja la reseña.

Calificación INT: Se usa un entero para representar la calificación de 1 a 5.

Comentario NVARCHAR(255): Se permite suficiente longitud para incluir comentarios.

10. Tabla Direcciones de Envío

UsuarioID INT: Se utiliza un entero para referenciar al usuario.

Dirección NVARCHAR(255): Se permite suficiente longitud para la dirección.

Ciudad NVARCHAR(100): Se usa NVARCHAR para permitir caracteres especiales.

Estado NVARCHAR(100): Similar a la ciudad, se permite un tamaño adecuado.

CódigoPostal NVARCHAR(20): Se utiliza NVARCHAR para permitir formatos variados de códigos postales.

11. Tabla Carritos

UsuarioID INT: Se utiliza un entero para referenciar al usuario.

FechaCreacion DATETIME: Se usa DATETIME para registrar cuándo se creó el carrito.

12. Tabla Items en Carrito

CarritoID INT: Se usa un entero para referenciar el carrito.

ProductoID INT: Se utiliza un entero para referenciar el producto.

Cantidad INT: Un entero es adecuado para representar la cantidad de productos en el carrito.

13. Tabla Cupones

Codigo NVARCHAR(50): Se utiliza NVARCHAR para permitir variaciones en los códigos de cupones.

Descuento DECIMAL(5, 2): Se usa DECIMAL para representar porcentajes de descuento con precisión.

FechaExpiracion DATETIME: Se utiliza DATETIME para registrar la fecha de expiración del cupón.

14. Tabla Órdenes_Cupones

OrdenID INT: Se utiliza un entero para referenciar la orden.

CuponID INT: Se usa un entero para referenciar el cupón.

15. Tabla Transacciones

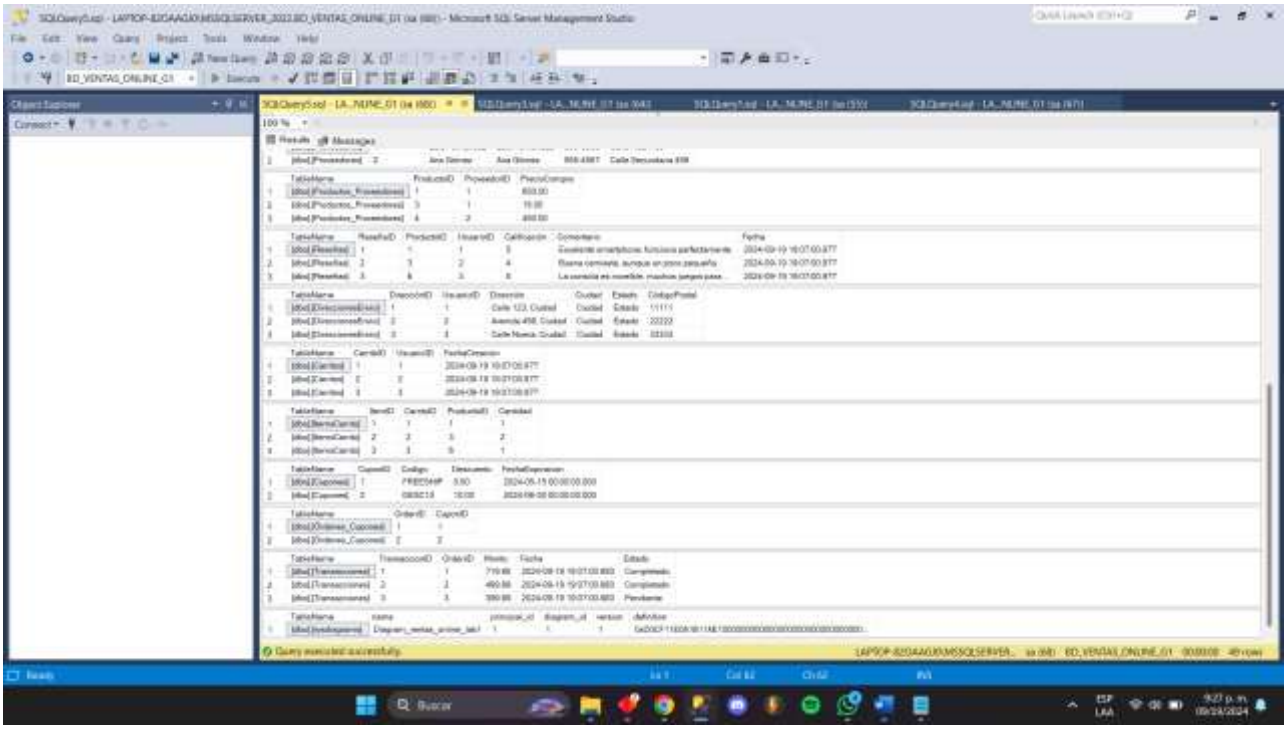
OrdenID INT: Se utiliza un entero para referenciar la orden asociada a la transacción.

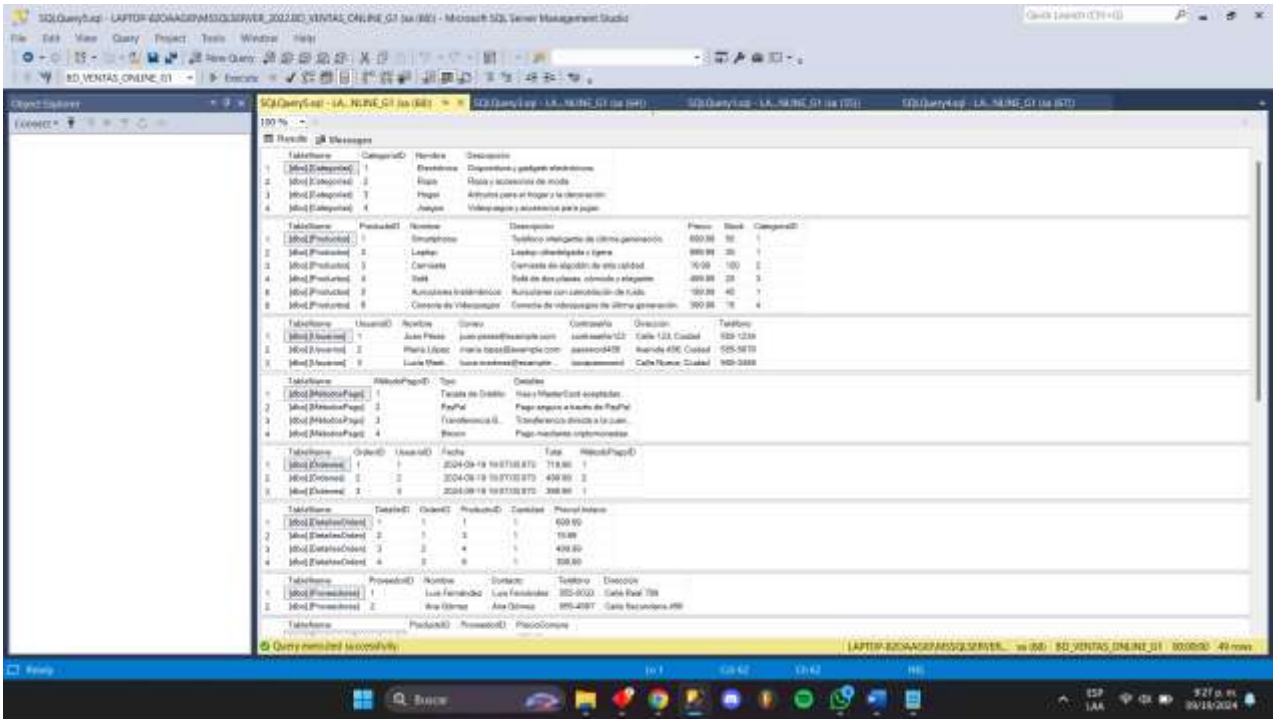
Monto DECIMAL(10, 2): Se usa DECIMAL para reflejar el monto de la transacción con precisión.

Fecha DATETIME: Se utiliza DATETIME para registrar la fecha de la transacción.

Estado NVARCHAR(50): Se usa NVARCHAR para describir el estado de la transacción (completado, pendiente, etc.).

Resultados de Tablas pobladas por inserts:





Codigo de Creación de base de datos en SQL:

```
-- Sistema de ventas en línea SQL

-- Autoria: GRUPO 1

-- Creación de la base de datos

USE master;

GO

CREATE DATABASE BD_VENTAS_ONLINE_G1 ON

(NAME = BD_VENTAS_ONLINE_G1,

 FILENAME =

'C:\SEMESTRE\BASE_DE_DATOS\BD_VENTAS_ONLINE_G1_Laboratorio1_modelado_s2_2024.mdf',

 SIZE = 100, -- Tamaño de BD_dat inicial

 MAXSIZE = 200,

 FILEGROWTH = 10 MB) -- Aumento de el tamaño

LOG ON(

NAME = BD_VENTAS_ONLINE_G1_log,

 FILENAME =

'C:\SEMESTRE\BASE_DE_DATOS\BD_VENTAS_ONLINE_G1_Laboratorio1_modelado_s2_2024.ldf',

 SIZE = 50 MB, -- Tamaño de BD_log inicial

 MAXSIZE = 100 MB,

 FILEGROWTH = 20%) -- Aumento del tamaño del log

GO
```

Conclusión:

El desarrollo de un sistema de base de datos para un entorno de ventas en línea es fundamental para el éxito de cualquier negocio en el comercio electrónico. A través de un diseño estructurado y bien planificado, se logra no solo el almacenamiento eficiente de datos, sino también la facilitación de procesos clave como la gestión de inventarios, el seguimiento de órdenes y la atención al cliente.

Las tablas creadas, que incluyen categorías, productos, usuarios y órdenes, representan las entidades esenciales para la operación del sistema. Las relaciones entre estas tablas permiten un flujo de información coherente y organizado, garantizando que los datos sean accesibles y estén interconectados de manera lógica. Esto no solo mejora la integridad de los datos, sino que también optimiza la experiencia del usuario, permitiendo una navegación fluida y un proceso de compra más eficiente.

La selección de tipos de datos apropiados en las inserciones asegura que la información se almacene de manera precisa y que se cumplan las restricciones necesarias para mantener la calidad de los datos. Cada decisión, desde el diseño de las tablas hasta la implementación de restricciones y tipos de datos, contribuye a la robustez del sistema.

En un mercado cada vez más competitivo, contar con un sistema de base de datos eficiente es crucial para la toma de decisiones informadas y para la adaptación a las demandas del consumidor. En conclusión, la implementación de un sistema de base de datos bien diseñado no solo es una herramienta operativa, sino un elemento estratégico que puede impulsar el crecimiento y la sostenibilidad de un negocio en el ámbito del comercio electrónico.