MANUAL DE USUARIO DE LA BASE DE DATOS

ANDERSON DANIEL CERON ASTAIZA

DOCENTE: BRAYAN ARCOS

INSTITUTO TECNOLOGICO DEL PUTUMAYO

DESARROLLO DE BASE DE DATOS II

MOCOA-PUTUMAYO

1/12/2024

Contenido

[Resumen Ejecutivo 3](#_Toc184373386)

[Introducción 4](#_Toc184373387)

[Metodología 5](#_Toc184373388)

[Desarrollo Del Informe 6](#_Toc184373389)

[Descripción de la Base de Datos: 6](#_Toc184373390)

[Entidades Principales y Relaciones 6](#_Toc184373391)

[Resultado de Consultas 13](#_Toc184373392)

[Diseño de base de datos 46](#_Toc184373393)

[Normalización: 46](#_Toc184373394)

[Consideraciones de diseño 46](#_Toc184373395)

[Análisis y discusiones 49](#_Toc184373396)

[Conclusión 51](#_Toc184373397)

[Recomendaciones 52](#_Toc184373398)

# Resumen Ejecutivo

El presente informe documenta el diseño, desarrollo y funcionalidad de la base de datos store , la cual está diseñada para gestionar de manera eficiente el inventario, las ventas y los usuarios asociados a una tienda virtual. Este documento cubre aspectos técnicos esenciales para su implementación y uso, así como buenas prácticas de gestión de bases de datos relacionales.

# Introducción

**Contexto y motivación**  
En un contexto donde la gestión de inventarios y transacciones comerciales requiere soluciones tecnológicas robustas, surge la necesidad de una base de datos que facilite la administración centralizada y optimizada de productos, usuarios y pagos en una tienda virtual. La base de datos store responde a esta necesidad, proporcionando un modelo relacional bien estructurado que permite realizar consultas eficientes y operaciones complejas.

**Alcance del Informe**  
Este informe abarca el diseño lógico y físico de la base de datos, ejemplos de consultas SQL esenciales, optimización de índices, procedimientos almacenados básicos y recomendaciones para el uso eficiente de las relaciones.

# Metodología

**Herramientas**

* MySQL Workbench : Para el diseño del modelo entidad-relación (EER) y la ejecución de consultas.

**Procedimientos**

* Análisis de requisitos

Identificación de las necesidades del sistema, como la gestión de usuarios, roles, productos, inventarios y transacciones.

Definición de entidades principales (usuarios, facturas, productos, etc.) y sus relaciones.

* Diseño del Modelo Relacional

Creación del modelo EER (Entidad-Relación) basado en los requisitos identificados.

Normalización de datos para evitar redundancia y asegurar consistencia en las tablas.

* Implementación de la Base de Datos

Traducción del modelo EER a scripts SQL para crear las tablas, índices y relaciones.

Configuración de restricciones como claves primarias y extranjeras.

* Validación y pruebas

Inserción de datos ficticios para probar la funcionalidad de la base de datos.

Ejecución de consultas básicas (SELECCIONAR, INSERTAR, ACTUALIZAR, ELIMINAR) para verificar la integridad y el rendimiento.

* Documentación y Optimización

Documentación detallada de los esquemas, campos y relaciones.

Optimización de consultas mediante la creación de índices.

# Desarrollo Del Informe

## Descripción de la Base de Datos:

La base de datos store está diseñada para gestionar eficientemente los procesos operativos de una tienda virtual. Este sistema centraliza la información de los usuarios, productos, inventarios, pagos y transacciones, asegurando una estructura robusta y escalable que facilita la interacción entre sus diferentes módulos. A continuación, se describen sus principales componentes y funcionalidades:

## Entidades Principales y Relaciones

1. **countries**

* **Descripción**: Contiene los países registrados.
* **Campos principales**:
  + id(INT, PK): Identificador único del país.
  + name(VARCHAR(100)): Nombre del país.
* **Relaciones**: Relacionada con departments(1 :N ).

Aquí está la descripción completa de las **16 tablas** de tu base de datos store, incluyendo claves principales, claves extranjeras y relaciones.

**1.countries**

* **Descripción** : Contiene los países registrados.
* **Campos principales** :
  + id(INT, PK): Identificador único del país.
  + name(VARCHAR(100)): Nombre del país.
* **Relaciones** : Relacionada con departments(1 :N ).

1. **departments**

* **Descripción**: Almacena los departamentos dentro de un país.
* **Campos principales**:
  + id(INT, PK): Identificador único del departamento.
  + name(VARCHAR(100)): Nombre del departamento.
  + id\_country(INT, FK): Clave foránea hacia countries.
* **Relaciones**:
  + Relacionado con countries(N:1).
  + Relacionado con municipalities(1 :N ).

1. **municipalities**

* **Descripción**: Lista los municipios asociados a los departamentos.
* **Campos principales**:
  + id(INT, PK): Identificador único del municipio.
  + name(VARCHAR(100)): Nombre del municipio.
  + id\_department(INT, FK): Clave foránea hacia departments.
* **Relaciones**: Relacionada con departments(N:1) y con users(1 :N ).

1. **users**

* **Descripción**: Usuarios del sistema, incluidos clientes y vendedores.
* **Campos principales**:
  + id(INT, PK): Identificador único del usuario.
  + email(VARCHAR(100)): Correo electrónico del usuario.
  + password(VARCHAR(50)): Contraseña.
  + id\_document\_type(INT, FK): Clave foránea hacia document\_types.
  + id\_municipality(INT, FK): Clave foránea hacia municipalities.
* **Relaciones**: Relacionada con document\_types(N:1), municipalities(N:1), peoples(1:1), y users\_roles(1 :N ).

1. **document\_types**

* **Descripción**: Tipos de documentos de identificación.
* **Campos principales**:
  + id(INT, PK): Identificador único del tipo de documento.
  + name(VARCHAR(50)): Nombre del tipo de documento.
* **Relaciones**: Relacionada con users(1 :N )

1. **peoples**

* **Descripción**: Información personal vinculada a los usuarios.
* **Campos principales**:
  + id(INT, PK): Identificador único de la persona.
  + name1, name2(VARCHAR(100)): Primer y segundo nombre.
  + last\_name1, last\_name2(VARCHAR(100)): Primer y segundo apellido.
  + document\_number(VARCHAR(50)): Número de documento.
  + id\_user(INT, FK): Clave foránea hacia users.
* **Relaciones**: Relacionada con users(1:1).

1. **roles**

* **Descripción**: Roles disponibles en el sistema.
* **Campos principales**:
  + id(INT, PK): Identificador único del rol.
  + name(VARCHAR(255)): Nombre del rol.
* **Relaciones**: Relacionada con users\_roles(1 :N ).

1. **users\_roles**

* **Descripción**: Relación usuarios con roles específicos.
* **Campos principales**:
  + id\_person(INT, FK): Clave foránea hacia peoples.
  + id\_role(INT, FK): Clave foránea hacia roles.
* **Relaciones**:
  + Relacionado con peoples(N:1).
  + Relacionado con roles(N:1).

1. **products**

* **Descripción**: Productos disponibles en la tienda.
* **Campos principales**:
  + id(INT, PK): Identificador único del producto.
  + name(VARCHAR(100)): Nombre del producto.
  + description(TEXTO): Descripción del producto.
  + price(DECIMAL(10,2)): Precio del producto.
  + id\_product\_type(INT, FK): Clave foránea hacia product\_types.
* **Relaciones**:
  + Relacionado con product\_types(N:1).
  + Relacionado con invoice\_products(1 :N ).
  + Relacionado con store\_products(1 :N ).

1. **product\_types**

* **Descripción**: Categorías de los productos.
* **Campos principales**:
  + id(INT, PK): Identificador único del tipo de producto.
  + name(VARCHAR(100)): Nombre del tipo de producto.
* **Relaciones**: Relacionada con products(1 :N ).

1. **invoices**

* **Descripción**: Facturas generadas por las transacciones.
* **Campos principales**:
  + id(INT, PK): Identificador único de la factura.
  + id\_seller(INT, FK): Clave foránea hacia users.
  + id\_customer(INT, FK): Clave foránea hacia users.
  + date(FECHA): Fecha de la factura.
  + total(DECIMAL(10,2)): Monto total de la factura.
* **Relaciones**:
  + Relacionada con users(N:1 para vendedor y cliente).
  + Relacionado con invoice\_products(1 :N ).
  + Relacionado con payments(1 :N ).

1. **invoice\_products**

* **Descripción**: Productos asociados a cada factura.
* **Campos principales**:
  + id(INT, PK): Identificador único.
  + id\_invoice(INT, FK): Clave foránea hacia invoices.
  + id\_product(INT, FK): Clave foránea hacia products.
  + quantity(INT): Cantidad de producto.
  + unit\_price(DECIMAL(10,2)): Precio unitario.
  + subtotal(DECIMAL(10,2)): Subtotal del producto.
* **Relaciones**:
  + Relacionado con invoices(N:1).
  + Relacionado con products(N:1).

1. **payments**

* **Descripción**: Pagos realizados para las facturas.
* **Campos principales**:
  + id(INT, PK): Identificador único del pago.
  + id\_invoice(INT, FK): Clave foránea hacia invoices.
  + id\_payment\_method(INT, FK): Clave foránea hacia payment\_methods.
  + amount(DECIMAL(10,2)): Monto del pago.
  + date(FECHA): Fecha del pago.
* **Relaciones**:
  + Relacionado con invoices(N:1).
  + Relacionado con payment\_methods(N:1).

1. **payment\_methods**

* **Descripción**: Métodos de pago disponibles.
* **Campos principales**:
  + id(INT, PK): Identificador único del método de pago.
  + name(VARCHAR(50)): Nombre del método de pago.
* **Relaciones**: Relacionada con payments(1 :N ).

1. **store\_products**

* **Descripción**: Relación entre las tiendas y sus productos.
* **Campos principales**:
  + id(INT, PK): Identificador único.
  + id\_product(INT, FK): Clave foránea hacia products.
  + stock(INT): Cantidad en inventario.
  + last\_update(FECHA): Fecha de última actualización.
* **Relaciones**:
  + Relacionado con products(N:1).
  + Relacionado con stores(N:1).

1. **stores**

* **Descripción**: Tiendas que almacenan productos.
* **Campos principales**:
  + id(INT, PK): Identificador único de la tienda.
  + id\_store\_product(INT, FK): Clave foránea hacia store\_products.
* **Relaciones**: Relacionada con store\_products(1 :N ).

## Resultado de Consultas

1. Consulta INNER JOIN
2. con la tabla person:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Consulta INNER JOIN con la tabla product:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Consulta INNER JOIN con la tabla munipality:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Consulta LEFT JOIN
2. Consulta con LEFT JOINpara obtener personas y su rol, (si lo tienen )

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Consulta con LEFT JOIN para obtener todos los productos y las facturas asociadas (si existen):

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

1. Consulta con LEFT JOIN
2. Consulta con RIGHT JOIN para obtener todas las personas y su rol (incluso si no tienen rol asignado)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Consulta con RIGHT JOIN para obtener el tipo de producto, aunque no esté en una factura

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Consulta con CROSS JOIN
2. Consulta con CROSS JOINpara combinar todas las personas con todos los roles

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Consulta con CROSS JOIN para combinar todos los productos con todas las facturas

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Diagrama de la Nueva Base de Datos

Diagrama

Descripción generada automáticamente

1. Subconsultas:
2. IN:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* La subconsulta interior obtiene las facturas de **John Doe** .
* Luego, la subconsulta intermedia obtiene los productos relacionados con esas facturas.
* Finalmente, la consulta exterior lista los nombres de esos productos.

1. EXISTS:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* La subconsulta EXISTSverifica si una persona ha realizado una compra, es decir, si existe al menos una factura con esa persona como cliente.
* La consulta principal lista los nombres de dichas personas.

1. ANY:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* La subconsulta selecciona los precios de todos los productos en la categoría de **Mobiliario** .
* La consulta principal busca productos cuyo precio sea mayor que el precio de **cualquier** producto en esa categoría.

1. ALL:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* La subconsulta selecciona los precios de todos los productos en la categoría de **Mobiliario** .
* La consulta principal busca productos cuyo precio sea mayor que **todos** los precios en esa categoría, es decir, aquellos productos que sean más caros que el más caro de la categoría "Muebles".

1. Manipulación de cadena de textos:
2. CONCAT: Combina cadenas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* **CONCAT**: Combina el nombre del cliente, el número de documento y la ciudad en una sola columna llamada client\_info.
* **JOIN**:

Se hace un JOIN entre la tabla person y municipality para obtener la ciudad donde vive cada persona.

También se hace JOIN con person\_role y role para filtrar solo aquellos que tienen el rol de "Customer" (clientes).

* **WHERE**: Filtra solo las personas que tienen el rol de "Customer" (clientes).

1. LENGHT: Longitud de una cadena.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* **LENGTH**: Calcula la longitud del nombre del producto (cantidad de caracteres).
* **SELECT**: Muestra el nombre del producto y el valor calculado por LENGTH como name\_length.
* **FROM product**: La consulta trabaja sobre la tabla product.

1. UPPER: Convierte a mayúsculas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **UPPER**: Convierte el nombre del cliente a mayúsculas.
* **SELECT**: Muestra el nombre del cliente en mayúsculas con el alias customer\_name\_uppercase.
* **JOIN**: Se hace un JOIN con person\_role y role para obtener solo las personas que tienen el rol de "Customer".
* **WHERE**: Filtra solo las personas con el rol de "Customer" (clientes).

1. LOWER: Convierte a minúsculas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **LOWER**: Convierte el nombre del proveedor a minúsculas.
* **SELECT**: Muestra el nombre del proveedor en minúsculas con el alias provider\_name\_lowercase.
* **JOIN**: Se hace un JOIN con person\_role y role para obtener solo las personas que tienen el rol de "Provider".
* **WHERE**: Filtra solo las personas con el rol de "Provider" (proveedores).

1. SUBSTRING: Extrae parte de una cadena.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* **SUBSTRING(name, 1, 3)**: Extrae una subcadena de la columna name, comenzando desde el primer carácter y tomando los tres primeros.
* **SELECT**: Muestra el nombre del producto completo y la subcadena extraída como product\_code.
* **FROM product**: La consulta se realiza sobre la tabla product.

1. FUNCIONES MATEMATICAS EN SQL:
2. ROUND: Redondea un número.

Tabla

Descripción generada automáticamente con confianza media

* **ROUND(price, 2)**: Redondea el valor de la columna price a 2 decimales.
* **SELECT**: Muestra el nombre del producto, el precio original, y el precio redondeado como rounded\_price.
* **FROM product**: La consulta trabaja sobre la tabla product.

1. ABS: Valor absoluto: es decir, convierte números negativos en su equivalente positivo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* La consulta selecciona el id del producto, su nombre (product\_name), el stock actual (current\_stock), y la diferencia absoluta entre el stock y 50 unidades.
* Se utiliza ABS(i.stock - 50) para calcular la diferencia y asegurarse de que sea un valor positivo.

1. FUNCIONES AGREGADAS EN SQL
2. COUNT: Cuenta el número de filas que cumplen una condición

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* COUNT(p.id) cuenta cuántos productos están en cada tipo.
* Se usa GROUP BY para agrupar los productos por el tipo de producto (product\_type), y el resultado te dará una lista con el nombre del tipo de producto y la cantidad de productos asociados a ese tipo.

1. SUM: Suma los valores de una columna numérica.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* SUM(total) suma el valor de la columna total de todas las filas en la tabla invoice.
* Esto te devolverá el monto total de todas las ventas/facturas.

1. AVG: Calcula el promedio de los valores de una columna numérica.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **AVG(price)** calcula el promedio del valor de la columna **price** en la tabla **product**.
* Esto te devolverá el precio promedio de todos los productos que tienes en la tabla **product**.

1. MIN: Encuentra el valor mínimo en una columna

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **MIN(price)** obtiene el valor más bajo (mínimo) de la columna **price** en la tabla **product**.
* Esto te devolverá el precio más barato de todos los productos.

1. MAX: Encuentra el valor máximo en una columna.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **MAX(price)** obtiene el valor más alto (máximo) de la columna **price** en la tabla **product**.
* Esto te devolverá el precio más caro de todos los productos.

1. Agrupación de Resultados con GROUP BY

¿ bvInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **SELECT p.id\_municipality**: Selecciona el **municipio** de las personas. Se tomará esta columna de la tabla **person** para agrupar las facturas por municipio.
* **AVG(i.total)**: Calcula el promedio de los totales de las facturas de la columna **total** en la tabla **invoice**.
* **FROM invoice i**: Indica que los datos de las facturas están en la tabla **invoice**.
* **JOIN person p ON i.id\_customer = p.id**: Conecta la tabla **invoice** con la tabla **person** para obtener el municipio de la persona que realizó la factura, relacionando **id\_customer** en la tabla **invoice** con **id** en la tabla **person**.
* **GROUP BY p.id\_municipality**: Agrupa las facturas por municipio. Esto hace que el promedio del total se calcule por cada municipio diferente.

1. Filtrado de Grupos con HAVING

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* SELECT p.name, SUM(i.stock) as total\_stock: selecciona el nombre del producto y la suma total del inventario para cada producto.
* FROM product p JOIN inventory i ON p.id = i.id\_product: une la tabla de productos con la de inventario según el ID de producto.
* GROUP BY p.name: agrupa los resultados por el nombre de cada producto.
* HAVING total\_stock > 50: filtra los productos que tienen más de 50 unidades en stock.

1. Modificación de la base de datos agregándole nuevas tablas como users y users\_roles también store\_products y store:

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

* 1. Proceso:

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza baja

VISTAS, PROCEDIMIENTOS ALMACENADOS Y TRIGGERS

1. Esta vista incluye solo el nombre y apellido de los clientes, junto con su correo electrónico en users.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Resultado de la vista:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

* **peoples**: Obtenemos los campos name1y last\_name1de esta tabla para el nombre y apellido del cliente.
* **users**: Usamos emailesta tabla para mostrar el correo electrónico del cliente.
* **Relación** : Las tablas se relacionan usando p.id\_user = u.id.

1. Esta vista muestra las facturas generadas por cada cliente, incluyendo el total de la factura, su nombre y su documento. Esto Facilita el análisis de las facturas generadas por cada cliente.

Tabla

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteResultado de la vista :

1. Esta vista muestra un desglose de los productos vendidos, incluyendo cantidad, precio unitario y subtotal por factura.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Resultado de la vista:

Tabla

Descripción generada automáticamente

1. **trigger**
2. se activa al insertar un nuevo registro en la tabla invoices. Este disparador actualizará automáticamente el campo last\_updatede la tabla stores cada vez que se inserte una nueva factura. Esto puede ser útil para llevar un control de cuándo fue la última vez que se realizó una venta en la tienda.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Trigger** : update\_last\_update\_on\_new\_invoicese activa después de una inserción ( AFTER INSERT) en la tabla invoices.
* **Tabla Afectada** : stores- se actualizará el campo last\_updatecon la fecha y hora actuales ( NOW()).
* **Condición** : Solo se actualiza la tienda correspondiente al vendedor ( id\_seller) especificado en la nueva factura.

1. Trigger para actualizar el total de la factura al insertar productos,

Este disparador recalcula el total de la factura cada vez que se inserta un producto en la tabla **invoice\_products**. Mantiene el total actualizado automáticamente al agregar productos a una factura.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Este trigger crea automáticamente un registro en la tabla **payments**al insertar una nueva factura en **invoices**, suponiendo que el pago inicial es el total de la factura.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

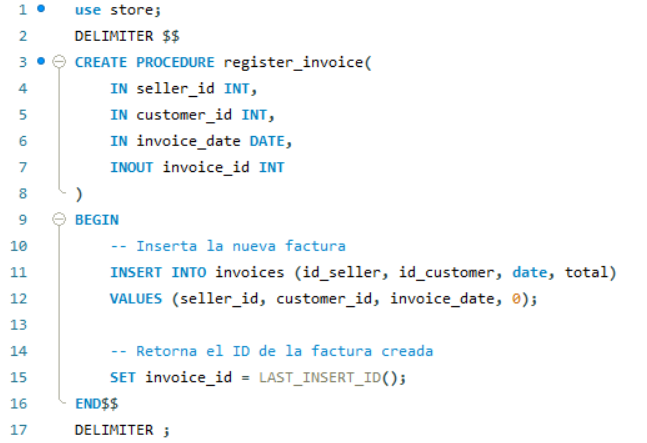
1. **procedimiento almacenado**
2. inserta un nuevo producto en la tabla products. Este procedimiento recibe como parámetros el nombre, descripción, precio y tipo de producto, e inserta estos valores en la tabla.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **Parámetros de Entrada (IN)** : Los parámetros product\_name, product\_description, product\_pricey product\_type\_idson datos que se pasan al procedimiento para insertar un nuevo producto.
* **Inserción** : El procedimiento ejecuta una instrucción INSERT INTOen la tabla products, usando los valores proporcionados en los parámetros.
* **DELIMITER** : Cambia el delimitador para definir correctamente el bloque de código.

1. Este procedimiento inserta una nueva factura y, opcionalmente, permite agregar productos asociados a esa factura.



1. procedimiento almacenado para agregar un nuevo método de pago a la base de

datos:

Interfaz de usuario gráfica, Texto, Aplicación

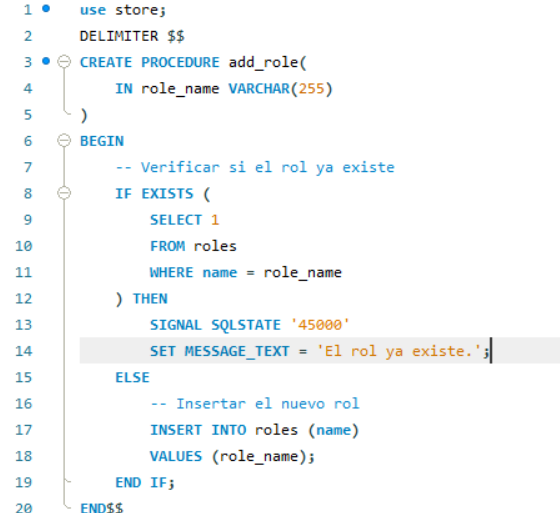
Descripción generada automáticamente

**Entrada** : El procedimiento recibe como parámetro el nombre del método de pago ( payment\_method\_name).

**Verificación** : Comprueba si ya existe un método de pago con el mismo nombre en la tabla **payment\_methods**.

**Inserción** : Si no existe, inserte el nuevo método en la tabla.

1. procedimiento almacenado para agregar un nuevo rol a la base de datos:



**Entrada** : Recibe como parámetro el nombre del rol ( role\_name).

**Verificación** : Comprueba si el rol ya existe en la tabla **roles**.

**Inserción** : Si no existe, lo inserta en la tabla **roles**.

## Diseño de base de datos

## Normalización:

La base de datos parece cumplir los principios de normalización hasta la tercera forma normal ( 3NF ). Aquí se explican los pasos seguidos:

Primera Forma Normal (1NF):

Cada tabla tiene un identificador único (clave primaria).

Todos los atributos contienen valores atómicos (sin repeticiones ni conjuntos de datos multivalorados).

Segunda Forma Normal (2NF):

Todas las tablas están relacionadas con claves primarias que identifican cada registro de manera única.

No hay dependencias parciales en tablas con claves compuestas (aunque no se detectan claves compuestas en este caso).

Tercera Forma Normal (3NF):

No hay dependencias transitivas, ya que los atributos no clave dependen directamente de la clave primaria.

## Consideraciones de diseño

**a) Elección de Claves Primarias:**

En cada tabla se selecciona como clave primaria ( PK) los identificadores únicos que cumplen los siguientes requisitos:

* Tipo: Todas las claves primarias son enteros ( INT), lo cual es ideal para mantener el rendimiento y facilitar la indexación.
* Auto incremental: Se supone que muchas de las claves primarias ( id) son auto incrementales ( AUTO\_INCREMENTen MySQL), lo que asegura que no haya duplicados.
* Significado: Las claves no contienen valores semánticos para evitar problemas de redundancia y facilitar cambios futuros.

Ejemplo:

* En la tabla users, la clave primaria es id, mientras que emailno se usa como clave porque podría cambiar con el tiempo.

**b) Relaciones entre Tablas:**

Se diseñaron relaciones basadas en el contexto lógico de la base de datos:

* **Relaciones 1:N:**
  + Un país puede tener múltiples departamentos ( countries → departments).
  + Un departamento puede tener múltiples municipios ( departments → municipalities).
  + Un usuario puede generar múltiples facturas ( users → invoices).
* **Relaciones N:1:**
  + Las facturas están relacionadas con vendedores y clientes, ambos representados en la tabla users.
  + Los productos pertenecen a una sola categoría ( products → product\_types).
* **Relaciones N:M:**
  + La tabla users\_rolesmodela la relación entre usuarios y roles, permitiendo que un usuario tenga múltiples roles y que un rol se asigne a varios usuarios.

**c) Índices y Optimizaciones:**

* **Índices Primarios:** Se definen automáticamente en las claves primarias ( PK).
* **Índices Foráneos:** Las claves foráneas ( FK) también están indexadas para optimizar las consultas de unión ( JOIN) entre tablas relacionadas.
* **Consideración adicional:** Si la cantidad de datos crece considerablemente, se podrían agregar índices secundarios en campos muy consultados, como emailen users.

**d) Integridad Referencial:**

* Las relaciones están aseguradas mediante claves foráneas ( FK), que garantizan que no existen registros huérfanos en tablas relacionadas.   
  Ejemplo:
  + No se puede agregar un municipio a un departamento inexistente.
  + No se pueden agregar productos en una factura si el producto no existe en la tabla products.

**e) Diseño Modular:**

La base de datos está diseñada de manera modular, separando los datos por categorías para facilitar:

* **Escalabilidad:** Se pueden añadir nuevas tablas (por ejemplo, para más tipos de pago) sin alterar la estructura principal.
* **Mantenibilidad:** Si necesitas hacer cambios en una parte (por ejemplo, en productos), no afecta a los demás.

# Análisis y discusiones

**Interpretación de resultados**

El diseño de la base de datos y su estructura refleja una organización sólida y modular que cumple con los principios fundamentales de bases de datos relacionales. Aquí se analizan los aspectos más relevantes y cómo estos contribuyen al cumplimiento de los objetivos planteados:

1. **Relaciones Definidas Claramente:**
   * Las relaciones establecidas entre las tablas ( 1:N, N:M, etc.) permiten modelar con precisión las conexiones entre los datos del sistema, como las facturas vinculadas a clientes y vendedores o los productos categorizados.
   * Esto cumple con el objetivo de diseñar una base de datos funcionales que soporte las operaciones clave de la tienda virtual.
2. **Normalización Efectiva:**
   * La estructura evita redundancias y asegura que cada tabla almacene únicamente los datos necesarios. Por ejemplo, los datos de los usuarios están separados de sus roles, y las facturas gestionan sus productos a través de tablas intermedias.
   * Esto mejora el rendimiento y facilita la escalabilidad, cumpliendo con el objetivo de optimizar el diseño.
3. **Integridad Referencial Garantizada:**
   * Las claves extranjeras aseguran que las relaciones sean consistentes. Por ejemplo, no es posible registrar un producto en una factura si no existe previamente en el catálogo. Esto previene errores y datos huérfanos.
   * Este enfoque está alineado con el objetivo de construir un sistema robusto y confiable.
4. **Flexibilidad para consultas:**
   * La estructura permite ejecutar consultas SQL eficientes para obtener datos esenciales, como:
     + Listar todas las facturas de un cliente en un rango de fechas.
     + Verificar el inventario de productos por tienda.
     + Generar informes de ventas por método de pago.

* **Objetivo Principal:** Diseñar una base de datos que permita gestionar una tienda virtual.
  + Se logró al incluir tablas y relaciones que soportan funciones como la gestión de usuarios, facturas, productos y pagos.
* **Optimización y Escalabilidad:**
  + La base de datos es escalable gracias a su diseño modular y normalización. Es posible añadir nuevas funcionalidades (por ejemplo, descuentos o promociones) sin afectar las estructuras existentes.

# Conclusión

**Diseño Bien Estructurado:**  
El diseño de la base de datos sigue los principios de normalización y asegura integridad referencial, lo cual garantiza un sistema eficiente y libre de redundancias.

**Relaciones Claras y Funcionales:**  
Las relaciones correctamente definidas permiten representar las operaciones clave del negocio, como facturación, manejo de inventarios y registro de pagos.

**Escalabilidad y Mantenimiento:**  
El modelo modular facilita el mantenimiento y futuras expansiones del sistema sin necesidad de reestructurarlo significativamente.

**Eficiencia en las Consultas:**  
Gracias al diseño optimizado, es posible ejecutar consultas complejas con eficiencia, lo cual contribuye al buen desempeño del sistema.

# Recomendaciones

**Pruebas de rendimiento:**  
Realizar pruebas de carga y optimización para verificar el desempeño de la base de datos con grandes volúmenes de datos. Esto incluye:

Escenarios simulares de alta concurrencia (varios usuarios ejecutando consultas simultáneamente).

Analizar tiempos de operaciones de respuesta en críticas, como generación de facturas o búsquedas en el inventario.

Ajustar configuraciones del servidor de base de datos (por ejemplo, caché, tamaños de tablas temporales).

**Índices adicionales:**  
Añadir índices secundarios en campos consultados frecuentemente, como:

Email en la tabla users, ya que es probable que se use para autenticación o búsquedas.

dateen la tabla invoices para consultas basadas en rangos de fechas.

Name en la tabla products para búsquedas rápidas de productos específicos.

**Validación de Datos:**  
Implementar restricciones y validaciones adicionales para evitar inconsistencias.