DOCUMENTACIÓN DE BASE DE DATOS: SISTEMA DE GESTIÓN DE INVENTARIOS

El sistema de gestión de inventarios tiene como objetivo gestionar productos y transacciones en diferentes almacenes de manera eficiente. La base de datos emplea una combinación de bases de datos relacionales para la gestión estructurada de los productos, usuarios y transacciones, mientras que también se contempla el uso de bases de datos no relacionales (como MongoDB) para almacenar registros históricos de transacciones y cambios en los productos.

- **Estructura de la Base de Datos**

La base de datos está formada por las siguientes tablas principales:

- **Almacen**: Contiene información sobre los almacenes donde se almacenan los productos.
- **Producto**: Información detallada sobre los productos en inventario.
- **Roles**: Roles de usuario en el sistema (administrador, operador).
- **Usuarios**: Información de los usuarios que acceden al sistema.
- **Transacciones**: Registra las transacciones de productos (ventas simuladas).
- **Detalle_Transaccion**: Detalles de cada transacción, especificando los productos vendidos.

- **Esquema de la Base de Datos**

Tabla: Almacen

Contiene la información sobre cada almacén.

Columna	Tipo de Dato	Descripción
id_almacen	INT	Clave primaria.
nombre	VARCHAR(100)	Nombre del almacén.
ubicacion	VARCHAR(55)	Dirección del almacén.

Tabla: Producto

Contiene la información sobre los productos disponibles en el inventario.

Columna	Tipo de Dato	Descripción
id_producto	INT	Clave primaria.
codigo	VARCHAR(50)	Código único del producto.
nombre	VARCHAR(255)	Nombre del producto.
descripcion	TEXT	Descripción detallada del producto.
precio	DECIMAL(10,2)	Precio unitario del producto.
stock	INT	Stock disponible.
id_almacen	INT	Clave foránea a la tabla **Almacen**.

Tabla: Roles

Define los roles de usuario en el sistema.

Columna	Tipo de Dato	Descripción
id_rol	INT	Clave primaria.
nombre	VARCHAR(50)	Nombre del rol (administrador, operador).

Tabla: Usuarios

Contiene los datos de los usuarios del sistema.

Columna	Tipo de Dato	Descripción
id_usuario	INT	Clave primaria.
nombre	VARCHAR(100)	Nombre del usuario.
email	VARCHAR(100)	Correo electrónico del usuario.
psswrd	VARCHAR(100)	Contraseña del usuario (cifrada).
id_rol	INT	Clave foránea a la tabla **Roles**.

Tabla: Transacciones

Registra cada transacción de venta (simulada).

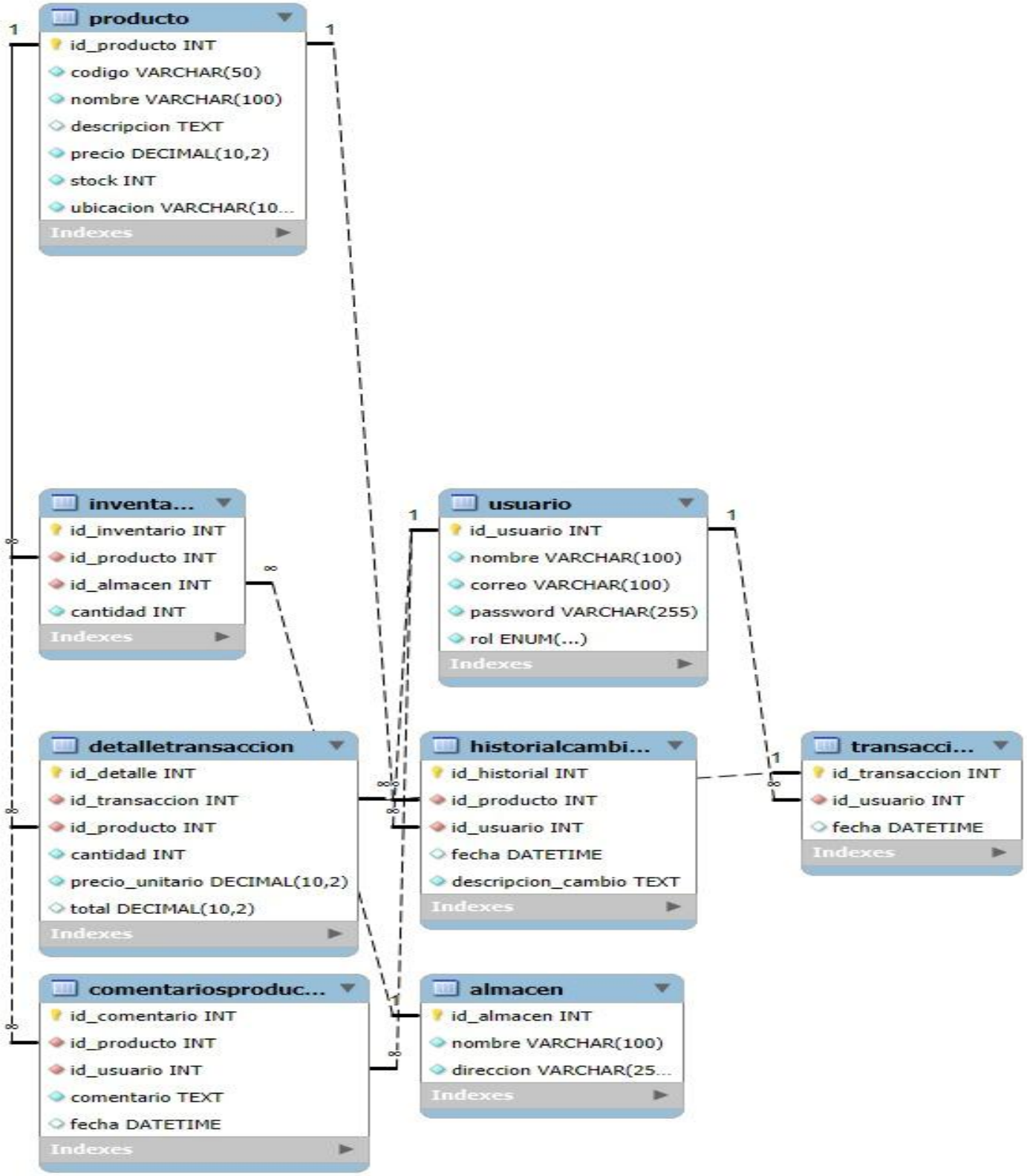
Columna	Tipo de Dato	Descripción
id_transaccion	INT	Clave primaria.
cantidad	INT	Cantidad de productos vendidos.
fecha	DATETIME	Fecha y hora de la transacción.
total	DECIMAL(10,2)	Total de la transacción (calculado).
id_producto	INT	Clave foránea a la tabla **Producto**.

Tabla: Detalle_Transaccion

Contiene el detalle de cada transacción.

Columna	Tipo de Dato	Descripción
id_detalle	INT	Clave primaria.
id_transaccion	INT	Clave foránea a la tabla **Transacciones**.
id_producto	INT	Clave foránea a la tabla **Producto**.
cantidad	INT	Cantidad del producto vendido.
precio_unitario	DECIMAL(10,2)	Precio unitario del producto.
subtotal	DECIMAL(10,2)	Subtotal de la transacción (cantidad * precio).

Diagramas E-R



- **Relaciones entre Tablas**

- **Almacen ↔ Producto**: Relación de uno a muchos (un almacén puede tener varios productos).
- **Producto ↔ Transacciones**: Relación de uno a muchos (un producto puede estar presente en múltiples transacciones).
- **Transacciones ↔ Detalle_Transaccion**: Relación de uno a muchos (una transacción puede tener varios detalles de productos vendidos).
- **Roles ↔ Usuarios**: Relación de uno a muchos (un rol puede ser asignado a varios usuarios).

- **Procedimientos y Triggers**

Trigger: calcular_total_venta

- **Descripción**: Este trigger se ejecuta antes de insertar una nueva transacción. Calcula el total de la venta (cantidad * precio unitario) y lo inserta en el campo total de la tabla **Transacciones**.

```
delimiter $$
create trigger calcular_total_venta before insert on Transacciones
for each row
begin
   declare precio_unitario decimal(10,2);

   select precio into precio_unitario
   from Producto
   where id_producto = NEW.id_producto;

   set NEW.total = NEW.cantidad * precio_unitario;
end;
$$
```

Trigger: verificar_stock

- **Descripción**: Este trigger se ejecuta antes de insertar un detalle de transacción. Verifica si el stock disponible es suficiente para la cantidad de producto que se va a vender. Si no hay suficiente stock, se genera un error.

```
delimiter $$
create trigger verificar_stock before insert on Detalle_Transaccion
for each row
begin
   declare stock_actual int;

   select stock into stock_actual
   from Producto
   where id_producto = NEW.id_producto;

   if stock_actual < NEW.cantidad then
      signal sqlstate '45000'
      set message_text = 'Error: Stock insuficiente para realizar la venta';
   end if;
end;
$$
```

Trigger: actualizar_stock

- **Descripción**: Este trigger se ejecuta después de insertar un detalle de transacción. Actualiza el stock disponible de los productos vendidos.

```
delimiter $$
create trigger actualizar_stock after insert on Detalle_Transaccion
```

```
for each row
begin
   update Producto
   set stock = stock - NEW.cantidad
   where id_producto = NEW.id_producto;
end;
$$
```

- **Funcionalidades Principales**

Gestión de Productos

- Registro de nuevos productos con los campos: código, nombre, descripción, precio, stock y ubicación en el almacén.
- Actualización y eliminación de productos.
- Verificación de duplicidad en los productos.

Consultas y Reportes Avanzados

- **Reporte de Inventario General**: Muestra código, nombre, stock actual, precio unitario y valor total de los productos.
- **Reporte de Productos por Ubicación**: Agrupa los productos por almacenes.
- **Reporte de Ventas Simuladas**: Muestra detalles de las transacciones realizadas, como los productos vendidos, cantidad, fecha y valor total.

Replicación y Partición de Datos

- **Replicación de Datos**: Configuración para garantizar alta disponibilidad.
- **Partición Horizontal**: Separación de datos de inventarios según almacenes.
- **Partición Vertical**: División de tablas grandes por frecuencia de acceso a las columnas.

Gestión Segura de Usuarios

- Roles definidos para usuarios (administrador, operador).
- Permisos avanzados para limitar el acceso a operaciones específicas.

- **Almacenamiento de Datos No Relacionales**

Se utiliza **MongoDB** para almacenar los siguientes registros históricos:

- **Transacciones pasadas**: Simuladas a través de una API.
- **Historial de cambios de productos**.
- **Comentarios de operadores sobre productos**.