Federico Di Iorio

ENTREGA FINAL SQL

Control de inventario

Índice

Introducción

Muchas empresas necesitan un sistema que les permita administrar su inventario de productos, registrar las ventas y llevar un control de los clientes. Sin una estructura organizada, pueden surgir problemas como:

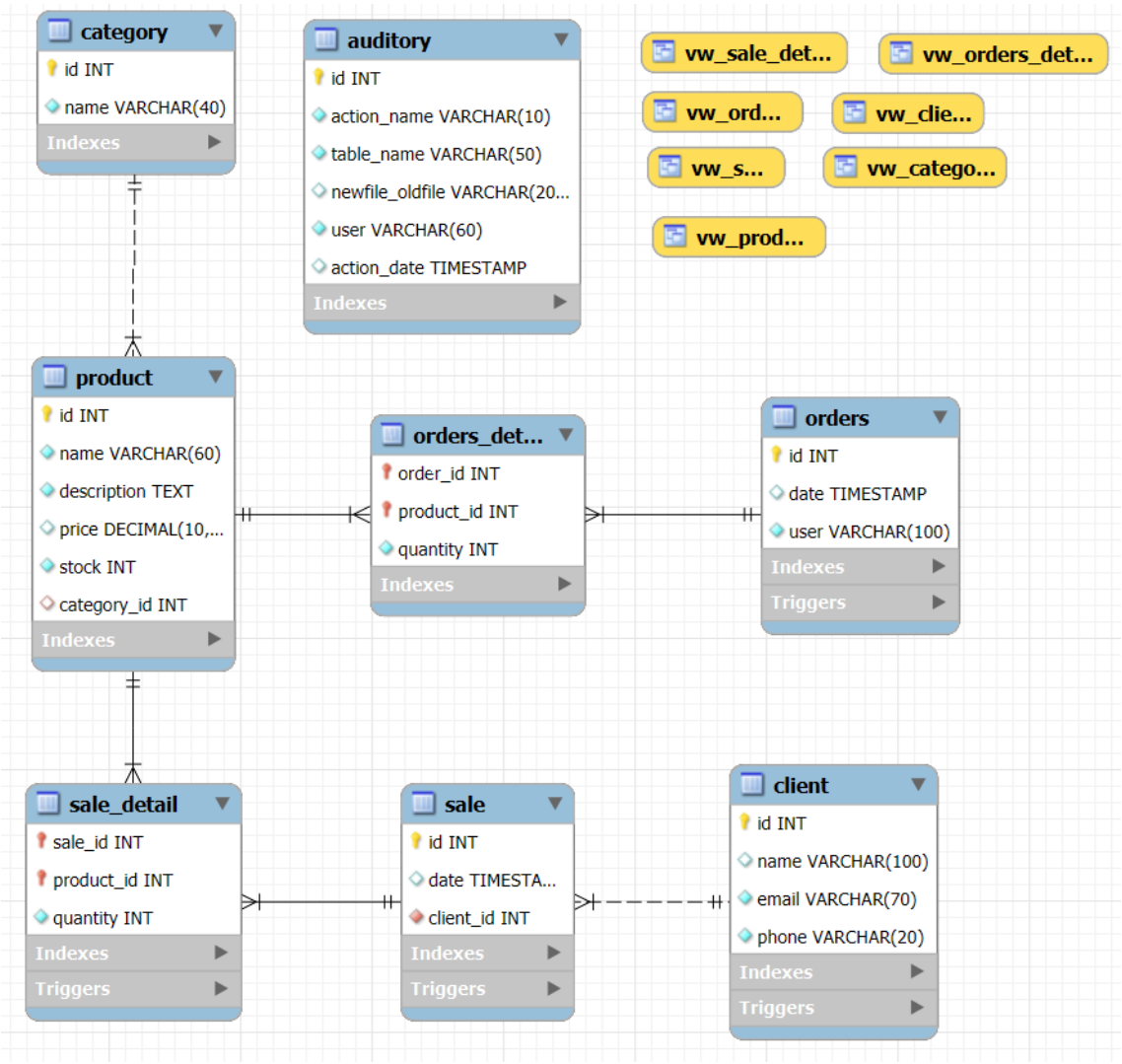
- Falta de control de stock, lo que puede llevar a vender productos no disponibles.
- Dificultad para gestionar las categorías de productos, impidiendo una búsqueda y organización eficiente.
- Registro inconsistente de ventas, dificultando el seguimiento de compras de los clientes.
- Falta de trazabilidad en los detalles de venta, impidiendo conocer qué productos se vendieron en cada transacción.
- Desorden en el ingreso de nuevos productos al inventario, lo que puede generar errores en la cantidad de stock disponible.

Objetivo

Este sistema busca ofrecer una solución a los problemas mencionados mediante:

- La organización de los productos en diferentes categorías para facilitar su gestión.
- El registro detallado de cada producto, incluyendo su precio, cantidad disponible y descripción.
- La administración de la información de los clientes para asegurar que cada compra esté correctamente asociada.
- El control de las ventas y su historial, permitiendo llevar un registro preciso de las transacciones realizadas.
- La descomposición de cada venta en sus elementos individuales, garantizando un seguimiento claro de los productos adquiridos y sus cantidades.
- **Un proceso estructurado para registrar la llegada de nuevos productos al inventario, asegurando que las cantidades ingresadas sean precisas y actualizadas.**

Diagrama Entidad Relación



Lista de tablas

Una tabla es una estructura utilizada para organizar y almacenar información de manera ordenada, permitiendo registrar y consultar datos de forma clara y accesible.

Categorías

Objetivo: Permite organizar los productos en distintos grupos según sus características, facilitando su clasificación y búsqueda.

CATEGORY			
CAMPO	TIPO DE DATO	PK	FK
id	Número entero	X	
name	Texto (40)		

- **Id:** Identificador único auto incremental.
- **Name:** Nombre de la categoría (hasta 40 caracteres).

Productos

Objetivo: Almacena información sobre cada producto, incluyendo su descripción, precio y cantidad disponible, para gestionar el inventario de manera eficiente.

PRODUCT			
CAMPO	TIPO DE DATO	PK	FK
id	Número entero	X	
name	Texto (40)		
description	Texto		
price	Número decimal (10,2)		
stock	Número entero		
category_id	Número entero		X

- **Id:** Identificador único auto incremental.
- **Name:** Nombre del producto (hasta 60 caracteres).
- **Description**: Descripción detallada del producto.
- **Price:** Precio del producto, debe ser mayor a 0.01.
- **Stock:** Cantidad disponible, no puede ser negativa.
- **Category_id:** Identificador de la categía a la que pertenece el producto (opcional)

Clientes

Objetivo: Permite almacenar la información de las personas que realizan compras para su identificación y contacto.

CLIENT			
CAMPO	TIPO DE DATO	PK	FK
id	Número entero	X	
name	Texto (100)		
email	Texto (70)		
phone	Texto (20)		

- **Id:** Identificador único auto incremental.
- **Name:** Nombre del cliente (hasta 100 caracteres y valor por defecto: "Sin Nombre").
- **Email:** Dirección de correo electrónico única (hasta 70 caracteres).
- **Phone:** Número de teléfono único (hasta 20 caracteres).

Ventas

Objetivo: Registra cada compra realizada, permitiendo un seguimiento de las transacciones y su relación con los clientes.

SALE			
CAMPO	TIPO DE DATO	PK	FK
id	Número entero	X	
date	TIMESTAMP		
client_id	Número entero		X

- **Id:** Identificador único auto incremental.
- **Date:** Fecha y hora en que se realizó la venta (se genera automáticamente).
- **Client_id:** Identificador del cliente asociado a la venta.

Detalle de venta

Objetivo: Permite registrar los productos incluidos en cada venta, junto con la cantidad adquirida.

SALE_DETAIL			
CAMPO	TIPO DE DATO	PK	FK
sale_id	Número entero		X
product_id	Número entero		X
quantity	Número entero		

- **Sale_id:** Identificador de la venta a la que pertenece el detalle.
- **Product_id:** Identificador del producto vendido.
- **Quantity:** Cantidad del producto en la venta (mínimo 1).

Órdenes de compra

Objetivo: Permite registrar solicitudes de reposición de productos para mantener un inventario adecuado.

ORDERS			
CAMPO	TIPO DE DATO	PK	FK
order_id	Número entero	X	
date	fecha		
user	Texto		

- **Id:** Identificador único auto incremental.
- **Date:** Fecha y hora en que se realizó la orden (se genera automáticamente).
- **User:** Nombre del usuario que generó la orden.

Detalles de órdenes de compra

Objetivo: Permite registrar los productos solicitados en cada orden de compra, asegurando un control adecuado del stock ingresado.

ORDERS_DETAIL			
CAMPO	TIPO DE DATO	PK	FK
order_id	Número entero		X
product_id	Número entero		X
quantity	Número entero		X

- **Order_id:** Identificador de la orden a la que pertenece el detalle.
- **Product_id:** Identificador del producto solicitado.
- **Quantity:** Cantidad solicitada del producto (mínimo 1).

Registro de actividad

Objetivo: Mantiene un historial de las modificaciones realizadas en el sistema, garantizando transparencia y trazabilidad.

AUDITORY			
CAMPO	TIPO DE DATO	PK	FK
id	Número entero	X	
action_name	Texto (10)		
table_name	Texto (50)		
newfile_oldfile	Texto (200)		
user	Texto (60)		
action_date	TIMESTAMP		

- **Id:** Identificador único auto incremental.
- **Action_name:** Tipo de acción realizada (ejemplo: inserción, actualización, eliminación).
- **Table_name:** Nombre del elemento modificado.
- **Newfile_oldfile:** Información relevante sobre el cambio.
- **User:** Nombre del usuario que realizó la acción.
- **Action_date:** Fecha y hora en que se efectuó el cambio (se genera automáticamente).

Listado de Vistas

Una vista es una forma de presentar información almacenada en una base de datos de manera organizada y simplificada. Permite mostrar solo los datos relevantes sin necesidad de acceder directamente a las tablas originales.

Las vistas pueden usarse para:

- Consultar información de manera más clara y estructurada.
- Restringir el acceso a ciertos datos, mostrando solo la información necesaria.
- Facilitar la creación de reportes sin alterar la estructura de la base de datos.

En términos simples, una vista es como una "ventana" a los datos existentes, que permite ver la información de una manera más conveniente sin modificar su almacenamiento.

Vista: vw_category

- **Objetivo**: Muestra todas las categorías de productos registradas, facilitando su consulta sin necesidad de acceder a la información completa de la base de datos.
- **Tablas involucradas**: category.

Vista: vw_product

- **Objetivo**: Presenta los productos disponibles junto con su información principal, como el nombre, la descripción, el precio y el stock. Además, incluye la categoría a la que pertenece cada producto, lo que facilita su organización.
- **Tablas involucradas**: product, category (para obtener el nombre de la categoría de cada producto).

Vista: vw_sale

- **Objetivo**: Muestra un resumen de cada venta realizada, incluyendo la fecha, el cliente que realizó la compra y el monto total de la venta. Esto ayuda a visualizar el historial de ventas de manera más clara.
- **Tablas involucradas**: sale, client, sale_detail, product (para calcular el total de cada venta).

Vista: vw_sale_detail

- **Objetivo**: Permite ver en detalle qué productos fueron comprados en cada venta, cuántas unidades se adquirieron y el subtotal correspondiente. Esto es útil para analizar qué productos se venden con más frecuencia.
- **Tablas involucradas**: sale_detail, product.

Vista: vw_client

- **Objetivo**: Facilita la consulta de los clientes registrados, mostrando su información básica junto con el número total de compras realizadas y la cantidad de productos adquiridos.
- **Tablas involucradas**: client, sale, sale_detail (para calcular el total de compras y productos comprados por cada cliente).

Vista: vw_orders

- **Objetivo**: Proporciona un resumen de los pedidos de stock realizados, incluyendo la fecha en la que se hicieron y el usuario que los registró. Esto permite un mejor control sobre el reabastecimiento de productos.
- **Tablas involucradas**: orders.

Vista: vw_orders_detail

- **Objetivo**: Muestra el detalle de cada pedido de stock, indicando los productos solicitados, la cantidad pedida y el costo total. Esto facilita la gestión del inventario y el control de nuevas adquisiciones.
- **Tablas involucradas**: orders_detail, product.

Listado de Funciones

Una función en SQL es un bloque de código que realiza una tarea específica y devuelve un resultado. Se utiliza para calcular valores a partir de los datos almacenados en la base de datos y puede ser reutilizada en múltiples consultas. A diferencia de los procedimientos almacenados, una función siempre devuelve un valor y no puede modificar los datos de las tablas.

Función: fn_get_top_client

- **Objetivo**: Devuelve el cliente con más compras realizadas, indicando su identificador, nombre y la cantidad de compras efectuadas. Esto permite identificar a los clientes más frecuentes.
- **Tablas involucradas**: clients, sales.

Función: fn_get_client_spent

- **Objetivo**: Calcula el total de dinero gastado por un cliente específico en todas sus compras. Si el cliente no ha realizado compras, devuelve 0. Esta función es útil para analizar el comportamiento de compra de cada cliente.
- **Tablas involucradas**: sales, sales_detail, products.

Listado de Procedimientos Almacenados

Un procedimiento almacenado es un conjunto de instrucciones SQL predefinidas que se almacenan en la base de datos y pueden ejecutarse cuando se necesiten. Se utilizan para automatizar tareas repetitivas, mejorar la eficiencia de las consultas y garantizar la integridad de los datos. A diferencia de las funciones, los procedimientos almacenados pueden modificar los datos y no están obligados a devolver un valor.

Procedimiento: sp_discount_stock

- **Objetivo**: Reduce la cantidad de stock de un producto específico en función de la cantidad solicitada. Si el stock disponible es insuficiente, lanza un error.
- **Tablas involucradas**: product.

Procedimiento: sp_add_stock

- **Objetivo**: Aumenta el stock de un producto en la cantidad especificada. Se utiliza cuando se recibe nueva mercadería.
- **Tablas involucradas**: product.

Procedimiento: sp_create_sale

- **Objetivo**: Registra una nueva venta en la base de datos y devuelve el ID de la venta generada.
- **Tablas involucradas**: sale.

Procedimiento: sp_process_sale

- **Objetivo**: Procesa una venta descontando el stock del producto y registrando los detalles de la venta. Si el stock no es suficiente, la operación se cancela.
- **Tablas involucradas**: sales, sales_detail, product.

Procedimiento: sp_create_order

- **Objetivo**: Registra un nuevo pedido en la base de datos y devuelve el ID del pedido generado.
- **Tablas involucradas**: orders.

Procedimiento: sp_process_order

- **Objetivo**: Procesa un pedido, aumentando el stock del producto y registrando los detalles del pedido.
- **Tablas involucradas**: orders, orders_detail, product.

Listado de Triggers

Un **trigger** es un mecanismo de la base de datos que permite ejecutar automáticamente una acción cuando ocurre un evento específico en una tabla, como una inserción una actualización o una eliminació. Los triggers se utilizan para mantener la integridad de los datos, auditar cambios y automatizar procesos sin necesidad de intervención manual.

Trigger: tr_insert_client

- Objetivo: Registra en la tabla auditory cada vez que se inserta un nuevo cliente en la tabla client, almacenando su ID y el usuario que realizó la acción.
- Evento activador: AFTER INSERT en client.
- Tablas involucradas: client, auditory.

Trigger: tr_update_sale

- Objetivo: Registra los cambios realizados en la tabla sale, almacenando los valores antiguos y nuevos de la venta, incluyendo la fecha y el ID del cliente.
- Evento activador: AFTER UPDATE en sale.
- Tablas involucradas: sale, auditory.

Trigger: tr_update_sales_details

- Objetivo: Registra los cambios en los detalles de ventas (sales_detail), incluyendo cambios en el ID de venta, ID del producto y cantidad vendida.
- Evento activador: AFTER UPDATE en sales_detail.
- Tablas involucradas: sales_detail, auditory.

Trigger: tr_before_delete_sale

- Objetivo: Antes de eliminar una venta (sale), guarda en auditory los detalles de la venta que serán eliminados por el CASCADE.
- Evento activador: BEFORE DELETE en sale.
- Tablas involucradas: sale, sales_detail, auditory.

Trigger: tr_delete_sale

- Objetivo: Registra en auditory la eliminación de una venta, guardando su ID, fecha y el ID del cliente asociado.
- Evento activador: AFTER DELETE en sale.
- Tablas involucradas: sale, auditory.

Trigger: tr_before_insert_order

- Objetivo: Antes de insertar un nuevo pedido en la tabla orders, asigna automáticamente el usuario que realiza la operación (CURRENT_USER()).
- Evento activador: BEFORE INSERT en orders.
- Tablas involucradas: orders.

Herramientas utilizadas

Para llevar adelante este proyecto y este informe se utilizaron las siguientes herramientas:

- MySQL Workbench
- GitHub
- Herramientas de office