ENTREGA Nº 2

Control de inventario

Problemática

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.

🔹 **Cómo lo soluciona**

Este esquema permite:
Categorizar productos en la tabla categories, facilitando la organización.

Registrar productos con su precio, stock y descripción en la tabla products.

Gestionar clientes en la tabla clients, asegurando que cada venta esté asociada a un cliente específico.

Controlar ventas y su historial en la tabla sales, con un registro de la fecha y el cliente asociado.

Desglosar cada venta en detalle en la tabla sales_detail, asegurando un control preciso de los productos vendidos y sus cantidades.

Diagrama Entidad Relación

PK: Color verde

FK: Color rojo

Lista de tablas

CATEGORIES			
CAMPO	TIPO DE DATO	PK	FK
id	INT	X	
name	VARCHAR (40)		

PRODUCTS			
CAMPO	TIPO DE DATO	PK	FK
id	INT	X	
name	VARCHAR (40)		
description	TEXT		
price	DECIMAL (10,2)		
stock	INT		
category_id	INT		X

CLIENTS			
CAMPO	TIPO DE DATO	PK	FK
id	INT	X	
name	VARCHAR (100)		
email	VARCHAR (70)		
phone	VARCHAR (20)		

SALES			
CAMPO	TIPO DE DATO	PK	FK
id	INT	X	
date	TIMESTAMP		
client_id	INT		X

SALES_DETAIL			
CAMPO	TIPO DE DATO	PK	FK
sale_id	INT		X
product_id	INT		X
quantity	INT		

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

Listado de Vistas y Descripciones Detalladas

1. vw_categories

Objetivo: Esta vista muestra todas las categorías disponibles en la tabla categories.

Tablas:

- categories: Contiene las categorías de productos con sus respectivos IDs y nombres.

2. vw_products

Objetivo: Ofrece una lista detallada de todos los productos disponibles, incluyendo información de nombre, descripción, precio, stock y categoría a la que pertenecen.

Tablas:

- products: Almacena la información principal de los productos como nombre, descripción, precio y stock.
- categories: Relacionada mediante el campo category_id para obtener el nombre de la categoría de cada producto.

3. vw_sales

Objetivo: Proporciona un resumen de las ventas realizadas, incluyendo detalles como la fecha, cliente, y el total monetario de cada venta.

Tablas:

- sales: Contiene información básica de cada venta, como fecha y cliente asociado.
- clients: Para obtener detalles del cliente como nombre, email y teléfono.
- sales_detail: Detalle de los productos vendidos en cada venta.
- products: Información adicional de los productos para calcular el total de cada venta.

4. vw_sales_detail

Objetivo: Muestra los detalles de cada venta, incluyendo el nombre del producto, precio unitario, cantidad vendida y el subtotal por producto.

Tablas:

- sales_detail: Detalle de los productos vendidos en cada venta, incluyendo la cantidad vendida.
- products: Información de cada producto para obtener su nombre y precio.

5. vw_clients

Objetivo: Presenta un resumen de los clientes registrados, incluyendo su nombre, email, teléfono, total de compras realizadas y cantidad total de productos comprados.

Tablas:

- clients: Información básica de los clientes como nombre, email y teléfono.
- sales: Para contar el número total de compras realizadas por cada cliente.
- sales_detail: Para sumar la cantidad total de productos comprados por cada cliente.

Listado de Funciones

1. fn_get_top_client()

Objetivo:
Esta función devuelve el cliente con la mayor cantidad de compras realizadas. La información incluye el ID del cliente, su nombre y la cantidad de compras efectuadas.

Tablas manipuladas:

- clients: Para obtener los datos del cliente.
- sales: Para contar la cantidad de compras realizadas por cada cliente.

Funcionamiento:

- Se agrupan las ventas por cliente (GROUP BY c.id).
- Se ordenan en orden descendente según la cantidad de compras (ORDER BY COUNT(s.id) DESC).
- Se obtiene el cliente con más compras (LIMIT 1).
- Se retorna un string con el ID, nombre y cantidad de compras.

2. fn_get_client_spent(client_id INT)

Objetivo:
Esta función calcula el total de dinero gastado por un cliente específico en todas sus compras.

Tablas manipuladas:

- sales: Para identificar las compras realizadas por el cliente.
- sales_detail: Para obtener los productos y cantidades adquiridas en cada compra.
- products: Para obtener el precio de cada producto.

Funcionamiento:

- Se filtran las ventas del cliente (WHERE s.client_id = client_id).
- Se suman los montos de cada producto comprado (SUM(p.price * sd.quantity)).
- Se usa COALESCE para devolver 0 si el cliente no ha realizado compras.
- Retorna el total gastado por el cliente como un valor decimal.

Listado de Stored Procedures

1. sp_discount_stock(IN p_quantity INT, IN p_product_id INT)

Objetivo:
Este procedimiento actualiza el stock de un producto restando la cantidad vendida. Si el stock disponible es insuficiente, genera un error.

Beneficio para el proyecto:

- Garantiza que no se vendan más productos de los disponibles.
- Evita problemas de stock negativo mediante una validación.

Tablas con las que interactúa:

- products: Se actualiza la columna stock restando la cantidad vendida.

Funcionamiento:

- Se realiza un UPDATE sobre la tabla products restando p_quantity al stock del producto indicado por p_product_id.
- Si el stock es insuficiente, se genera un error con SIGNAL SQLSTATE '45000'.

2. sp_create_sale(IN p_client_id INT, OUT p_sale_id INT)

Objetivo:
Registra una nueva venta en la tabla sales y devuelve el ID generado para la venta.

Beneficio para el proyecto:

- Automatiza la inserción de una nueva venta asociada a un cliente.
- Permite obtener el ID de la venta para realizar otros procesos relacionados.

Tablas con las que interactúa:

- sales: Se inserta una nueva fila con el client_id del cliente que realiza la compra.

Funcionamiento:

- Se inserta una nueva fila en sales con el ID del cliente (p_client_id).
- Se obtiene el ID generado con LAST_INSERT_ID() y se asigna a p_sale_id.

3. sp_process_sale(IN p_sale_id INT, IN p_product_id INT, IN p_quantity INT)

Objetivo:
Procesa una venta en tres pasos:

1. Verifica y descuenta el stock llamando al procedimiento sp_discount_stock.
2. Inserta el detalle de la venta en la tabla sales_detail.
3. Confirma la transacción con COMMIT.

Beneficio para el proyecto:

- Garantiza que el stock se descuente antes de registrar la venta.
- Evita inconsistencias en caso de errores al procesar la venta.
- Mejora la integridad de los datos al utilizar transacciones.

Tablas con las que interactúa:

- products: Se actualiza el stock a través de sp_discount_stock.
- sales_detail: Se inserta el producto comprado con la cantidad correspondiente.

Funcionamiento:

- Se inicia una transacción (START TRANSACTION).
- Se llama al procedimiento sp_discount_stock para verificar y actualizar el stock.
- Se inserta el detalle de la venta en sales_detail.
- Se confirma la transacción con COMMIT.
- Se devuelve un mensaje indicando el éxito de la operación.

Listado de Triggers

1. tr_insert_client

Objetivo:
Este trigger registra en la tabla auditory cada vez que se inserta un nuevo cliente en la tabla clients, guardando el ID del cliente y el usuario que realizó la operación.

Tablas involucradas:

- clients: Se activa después de una inserción en esta tabla.
- auditory: Se inserta un registro con los detalles de la acción.

Funcionamiento:

- Se ejecuta después de una inserción en clients.
- Inserta un registro en auditory con la acción realizada, la tabla afectada y el ID del nuevo cliente.

2. tr_update_sale

Objetivo:
Este trigger registra en la tabla auditory los cambios en la tabla sales, almacenando los valores anteriores y nuevos de los campos id, date y client_id.

Tablas involucradas:

- sales: Se activa después de una actualización en esta tabla.
- auditory: Se inserta un registro con los detalles del cambio.

Funcionamiento:

- Se ejecuta después de actualizar un registro en sales.
- Guarda en auditory un mensaje con los valores anteriores y los nuevos de id, date y client_id.

3. tr_update_sales_details

Objetivo:
Este trigger registra en la tabla auditory los cambios en la tabla sales_detail, guardando los valores antiguos y nuevos del sale_id, product_id y quantity.

Tablas involucradas:

- sales_detail: Se activa después de una actualización en esta tabla.
- auditory: Se inserta un registro con los detalles del cambio.

Funcionamiento:

- Se ejecuta después de actualizar un registro en sales_detail.
- Registra en auditory los valores anteriores y nuevos de sale_id, product_id y quantity.

4. tr_before_delete_sale

Objetivo:
Antes de eliminar una venta de la tabla sales, este trigger guarda todos los detalles de la venta en la tabla auditory para evitar la pérdida de información en cascada.

Tablas involucradas:

- sales: Se activa antes de eliminar un registro en esta tabla.
- sales_detail: Se consultan los registros asociados a la venta eliminada.
- auditory: Se inserta un registro con los detalles de los productos eliminados.

Funcionamiento:

- Se ejecuta antes de eliminar un registro en sales.
- Busca los detalles de la venta en sales_detail y los guarda en auditory.

5. tr_delete_sale

Objetivo:
Este trigger registra en la tabla auditory cuando una venta es eliminada, guardando el id, la date y el client_id de la venta eliminada.

Tablas involucradas:

- sales: Se activa después de eliminar un registro en esta tabla.
- auditory: Se inserta un registro con los datos eliminados.

Funcionamiento:

- Se ejecuta después de eliminar un registro en sales.
- Inserta en auditory los detalles de la venta eliminada.