LISTADO DE VISTAS

En esta sección se detallan las vistas creadas en la base de datos “distribuidora”, indicando su objetivo y las tablas que intervienen en su construcción.

1. Vista: vw_detalle_pedidos

Descripción:

La vista vw_detalle_pedidos permite visualizar el detalle completo de cada pedido realizado en el sistema. Muestra información del pedido, del cliente y de los artículos incluidos, junto con la cantidad solicitada, el precio unitario y el subtotal calculado.

Objetivo

Su objetivo es simplificar la consulta de información relacionada a los pedidos, evitando realizar múltiples JOIN manualmente cada vez que se necesite analizar ventas. Facilita la generación de reportes comerciales y el control de operaciones.

Tablas involucradas

pedidos

clientes

detalle_pedido

articulos

2. Vista: vw_stock_articulos

Descripción:

La vista vw_stock_articulos muestra el stock actual de cada artículo junto con la categoría a la que pertenece.

Objetivo

Permite tener una visualización rápida del inventario disponible, facilitando el control de stock y la toma de decisiones relacionadas con reposición o planificación de compras.

Tablas involucradas

articulos

categorias

3. Vista: vw_total_pedidos_por_cliente

Descripción:

La vista vw_total_pedidos_por_cliente calcula el total acumulado de compras realizadas por cada cliente, sumando los subtotales de los artículos incluidos en sus pedidos.

Objetivo

Su finalidad es permitir el análisis del comportamiento de compra de los clientes, identificando aquellos con mayor volumen de operaciones y facilitando la gestión comercial.

Tablas involucradas

clientes

pedidos

detalle_pedido

Comentario final

Las vistas fueron diseñadas para centralizar consultas frecuentes y mejorar la legibilidad del sistema, evitando la repetición de consultas complejas y favoreciendo una estructura organizada de la base de datos.

LISTADO DE FUNCIONES

Se detallan las funciones creadas en la base de datos “distribuidora”, indicando su finalidad y las tablas sobre las que operan.

1. Función: fn_calcular_subtotal

Descripción:

La función fn_calcular_subtotal recibe como parámetros una cantidad y un precio unitario, y devuelve el resultado de la multiplicación entre ambos valores.

Objetivo

Su objetivo es encapsular el cálculo del subtotal de un artículo dentro de un pedido, evitando repetir la operación matemática en múltiples consultas y mejorando la claridad del código SQL.

Tablas involucradas

No modifica tablas directamente.
Se utiliza en conjunto con los datos de la tabla:

- detalle_pedido

2. Función: fn_total_pedido

Descripción:

La función fn_total_pedido recibe como parámetro el identificador de un pedido y devuelve el total del mismo, calculado como la suma de los subtotales de todos los artículos asociados.

Internamente realiza una consulta sobre los registros de detalle_pedido correspondientes al pedido indicado.

🎯 **Objetivo**

Permitir obtener el total de un pedido de manera automática y centralizada, garantizando consistencia en los cálculos y facilitando su reutilización en consultas o reportes.

Tablas involucradas

- detalle_pedido

3. Función: fn_stock_disponible

Descripción:

La función fn_stock_disponible recibe como parámetro el identificador de un artículo y devuelve el stock actual registrado en la base de datos.

Objetivo

Facilitar la consulta del stock disponible, pudiendo ser utilizada para validaciones o controles dentro de procedimientos almacenados.

Tablas involucradas

- articulos

Comentario final:

Las funciones fueron creadas con el objetivo de encapsular cálculos frecuentes dentro del sistema, promoviendo reutilización, claridad y mantenimiento simplificado del código.

LISTADO DE STORED PROCEDURES

Se detallan los procedimientos almacenados creados en la base de datos “distribuidora”, indicando su objetivo y las tablas sobre las cuales interactúan.

1. Stored Procedure: sp_actualizar_stock

Descripción:

El procedimiento almacenado sp_actualizar_stock permite modificar el stock de un artículo determinado, recibiendo como parámetros el identificador del artículo y el nuevo valor de stock.

Realiza una actualización directa sobre la tabla articulos.

Objetivo

Centralizar la lógica de actualización de stock en un único punto controlado, evitando modificaciones manuales directas sobre la tabla y favoreciendo una estructura más organizada del sistema.

Tablas involucradas

- articulos

2. Stored Procedure: sp_reporte_pedidos_por_fecha

Descripción

El procedimiento sp_reporte_pedidos_por_fecha permite obtener un listado de pedidos realizados dentro de un rango de fechas determinado. Recibe como parámetros una fecha de inicio y una fecha de fin.

Devuelve todos los pedidos cuya fecha se encuentre dentro del rango especificado.

Objetivo

Facilitar la generación de reportes comerciales y análisis de ventas en períodos específicos, permitiendo filtrar información sin necesidad de escribir consultas complejas cada vez.

Tablas involucradas

- pedidos

Comentario final:

Los Stored Procedures fueron implementados para encapsular operaciones frecuentes del sistema, promoviendo mayor orden, reutilización y control sobre las modificaciones y consultas realizadas en la base de datos.

LISTADO DE TRIGGERS

Se detallan los triggers implementados en la base de datos “distribuidora”, indicando su funcionalidad, el momento en que se ejecutan y las tablas involucradas.

1. Trigger: tr_descontar_stock

Descripción:

El trigger tr_descontar_stock se ejecuta automáticamente después de que se inserta un registro en la tabla detalle_pedido (evento AFTER INSERT).

Su función es descontar del stock del artículo la cantidad correspondiente al nuevo detalle de pedido ingresado.

Momento de ejecución

- AFTER INSERT

Sobre la tabla detalle_pedido

Objetivo

Garantizar que el stock se actualice automáticamente cada vez que se registra un artículo en un pedido, manteniendo la coherencia entre ventas e inventario sin depender de actualizaciones manuales.

Tablas involucradas

- detalle_pedido
- articulos

2. Trigger: tr_evitar_stock_negativo

Descripción:

El trigger tr_evitar_stock_negativo se ejecuta antes de que se actualice un registro en la tabla articulos (evento BEFORE UPDATE).

Verifica que el nuevo valor de stock no sea menor a cero. En caso contrario, genera un error y cancela la operación.

Momento de ejecución

- BEFORE UPDATE
- Sobre la tabla articulos

Objetivo

Evitar inconsistencias en el sistema, asegurando que el stock de un artículo nunca quede en valores negativos, lo cual representaría una situación inválida en términos comerciales.

Tablas involucradas

- articulos

Comentario final

Los triggers fueron implementados para reforzar la integridad de los datos y automatizar comportamientos clave del sistema, como la actualización del stock y el control de valores inválidos.

ARCHIVOS SQL ENTREGADOS

Se incluyen dos archivos en formato .sql, organizados de manera separada para mantener una estructura clara proyecto.

1. 01_creacion_objetos_distribuidora.sql

Contenido:

Este archivo contiene:

- Sentencias CREATE VIEW
- Sentencias CREATE FUNCTION
- Sentencias CREATE PROCEDURE
- Sentencias CREATE TRIGGER

Antes de la creación de los objetos se incluye la sentencia:

USE distribuidora;

Además, se incorporan sentencias DROP IF EXISTS para permitir que el script pueda ejecutarse múltiples veces sin generar errores por objetos ya existentes.

Objetivo:

Centralizar toda la lógica del sistema (consultas reutilizables, cálculos, automatizaciones y validaciones) en un único script ejecutable de forma independiente.

Este archivo crea la estructura lógica del sistema, pero no inserta datos.

2. 02_insercion_datos_distribuidora.sql

Contenido

Este archivo contiene sentencias INSERT INTO para todas las tablas del sistema:

- localidades
- categorias
- proveedores
- vendedores
- clientes
- estados_pedido
- formas_pago
- articulos

- pedidos
- detalle_pedido

También incluye la sentencia:

USE distribuidora;

Objetivo:

Permitir la carga inicial de datos de prueba, necesarios para verificar el funcionamiento de:

- Vistas
- Funciones
- Stored Procedures
- Triggers

Este archivo permite demostrar que el sistema funciona correctamente en un entorno real.

ORDEN DE EJECUCIÓN

Para garantizar el correcto funcionamiento del sistema, los scripts deben ejecutarse en el siguiente orden:

1. 01_creacion_objetos_distribuidora.sql
2. 02_insercion_datos_distribuidora.sql

De esta manera se crean primero los objetos lógicos del sistema y luego se cargan los datos para su prueba y validación.

CONCLUSIÓN DEL PROYECTO

El presente proyecto permitió desarrollar una base de datos relacional para la gestión de una distribuidora, contemplando la administración de clientes, artículos, pedidos, vendedores y proveedores.

Durante esta segunda etapa se incorporaron objetos avanzados de base de datos, tales como vistas, funciones, procedimientos almacenados y triggers, con el objetivo de mejorar la organización, reutilización y control de la información.

Se logró:

- Automatizar la actualización del stock.
- Evitar inconsistencias como stock negativo.
- Centralizar cálculos de totales.
- Simplificar consultas mediante vistas.

- Facilitar reportes por rango de fechas.

El sistema implementado cumple con los requisitos funcionales y técnicos solicitados, manteniendo una estructura clara, coherente y normalizada.