Base de Datos – Entrega 2

Alexei Sandoval

1. Introducción y Objetivo

Primer documento entregado (No sabía si tengo que entregar eso denuevo, aunque está en el drive)

2. Vistas

`vista_detalle_ventas`

Descripción: Esta vista combina información de las tablas `pedidos`, `clientes`, `detalles_pedido` y `productos` para ofrecer un resumen completo de cada ítem vendido.

Objetivo: Simplificar el proceso de generación de reportes de ventas detallados, eliminando la necesidad de realizar múltiples `JOIN`s en cada consulta. Por ejemplo, permite ver rápidamente qué productos compró cada cliente en un pedido específico y el subtotal de cada línea de producto.

Tablas que la componen: `pedidos`, `clientes`, `detalles_pedido`, `productos`.

Script de creación:

```
CREATE VIEW vista_detalle_ventas AS
SELECT
    p.id_pedido,
    c.nombre AS nombre_cliente,
    c.apellido AS apellido_cliente,
    p.fecha_pedido,
    pr.nombre_producto,
    dp.cantidad,
    dp.precio_unitario,
    (dp.cantidad * dp.precio_unitario) AS subtotal_linea
FROM pedidos p
JOIN clientes c ON p.id_cliente = c.id_cliente
JOIN detalles_pedido dp ON p.id_pedido = dp.id_pedido
JOIN productos pr ON dp.id_producto = pr.id_producto;
```

`vista_resumen_stock`

Descripción: Muestra el stock actual de todos los productos disponibles, categorizados y con su proveedor asociado.

Objetivo: Facilitar la gestión de inventario y la toma de decisiones. Permite al usuario del negocio ver de un vistazo qué productos tiene en stock, a qué categoría pertenecen y qué proveedor los suministra, lo cual es útil para planificar compras.

Tablas que la componen: `productos`, `categoriass`, `proveedores`.

Script de creación:

```
CREATE VIEW vista_resumen_stock AS
SELECT
    pr.nombre_producto,
    c.nombre_categoria AS categoria,
    p.nombre_proveedor AS proveedor,
    pr.stock
FROM productos pr
JOIN categorias c ON pr.id_categoria = c.id_categoria
JOIN proveedores p ON pr.id_proveedor = p.id_proveedor;
```

3. Funciones

`calcular_total_pedido(id_pedido_in)`

Descripción: Esta función calcula el total de un pedido sumando la cantidad por el precio unitario de todos los productos en el `detalles_pedido` asociado.

Objetivo: Asegurar que el cálculo del total de un pedido sea consistente y se realice de forma eficiente. Se puede usar en consultas y en otros objetos como triggers o procedimientos almacenados.

Datos/Tablas que utiliza: La tabla `detalles_pedido`, usando el `id_pedido`.

Script de creación:

```
DELIMITER //
CREATE FUNCTION calcular_total_pedido(id_pedido_in INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_pedido INT;
    SELECT SUM(cantidad * precio_unitario) INTO total_pedido
    FROM detalles_pedido
    WHERE id_pedido = id_pedido_in;
    RETURN total_pedido;
END //
```

```
DELIMITER ;
```

`obtener_stock_disponible(id_producto_in)`

Descripción: Función que devuelve la cantidad de stock disponible para un producto específico.

Objetivo: Proporcionar una manera sencilla y directa de consultar el stock de un producto, útil para validaciones o consultas rápidas.

Datos/Tablas que utiliza: La tabla `productos`, usando el `id_producto`.

Script de creación:

```
DELIMITER //
CREATE FUNCTION obtener_stock_disponible(id_producto_in INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE stock_actual INT;
    SELECT stock INTO stock_actual
    FROM productos
    WHERE id_producto = id_producto_in;
    RETURN stock_actual;
END //
DELIMITER ;
```

4. Procedimientos Almacenados

`sp_insertar_nuevo_pedido(id_cliente_in, id_producto_in, cantidad_in)`

Descripción: Este procedimiento maneja la lógica completa para crear un nuevo pedido desde cero. Inserta un registro en la tabla `pedidos` y luego en la tabla `detalles_pedido`, asegurando que el proceso se ejecute de manera atómica (como una sola transacción).

Objetivo: Automatizar y simplificar el proceso de venta. En lugar de ejecutar múltiples comandos SQL, la aplicación puede llamar a este procedimiento con los datos del cliente y el producto, y el sistema se encargará del resto.

Tablas que manipula: `pedidos`, `detalles_pedido`, `productos`.

`sp_obtener_ventas_cliente(id_cliente_in)`

Descripción: Procedimiento que genera un reporte de todos los pedidos realizados por un cliente específico.

Objetivo: Proporcionar una herramienta para el análisis del comportamiento de compra de los clientes. Permite ver el historial de pedidos de un cliente, incluyendo la fecha, el total y el detalle de los productos comprados.

Tablas que manipula: `pedidos`, `detalles_pedido`, `productos`.

5. Disparadores

`tr_actualizar_total_pedido`

Propósito: Asegurar la consistencia del campo `total` en la tabla `pedidos`.

Evento: `AFTER INSERT` en la tabla `detalles_pedido`.

Sobre qué tabla: `detalles_pedido`.

Script SQL:

```
DELIMITER //
CREATE TRIGGER tr_actualizar_total_pedido
AFTER INSERT ON detalles_pedido
FOR EACH ROW
BEGIN
    UPDATE pedidos
    SET total = calcular_total_pedido(NEW.id_pedido)
    WHERE id_pedido = NEW.id_pedido;
END //
DELIMITER ;
```

`tr_disminuir_stock`

Propósito: Garantizar que no se venda un producto sin stock y de ser así, actualizar el inventario automáticamente.

Evento: `BEFORE INSERT` en la tabla `detalles_pedido`.

Sobre qué tabla: `detalles_pedido`.

Script SQL:

```
DELIMITER //
CREATE TRIGGER tr_disminuir_stock
BEFORE INSERT ON detalles_pedido
```

```
FOR EACH ROW
BEGIN
   DECLARE stock_actual INT;
   SELECT stock INTO stock_actual FROM productos WHERE id_producto =
NEW.id_producto;

   IF stock_actual < NEW.cantidad THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No hay stock suficiente para
este producto.';
   ELSE
      UPDATE productos
      SET stock = stock - NEW.cantidad
      WHERE id_producto = NEW.id_producto;
   END IF;
END //
DELIMITER ;
```

GITHUB

https://github.com/AlexeiSandoval23/bdcoderhouse