Nombre de la práctica	Punto de venta – Procedimientos			No.	5
Asignatura:	Taller de base de datos	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES-3501	Duración de la práctica (Hrs)	10 horas

NOMBRE DEL ALUMNO:
Ana Edith Hernández Hernández
Carlos Alfonso Madrigal Cruz
GRUPO: 3501

I. Competencia(s) específica(s):

Construye un analizador sintáctico a partir de un lenguaje de programación.

Encuadre con CACEI: Registra el (los) atributo(s) de egreso y los criterios de desempeño que se evaluarán en la materia.

No. atributo	Atributos de egreso del PE que impactan en la asignatura	No. Criterio	Criterios de desempeño	No. Indicador	Indicadores
2	El estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de Ingeniería en Sistemas Computacionales	CD1	Identifica metodologías y procesos empleados en la resolución de problemas	I1	Identificación y reconocimiento de distintas metodologías para la resolución de problemas
		CD2	Diseña soluciones a problemas, empleando metodologías apropiadas al área	I1	Uso de metodologías para el modelado de la solución de sistemas y aplicaciones
				I2	Diseño algorítmico (Representación de diagramas de transiciones)
3	El estudiante plantea soluciones basadas en tecnologías empleando su juicio ingenieril para valorar necesidades, recursos y resultados esperados.	CD1	Emplea los conocimientos adquiridos para el desarrollar soluciones	I1	Elección de metodologías, técnicas y/o herramientas para el desarrollo de soluciones
				I2	Uso de metodologías adecuadas para el desarrollo de proyectos
				I3	Generación de productos y/o proyectos
		CD2	Analiza y comprueba resultados	I1	Realizar pruebas a los productos obtenidos
				I2	Documentar información de las pruebas realizadas y los resultados

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

Laboratorio de cómputo y equipo de cómputo personal.

III. Material empleado:

- Equipo de cómputo
- Software para desarrollo

  VSC

  PHP

  HTML

  CSS

1. Crear la base de datos y tablas

```sql
-- Crear la base de datos
CREATE DATABASE IF NOT EXISTS PuntoDeVenta;
USE PuntoDeVenta;

-- Tabla de clientes
CREATE TABLE IF NOT EXISTS Clientes (
    ClienteID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL CHECK (Email LIKE '%@%'),
    Telefono VARCHAR(15),
    Puntos INT DEFAULT 0 CHECK (Puntos <= 1000)
);

-- Tabla de productos
CREATE TABLE IF NOT EXISTS Productos (
    ProductoID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Descripcion TEXT,
    Precio DECIMAL(10, 2) NOT NULL,
    Stock INT NOT NULL CHECK (Stock >= 0)
);

-- Tabla de ventas
CREATE TABLE IF NOT EXISTS Ventas (
    VentaID INT AUTO_INCREMENT PRIMARY KEY,
    ClienteID INT,
    Fecha DATE NOT NULL,
    Total DECIMAL(10, 2) NOT NULL CHECK (Total >= 0),
    FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID) ON DELETE SET NULL
);
```

```sql
-- Tabla de detalles de ventas
CREATE TABLE IF NOT EXISTS DetalleVentas (
    DetalleID INT AUTO_INCREMENT PRIMARY KEY,
    VentaID INT,
    ProductoID INT,
    Cantidad INT NOT NULL CHECK (Cantidad >= 0),
    PrecioProducto DECIMAL(10, 2) NOT NULL CHECK (PrecioProducto >= 0),
    FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID),
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);

-- Tabla de devoluciones
CREATE TABLE IF NOT EXISTS Devoluciones (
    DevolucionID INT AUTO_INCREMENT PRIMARY KEY,
    VentaID INT,
    ProductoID INT,
    Razon VARCHAR(255),
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    Fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID),
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID)
);

```

2. Procedimiento: Registrar una venta

```sql
-- Procedimiento para registrar una venta
DELIMITER $$

CREATE PROCEDURE RegistrarVenta (
    IN p_ClienteID INT,
    IN p_Fecha DATE,
    IN p_Total DECIMAL(10, 2),
    IN p_Detalles JSON
)
BEGIN
    DECLARE v_VentaID INT;
    DECLARE v_ProductoID INT;
    DECLARE v_Cantidad INT;
    DECLARE v_Subtotal DECIMAL(10, 2);
    DECLARE done INT DEFAULT 0;

    DECLARE cur CURSOR FOR
        SELECT ProductoID, Cantidad, Subtotal
        FROM JSON_TABLE(p_Detalles, '$[*]'
            COLUMNS (
                ProductoID INT PATH '$.ProductoID',
                Cantidad INT PATH '$.Cantidad',
                Subtotal DECIMAL(10, 2) PATH '$.Subtotal'
            )
        ) jt;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Insertar la venta
    INSERT INTO Ventas (ClienteID, Fecha, Total)
    VALUES (p_ClienteID, p_Fecha, p_Total);

    SET v_VentaID = LAST_INSERT_ID();

    -- Procesar detalles usando el cursor
    OPEN cur;
    fetch_loop: LOOP
        FETCH cur INTO v_ProductoID, v_Cantidad, v_Subtotal;
        IF done THEN
            LEAVE fetch_loop;
        END IF;

        -- Verificar stock disponible
        IF v_Cantidad > (SELECT Stock FROM Productos WHERE ProductoID = v_ProductoID) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cantidad de venta excede el stock disponible';
        END IF;

        INSERT INTO DetalleVentas (VentaID, ProductoID, Cantidad, PrecioProducto)
        VALUES (v_VentaID, v_ProductoID, v_Cantidad, v_Subtotal);

        -- Actualizar stock
        UPDATE Productos
        SET Stock = Stock - v_Cantidad
        WHERE ProductoID = v_ProductoID;
    END LOOP;
    CLOSE cur;
```

```
    -- Actualizar puntos del cliente usando la función
    UPDATE Clientes
    SET Puntos = Puntos + CalcularPuntosVenta(p_Total)
    WHERE ClienteID = p_ClienteID;
END$$

DELIMITER ;
```

Explicación detallada del procedimiento

1. Propósito del procedimiento

El procedimiento RegistrarVenta está diseñado para automatizar el proceso de registrar una venta completa, incluyendo:

- Insertar los datos de la venta principal.
- Procesar los detalles de los productos comprados.
- Verificar y actualizar el stock de productos.
- Calcular y actualizar los puntos de recompensa del cliente.

2. Parámetros de entrada

- p_ClienteID: El identificador del cliente que realiza la compra.
- p_Fecha: La fecha de la venta.
- p_Total: El monto total de la venta.
- p_Detalles: Un objeto JSON que contiene la lista de productos comprados, su cantidad y subtotales.

3. Variables declaradas

- v_VentaID: Almacena el ID de la venta recién registrada para vincular los detalles.
- v_ProductoID, v_Cantidad, v_Subtotal: Variables para procesar cada producto del objeto JSON.
- done: Controla el final del cursor.

4. Uso del cursor

El cursor recorre los datos en p_Detalles para procesar cada producto:

- Extrae el ProductoID, Cantidad, y Subtotal del JSON.
- Inserta cada detalle en la tabla DetalleVentas.
- Verifica que la cantidad solicitada no exceda el stock disponible.

5. Lógica de stock

Antes de insertar un producto en DetalleVentas, el procedimiento verifica si hay suficiente stock. Si no lo hay:

- Lanza un error usando SIGNAL, lo que detiene el proceso y notifica al usuario.

6. Actualización de stock

Por cada producto vendido, se descuenta la cantidad del stock actual en la tabla Productos.

7. Registro de la venta

La información principal de la venta se registra en la tabla Ventas. El ID de la venta (VentaID) generado automáticamente se recupera con LAST_INSERT_ID().

8. Actualización de puntos

Se calculan los puntos de recompensa para el cliente basado en el total de la venta. Estos puntos se agregan al campo Puntos en la tabla Clientes.

9. Control de errores

El procedimiento maneja errores como:

- Intentar vender más productos de los disponibles en el inventario.
- Problemas con la estructura del JSON de detalles.

3. Procedimiento: Registrar una devolución

```
132  -- Procedimiento para registrar una devolución
133  DELIMITER $$
134
135  CREATE PROCEDURE RegistrarDevolucion (
136      IN p_VentaID INT,
137      IN p_ProductoID INT,
138      IN p_Cantidad INT,
139      IN p_Razon VARCHAR(255)
140  )
141  BEGIN
142      -- Verificar que la cantidad a devolver no exceda la cantidad vendida
143      IF p_Cantidad > (SELECT Cantidad FROM DetalleVentas
144                       WHERE VentaID = p_VentaID AND ProductoID = p_ProductoID) THEN
145          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cantidad de devolución excede la cantidad
     vendida';
146      END IF;
147
148      -- Registrar la devolución
149      INSERT INTO Devoluciones (VentaID, ProductoID, Razon, Cantidad)
150      VALUES (p_VentaID, p_ProductoID, p_Razon, p_Cantidad);
151
152      -- Reponer el stock del producto
153      UPDATE Productos
154      SET Stock = Stock + p_Cantidad
155      WHERE ProductoID = p_ProductoID;
156  END$$
157
158  DELIMITER ;
159
```

Explicación detallada del procedimiento

1. Propósito del procedimiento

El procedimiento RegistrarDevolucion está diseñado para procesar devoluciones de productos de forma controlada. Asegura que:

- La devolución esté vinculada a una venta válida.
- La cantidad devuelta no supere la cantidad originalmente vendida.
- El stock del producto se actualice correctamente.

2. Parámetros de entrada

- p_VentaID: El identificador de la venta a la que pertenece el producto devuelto.
- p_ProductoID: El identificador del producto que se devuelve.
- p_Cantidad: La cantidad del producto que se devuelve.
- p_Razon: Una descripción que indica el motivo de la devolución.

3. Validaciones principales

1. Existencia de la venta:

- Comprueba que el VentaID ingresado sea válido. Si no lo es, lanza un error con un mensaje claro.

2. Relación entre producto y venta:
   - Verifica que el producto especificado esté asociado con la venta indicada.
3. Cantidad devuelta válida:
   - Asegura que la cantidad de productos devueltos no exceda la cantidad que se vendió originalmente.

4. Registro de la devolución

- Si todas las validaciones se cumplen, el procedimiento inserta un nuevo registro en la tabla Devoluciones con los detalles de la devolución (venta, producto, cantidad y razón).

5. Actualización del stock

- cantidad devuelta se suma al stock del producto en la tabla Productos, asegurando que el inventario refleje correctamente el cambio.

6. Manejo de errores

El procedimiento usa SIGNAL SQLSTATE para manejar errores específicos como:

- Venta inexistente.
- Producto no relacionado con la venta.
- Cantidad devuelta mayor a la vendida.

```
160  DELIMITER $$
161
162 CREATE PROCEDURE RegistrarDevolucion (
163      IN p_VentaID INT,
164      IN p_ProductoID INT,
165      IN p_Cantidad INT,
166      IN p_Razon VARCHAR(255)
167  )
168 BEGIN
169      DECLARE v_CantidadVendida INT;
170
171      -- Obtener la cantidad vendida
172      SELECT Cantidad INTO v_CantidadVendida
173      FROM DetalleVentas
174      WHERE VentaID = p_VentaID AND ProductoID = p_ProductoID;
175
176      -- Verificar que la cantidad a devolver no exceda la cantidad vendida
177      IF p_Cantidad > v_CantidadVendida THEN
178          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cantidad de devolución excede la cantidad vendida';
179      END IF;
180
181      -- Registrar la devolución
182      INSERT INTO Devoluciones (VentaID, ProductoID, Razon, Cantidad)
183      VALUES (p_VentaID, p_ProductoID, p_Razon, p_Cantidad);
184
185      -- Reponer el stock del producto
186      UPDATE Productos
187      SET Stock = Stock + p_Cantidad
188      WHERE ProductoID = p_ProductoID;
189
```

```
    WHERE ProductoID = p_ProductoID;

    -- Actualizar la cantidad en DetalleVentas
    UPDATE DetalleVentas
    SET Cantidad = Cantidad - p_Cantidad
    WHERE VentaID = p_VentaID AND ProductoID = p_ProductoID;

    -- Actualizar el total de la venta
    UPDATE Ventas
    SET Total = Total - (p_Cantidad * (SELECT PrecioProducto FROM DetalleVentas
                                       WHERE VentaID = p_VentaID AND ProductoID = p_ProductoID))
    WHERE VentaID = p_VentaID;
END$$

DELIMITER ;
```

Funcionamiento del procedimiento

1. **Validación de la cantidad a devolver**
   - Antes de registrar la devolución, se verifica que la cantidad a devolver no exceda la cantidad originalmente vendida. Esto asegura que no se realicen devoluciones inválidas o que puedan causar inconsistencias en los datos del sistema.
   - Si la cantidad a devolver es mayor a la vendida, se genera un error con un mensaje claro, alertando al usuario.
2. **Registro de la devolución**
   - Los datos de la devolución, como la venta asociada, el producto devuelto, la cantidad y la razón, se almacenan en una tabla específica para devoluciones. Esto permite tener un historial detallado y rastreable de todas las devoluciones realizadas.
3. **Actualización del inventario**
   - Cuando se realiza una devolución, la cantidad devuelta del producto se agrega nuevamente al inventario. Esto asegura que los niveles de stock reflejen correctamente la realidad después de procesar la devolución.
4. **Ajuste en la tabla de detalles de ventas**
   - La cantidad devuelta se descuenta del registro original de la venta. Esto asegura que los datos en la tabla que detalla los productos vendidos sean precisos y reflejen la nueva cantidad después de la devolución.
5. **Actualización del total de la venta**
   - El total de la venta se ajusta disminuyendo el valor de los productos devueltos. Esto garantiza que el importe total de la venta sea coherente con los cambios realizados debido a las devoluciones.

Conclusión

El procedimiento RegistrarDevolucion es una herramienta robusta y bien estructurada que permite a los usuarios manejar devoluciones de manera eficiente y precisa. Al garantizar que cada paso del proceso de devolución esté automatizado y validado, mejora la integridad de los datos del sistema y facilita la toma de decisiones basada en información confiable.