Nombre de la práctica	Punto de venta – Triggers			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. Trigger 1: Actualizar puntos al eliminar una venta:

```
204  -- Trigger para actualizar puntos del cliente al eliminar una venta
205  DELIMITER $$
206
207  CREATE TRIGGER ActualizarPuntosAlEliminarVenta BEFORE DELETE ON Ventas
208  FOR EACH ROW
209  BEGIN
210      UPDATE Clientes
211      SET Puntos = Puntos - CalcularPuntosVenta(OLD.Total)
212      WHERE ClienteID = OLD.ClienteID;
213  END$$
214
215  DELIMITER ;
216
```

Explicacion del TRIGGER:

Evento activador:

- El trigger se ejecuta antes (BEFORE) de que se elimine un registro en la tabla Ventas (es decir, ante un comando DELETE sobre esta tabla).
-

Afecta a cada registro eliminado:

- La cláusula FOR EACH ROW indica que el trigger actúa para cada fila eliminada individualmente. Es decir, si se eliminan varias ventas a la vez, el trigger se ejecuta para cada una.

Lógica del trigger:

- Usa los valores de la fila eliminada, accesibles mediante la palabra clave OLD. Esto incluye los valores de las columnas de la fila antes de que se elimine.
- El trigger:
    - Resta del campo Puntos en la tabla Clientes el resultado de la función CalcularPuntosVenta, que toma como argumento el total de la venta eliminada (OLD.Total).
    - Actualiza la fila en Clientes correspondiente al cliente afectado, identificado por OLD.ClienteID.

Estructura del bloque de código:

- La instrucción UPDATE Clientes ajusta el campo Puntos para reflejar el impacto de la eliminación de la venta.

3. Trigger 2: Validar stock al actualizar un detalle de venta

```
216
217  -- Trigger para validar stock al actualizar detalles de venta
218  DELIMITER $$
219
220  CREATE TRIGGER ValidarStockAlActualizar AFTER UPDATE ON DetalleVentas
221  FOR EACH ROW
222  BEGIN
223      IF NEW.Cantidad > (SELECT Stock FROM Productos WHERE ProductoID = NEW.ProductoID) THEN
224          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stock insuficiente para la actualización';
225      END IF;
226  END$$
227
228  DELIMITER ;
229
```

Análisis del trigger

1. **Evento activador:**
   - El trigger se ejecuta después (AFTER) de actualizar un registro en la tabla DetalleVentas (es decir, tras ejecutar un comando UPDATE).
2. **Afecta a cada registro actualizado:**
   - La cláusula FOR EACH ROW indica que el trigger actúa sobre cada fila modificada individualmente.
3. **Lógica del trigger:**
   - Se compara la cantidad actualizada (NEW.Cantidad) con el stock actual del producto relacionado, que se obtiene de la tabla Productos usando el identificador del producto (NEW.ProductoID).
   - Si NEW.Cantidad es mayor que el valor del stock:
     - Se usa el comando SIGNAL para generar un error SQL.
     - El error tiene el código 45000, que es un estado definido por el usuario en SQL.
     - Además, se envía un mensaje: "Stock insuficiente para la actualización".
4. **Estructura del bloque de código:**
   - La instrucción IF evalúa la condición.
   - Si la condición es verdadera, la instrucción SIGNAL interrumpe la transacción con un error.

Conclusion:

Los triggers presentados aseguran la consistencia e integridad de los datos automatizando reglas de negocio críticas: el primero ajusta automáticamente los puntos de clientes al eliminar una venta, sincronizando datos relacionados, mientras que el segundo valida que las actualizaciones en las cantidades de ventas no excedan el stock disponible, previniendo inconsistencias en el inventario. Ambos demuestran el valor de centralizar validaciones en la base de datos, reduciendo errores manuales y garantizando que el sistema refleje estados precisos y coherentes en todo momento.