1. Definir dónde pretenden utilizar el trigger.

odrían ser utilizados en situaciones específicas donde se requiera ejecutar automáticamente ciertas acciones en respuesta a eventos específicos en la base de datos.

```
CREATE TRIGGER actualiza_saldo

AFTER INSERT ON Transaccion

FOR EACH ROW

BEGIN

  IF NEW.TipoOperacion = 'Transferencia' THEN

    UPDATE Cuenta SET Saldo = Saldo - NEW.Monto WHERE NumeroCuenta = NEW.CuentaOrigen;

    UPDATE Cuenta SET Saldo = Saldo + NEW.Monto WHERE NumeroCuenta = NEW.CuentaDestino;

  ELSE

    -- Otras acciones para otros tipos de operación si es necesario.

  END IF;

END;


CREATE TRIGGER generaContraseña

BEFORE INSERT ON RetiroSinCuenta

FOR EACH ROW

BEGIN

  SET NEW.Contraseña = FLOOR(RAND() * 100000000); -- Genera un número aleatorio de 8 dígitos.

END;
```

2. Definir dónde pretenden utilizar la transacción.

Las transacciones se utilizan en el sistema bancario para realizar operaciones financieras

Transferencia:

Un cliente puede utilizar la transacción de transferencia para mover fondos entre sus propias cuentas o hacia cuentas de otros clientes del banco.

La transferencia implica retirar dinero de una cuenta de origen y depositarlo en una cuenta de destino, ya sea perteneciente al mismo cliente o a otro cliente del banco.

Este tipo de transacción se realizará a través del sistema en línea del banco, permitiendo a los clientes gestionar sus fondos de manera electrónica.

Retiro sin Cuenta:

Una persona que no sea cliente del banco puede realizar un retiro sin cuenta a nombre de un cliente existente.

Este tipo de transacción se lleva a cabo utilizando un folio de operación y una contraseña generada por el sistema.

La persona debe proporcionar el folio de operación y la contraseña al banco para retirar el monto solicitado, sin necesidad de tener una cuenta en el banco.

Este servicio se puede utilizar en sucursales bancarias, permitiendo a personas no clientes retirar efectivo de manera segura y controlada.

Consulta de Historial de Operaciones:

Los clientes pueden utilizar esta funcionalidad para revisar su historial de operaciones, lo que incluye todas las transferencias y retiros realizados.

La consulta se realiza a través del sistema en línea del banco, donde el cliente puede seleccionar un rango de fechas y filtrar por tipo de operación.

Proporciona a los clientes un medio eficiente para monitorear y revisar sus actividades financieras.

```
-- Crear una transacción para la transferencia de fondos
BEGIN TRY
  BEGIN TRANSACTION;

  -- Verificar saldo suficiente en la cuenta de origen
  IF (SELECT Saldo FROM Cuenta WHERE ID_Cuenta = @ID_Cuenta_Origen) >= @Monto
  BEGIN
    -- Realizar la transferencia
```

```
        UPDATE Cuenta SET Saldo = Saldo - @Monto WHERE ID_Cuenta = @ID_Cuenta_Origen;
        UPDATE Cuenta SET Saldo = Saldo + @Monto WHERE ID_Cuenta = @ID_Cuenta_Destino;

        -- Registrar la transacción
        INSERT INTO Transaccion (Tipo, Monto, Fecha, ID_Cuenta)
        VALUES ('Transferencia', @Monto, GETDATE(), @ID_Cuenta_Origen);

        -- Confirmar la transacción
        COMMIT;
    END
    ELSE
    BEGIN
        -- Manejar el caso de saldo insuficiente
        RAISEERROR('Saldo insuficiente en la cuenta de origen.', 16, 1);
    END
END TRY
BEGIN CATCH
    -- Deshacer la transacción en caso de error
    ROLLBACK;
    -- Manejar el error (log, notificación, etc.)
    PRINT ERROR_MESSAGE();
END CATCH;
```

3. Definir dónde pretenden utilizar el stored procedure

Los Stored procedures (procedimientos almacenados) podrían ser utilizados para mejorar la eficiencia y seguridad en la ejecución de ciertas operaciones y consultas en la base de datos

Se podria usar en

- Registro y actualización de cliente
- Gestión de cuentas
- Transacciones
- Historial de operaciones
- Retiro sin cuenta

```
CREATE PROCEDURE sp_GestionarCliente
```

```
    @ID_Cliente INT,
    @Nombre_Completo NVARCHAR(100),
    @Domicilio NVARCHAR(255),
    @Fecha_Nacimiento DATE,
    @Edad INT
AS
BEGIN
    IF @ID_Cliente IS NULL
    BEGIN
        -- Inserción de nuevo cliente
        INSERT INTO Cliente (Nombre_Completo, Domicilio, Fecha_Nacimiento, Edad)
        VALUES (@Nombre_Completo, @Domicilio, @Fecha_Nacimiento, @Edad);
    END
    ELSE
    BEGIN
        -- Actualización de datos de cliente existente
        UPDATE Cliente
        SET Nombre_Completo = @Nombre_Completo,
            Domicilio = @Domicilio,
            Fecha_Nacimiento = @Fecha_Nacimiento,
            Edad = @Edad
        WHERE ID_Cliente = @ID_Cliente;
    END
END;

CREATE PROCEDURE sp_TransferenciaFondos
    @ID_Cuenta_Origen INT,
    @ID_Cuenta_Destino INT,
    @Monto DECIMAL(18, 2)
AS
BEGIN
```

```
    -- Verificar saldo suficiente en la cuenta de origen
    IF (SELECT Saldo FROM Cuenta WHERE ID_Cuenta = @ID_Cuenta_Origen) >= @Monto
    BEGIN
        -- Realizar la transferencia
        UPDATE Cuenta SET Saldo = Saldo - @Monto WHERE ID_Cuenta = @ID_Cuenta_Origen;
        UPDATE Cuenta SET Saldo = Saldo + @Monto WHERE ID_Cuenta = @ID_Cuenta_Destino;

        -- Registrar la transacción
        INSERT INTO Transaccion (Tipo, Monto, Fecha, ID_Cuenta)
        VALUES ('Transferencia', @Monto, GETDATE(), @ID_Cuenta_Origen);
    END
    ELSE
    BEGIN
        -- Manejar el caso de saldo insuficiente
        RAISEERROR('Saldo insuficiente en la cuenta de origen.', 16, 1);
    END
END;
```