GUIA 03: TRANSACCIONES

1. MARCO TEORICO

1.1 Concepto de transacciones:

Una transacción es una unidad lógica de trabajo que agrupa una o más operaciones en una base de datos, garantizando que se ejecuten de manera completa y coherente o, en caso de fallo, que ninguna de las operaciones tenga efecto. Esto asegura la integridad de los datos y permite que la base de datos mantenga un estado válido antes y después de la transacción, siguiendo las propiedades ACID: atomicidad, consistencia, aislamiento y durabilidad.

Las transacciones son como redes de seguridad para las bases de datos en SQL Server. Ayudan a mantener los datos seguros y consistentes al garantizar que, cuando realizamos cambios, todos se realicen a la vez o ninguno. Si algo sale mal, las transacciones nos permiten deshacer todo, evitando actualizaciones a medias. Son esenciales para operaciones de base de datos confiables y evitan problemas extraños con los datos. Las transacciones también permiten que muchas personas trabajen en la base de datos al mismo tiempo sin estropear las cosas.

Propiedades ACID:

- Atomicidad: Todas las operaciones se completan o ninguna lo hace.
- Consistencia: La base de datos permanece en un estado válido antes y después de la transacción.
- Aislamiento: Las transacciones no afectan a otras en ejecución.
- Durabilidad: Los cambios realizados son permanentes, incluso si ocurre un fallo.

1.2 Tipos de transacciones:

Hay tres tipos principales de transacciones en SQL Server:

Modo de transacción implícita

Para utilizar el modo de transacción implícita en SQL Server, primero debemos configurar el modo de transacción implícita en ON mediante la instrucción SET IMPLICIT_TRANSACTIONS . El valor de IMPLICIT_TRANSACTIONS puede ser ON u OFF . Cuando el valor del modo de transacción implícita se configura en ON, SQL Server inicia automáticamente una nueva transacción cada vez que ejecutamos cualquier instrucción SQL (Insert, Select, Delete y Update).

Modo de transacción explícita

Definidas manualmente por el usuario con comandos como BEGIN TRANSACTION, COMMIT y ROLLBACK.

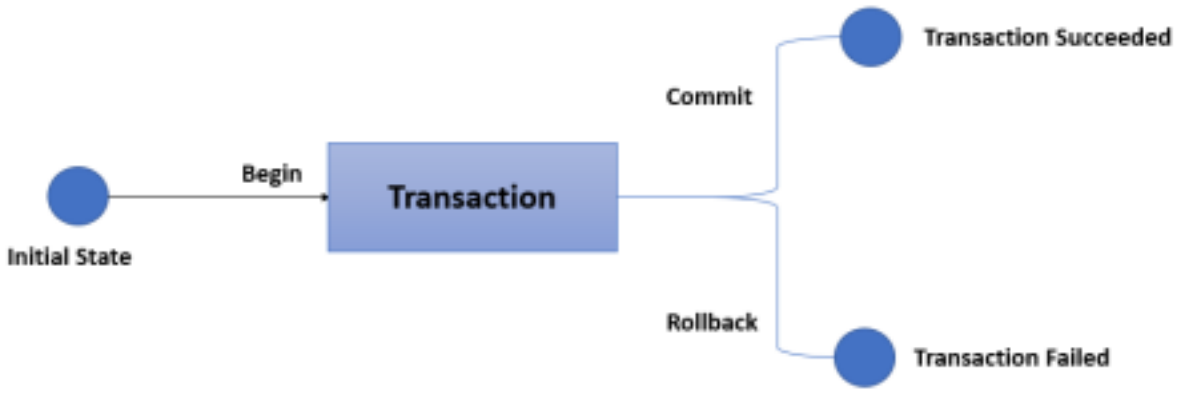
En el modo explícito de transacción, el desarrollador solo es responsable de iniciar la transacción, así como de finalizarla. En otras palabras, podemos decir que las transacciones

que tienen un INICIO y un FIN escritos explícitamente por el programador se denominan transacciones explícitas.

Aquí, cada transacción debe comenzar con la instrucción BEGIN TRANSACTION y finalizar con una instrucción ROLLBACK TRANSACTION (cuando la transacción no se completa correctamente) o una instrucción COMMIT TRANSACTION (cuando la transacción se completa correctamente). El modo de transacción explícita se utiliza con mayor frecuencia en activadores, procedimientos almacenados y programas de aplicación.

Modo de transacción de confirmación automática

Este es el modo de transacción predeterminado en SQL Server. En este modo de transacción, cada instrucción SQL se trata como una transacción independiente. En este modo de transacción, como desarrolladores, no somos responsables ni de iniciar la transacción (es decir, iniciar transacción) ni de finalizarla (es decir, confirmarla o revertirla). Siempre que ejecutamos una instrucción DML, SQL Server comenzará automáticamente la transacción y la finalizará con una confirmación o reversión, es decir, si la transacción se completa correctamente, se confirma. Si la transacción presenta algún error, se revierte. Por lo tanto, el programador no tiene ningún control sobre ellas.



1.3 Importancia en el manejo de errores:

- Cómo el uso de transacciones ayuda a evitar inconsistencias en la base de datos en caso de errores o fallos del sistema.

2. DESARROLLO DE LA GUÍA

a) Crear la Base de Datos y las Tablas.
Creación de una base de datos para manejo de cuentas bancarias con las siguientes tablas:

- TCliente: ID_Cliente, Nombre, Email, Telefono.
- TCuenta: ID_Cuenta, ID_Cliente, Saldo, TipoCuenta.
- TTransaccion: ID_Cuenta, TipoTransaccion, Monto.

```
USE Master
GO
--Crear la base de datos BDCuentasBancarias
IF EXISTS (SELECT Name FROM SysDatabases WHERE Name IN ('BDCuentasBancarias'))
    DROP DATABASE BDCuentasBancarias
GO
CREATE DATABASE BDCuentasBancarias
GO
```

```
USE BDCuentasBancarias
GO
-- Crear tabla Cliente
CREATE TABLE TCliente
(
    ID_Cliente varchar(10) PRIMARY KEY,
    Nombre varchar(60),
    Email varchar(60),
    Telefono varchar(15),
)
GO
-- Crear tabla Cuentas
CREATE TABLE TCuenta
(
    ID_Cuenta int PRIMARY KEY,
    ID_Cliente varchar(10) NOT NULL,
    Saldo decimal(10, 2) NOT NULL CHECK (Saldo >= 0),
    TipoCuenta varchar(50) CHECK (TipoCuenta IN ('Ahorros', 'Corriente')),
    FechaCreacion date DEFAULT GETDATE(),
    FOREIGN KEY (ID_Cliente) REFERENCES TCliente(ID_Cliente)
)
GO
-- Crear tabla Transaccion
CREATE TABLE TTransaccion
(
    ID_Transaccion int identity(1,1) PRIMARY KEY ,
    ID_Cuenta int NOT NULL,
    TipoTransaccion varchar(50) CHECK(TipoTransaccion IN ('Depósito', 'Retiro', 'Transferencia')),
    Monto decimal(10, 2) NOT NULL,
    ID_CuentaDestino int NULL,
    Fecha date DEFAULT GETDATE(),
    FOREIGN KEY (ID_Cuenta) REFERENCES TCuenta(ID_Cuenta),
    FOREIGN KEY (ID_CuentaDestino) REFERENCES TCuenta(ID_Cuenta)
)
GO
```

b)

Insertar Datos de Prueba

```
USE BDCuentasBancarias
GO
-- Insertar clientes
INSERT INTO TCliente (ID_Cliente, Nombre, Email, Telefono)
VALUES ('C001', 'Juan Pérez', 'juan.perez@gmail.com', '931456789'),
       ('C002', 'Ana Gómez', 'ana.gomez@gmail.com', '987654321'),
       ('C003', 'Maria Loza', 'marita@gmail.com', '984653322'),
       ('C004', 'Carlos Alvarez', 'carlitos.alva@hotmmail.com', '985655524'),
       ('C005', 'Milagros Leiva', 'mili.leiva@gmail.com', '944443321')

-- Insertar cuentas
INSERT INTO TCuenta (ID_Cuenta, ID_Cliente, Saldo, TipoCuenta)
VALUES (101,'C001', 1000.00, 'Ahorros'),
       (102,'C004', 500.00, 'Corriente')

-- Insertar transacciones
INSERT INTO TTransaccion (ID_Cuenta, TipoTransaccion, Monto)
VALUES (101, 'Depósito', 200.00),
       (102, 'Retiro', 100.00)
```

c) Implementar el Sistema con Transacciones

Implementación un sistema de gestión de transacciones bancarias donde:

• Se registre una transacción cada vez que se realice una operación de transferencia de dinero entre cuentas.

• Si la cuenta no tiene fondos suficientes, la transacción debe revertirse con ROLLBACK. • Usar los comandos BEGIN TRANSACTION, COMMIT y ROLLBACK para manejar las operaciones.

```
USE BDCuentasBancarias
GO

-- Variables para la transferencia
DECLARE @Cuenta_Origen int = 101; -- ID de la cuenta de origen
DECLARE @Cuenta_Destino int = 102; -- ID de la cuenta de destino
DECLARE @Monto decimal(10, 2) = 300.00; -- Monto a transferir
```

```sql
-- Iniciar transacción
BEGIN TRANSACTION;

-- Verificar si la cuenta de origen tiene saldo suficiente
IF EXISTS (SELECT 1 FROM TCuenta WHERE ID_Cuenta = @Cuenta_Origen AND Saldo >= @Monto)
BEGIN
    -- Restar el monto de la cuenta de origen
    UPDATE TCuenta
    SET Saldo = Saldo - @Monto
    WHERE ID_Cuenta = @Cuenta_Origen;

    -- Agregar el monto a la cuenta de destino
    UPDATE TCuenta
    SET Saldo = Saldo + @Monto
    WHERE ID_Cuenta = @Cuenta_Destino;

    -- Registrar la transacción en la tabla Transacciones
    INSERT INTO TTransaccion (ID_Cuenta, TipoTransaccion, Monto,ID_CuentaDestino)
    VALUES (@Cuenta_Origen, 'Transferencia', @Monto, @Cuenta_Destino);

    -- Confirmar la transacción
    COMMIT;
    PRINT 'Transferencia realizada con éxito.';
END
ELSE
BEGIN
    -- Revertir la transacción
    ROLLBACK;
    PRINT 'Saldo insuficiente. Transferencia cancelada.';
END;
```

d) Prueba del Sistema

Para probar:

· Intenta realizar una transferencia con una cuenta que tiene fondos suficientes. · Luego, intenta realizar una transferencia que exceda sus fondos y verifica que no se registre la transferencia.

3. EJERCICIOS PROPUESTOS

Usar los comandos BEGIN TRANSACTION, COMMIT y ROLLBACK para manejar las operaciones de depósito y retiro.