Tecnológico de Monterrey

Bases de Datos

Laboratorio 24

Manipulación de Datos Usando Transacciones en SQL Server

Reporte

Miguel Ángel Marines Olvera | A01705317

1. Comprobando la propiedad de aislamiento en una BD.

Inserta la siguiente transacción y ejecútala:

Sentencia

```
BEGIN TRANSACTION PRUEBA1
INSERT INTO CLIENTES_BANCA VALUES('001', 'Manuel Rios Maldonado', 9000);
INSERT INTO CLIENTES_BANCA VALUES('002', 'Pablo Perez Ortiz', 5000);
INSERT INTO CLIENTES_BANCA VALUES('003', 'Luis Flores Alvarado', 8000);
COMMIT TRANSACTION PRUEBA1
```

Posteriormente revisa con la siguiente instrucción si la tabla de clientes_banca sufrió algún cambio. Es decir, si dio de alta el registro que se describe en la transacción.

Sentencia

```
SELECT * FROM Clientes_Banca;
```

Salida

3 Rows

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	9000.00
2	002	Pablo Perez Ortiz	5000.00
3	003	Luis Flores Alvarado	8000.00

Se dió de alta el registro que se describe en la transacción.

2. Abre una nueva conexión a tu base de datos y revisa el contenido de la tabla Clientes_Banca desde la ventana que inicializaste como la segunda sesión:

¿Que pasa cuando deseas realizar esta consulta?

Sentencia

```
SELECT * FROM Clientes_Banca;
```

Salida

3 Rows

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	9000.00
2	002	Pablo Perez Ortiz	5000.00
3	003	Luis Flores Alvarado	8000.00

Se obtienen los mismos datos que en la primera conexión.

3. ¿Qué pasa cuando deseas realizar esta consulta?

Inserta la siguiente transacción y ejecútala:

Sentencia

```
BEGIN TRANSACTION PRUEBA2
INSERT INTO CLIENTES_BANCA VALUES('004','Ricardo Rios Maldonado',19000);
INSERT INTO CLIENTES_BANCA VALUES('005','Pablo Ortiz Arana',15000);
INSERT INTO CLIENTES_BANCA VALUES('006','Luis Manuel Alvarado',18000);
```

Revisa el contenido de la tabla clientes_banca desde la ventana que inicializaste como la primera sesión con la siguiente consulta:

Sentencia

```
SELECT * FROM Clientes_Banca;
```

No se ven las nuevas inserciones de los clientes 004, 005 y 006, porque no se ha hecho un commit para garantizar la unicidad.

Revisa el contenido de la tabla clientes_banca desde la ventana que inicializaste como la segunda sesión.

Sentencia

```
SELECT * FROM Clientes_Banca;
```

¿Qué pasa cuando deseas realizar esta consulta?

Se despliegan y se ven lo 6 clientes.
Se ven las inserciones de las 2 transacciones.

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	9000.00
2	002	Pablo Perez Ortiz	5000.00
3	003	Luis Flores Alvarado	8000.00
4	004	Ricardo Rios Maldonado	19000.00
5	005	Pablo Ortiz Arana	15000.00
6	006	Luis Manuel Alvarado	18000.00

4. Intenta con la siguiente consulta desde la segunda sesión:

Sentencia

```
SELECT * FROM CLIENTES_BANCA where NoCuenta='001'
```

Salida

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	9000.00

Explica por qué ocurre dicho evento.

Porque los datos que se consultan son parte de la primera transacción a la cual si se le hizo un commit, por eso se puede consultar y desplegar en las dos sesiones.

5. Por último regresa a la ventana que mantiene activa tu primer sesión, agrega el siguiente comando a la pantalla y ejecútalo:

Sentencia

```
ROLLBACK TRANSACTION PRUEBA2
```

Revisa nuevamente el contenido de la tabla clientes_banca desde la ventana que inicializaste como la segunda sesión:

Sentencia

```
SELECT * FROM Clientes_Banca;
```

Salida

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	9000.00
2	002	Pablo Perez Ortiz	5000.00
3	003	Luis Flores Alvarado	8000.00

¿Qué ocurrió y por qué?

Vuelve a desplegar solo los 3 primeros clientes porque ROLLBACK revierte las transacciones.

6. Comprobando la propiedad de Atomicidad en una BD

Inserta la siguiente transacción y ejecútala:

Sentencia:

```
BEGIN TRANSACTION PRUEBA3
INSERT INTO TIPOS_MOVIMIENTO VALUES('A','Retiro Cajero Automatico');
INSERT INTO TIPOS_MOVIMIENTO VALUES('B','Deposito Ventanilla');
COMMIT TRANSACTION PRUEBA3


BEGIN TRANSACTION PRUEBA4
INSERT INTO MOVIMIENTOS VALUES('001','A',GETDATE(),500);
UPDATE CLIENTES_BANCA SET SALDO = SALDO -500
WHERE NoCuenta='001'
COMMIT TRANSACTION PRUEBA4
```

Posteriormente revisa si las tablas de clientes_banca y movimientos sufrieron algún cambio, es decir, si dio de alta el registro que se describe en la transacción y su actualización.

Sentencia:

```
SELECT * FROM Clientes_Banca;

SELECT * FROM Movimientos;
```

Salida:

	NoCuenta	Nombre	Saldo
1	001	Manuel Rios Maldonado	8500.00
2	002	Pablo Perez Ortiz	5000.00
3	003	Luis Flores Alvarado	8000.00

	NoCuenta	ClaveM	Fecha	Monto
1	001	A	2020-11-19 11:47:22.880	500.00

7. Manejando Fallas en una Transacción

Inserta la siguiente transacción y ejecútala:

Sentencia:

```
BEGIN TRANSACTION PRUEBA5
INSERT INTO CLIENTES_BANCA VALUES('005','Rosa Ruiz Maldonado',9000);
INSERT INTO CLIENTES_BANCA VALUES('006','Luis Camino Ortiz',5000);
INSERT INTO CLIENTES_BANCA VALUES('001','Oscar Perez Alvarado',8000);


IF @@ERROR = 0
COMMIT TRANSACTION PRUEBA5
ELSE
BEGIN
PRINT 'A transaction needs to be rolled back'
ROLLBACK TRANSACTION PRUEBA5
END
```

Revisa las tablas de clientes_banca y movimientos.

Sentencia:

```
SELECT * FROM Clientes_Banca;

SELECT * FROM Movimientos;
```

¿Para qué sirve el comando @@ERROR revisa la ayuda en línea?

Para saber si la transacción mandó algún error, si es 0, no hubo error. Pero si si hubo algún error, devuelve él ID del error.

¿Qué hace la transacción?

Inserta clientes en la tabla ClientesBanca y si hay errores, hace rollback.

¿Hubo alguna modificación en la tabla? Explica qué pasó y por qué.

No, porque el tercero de los INSERT INTO tiene una llave primaria duplicada y el @@ERROR va a ser diferente de 0, y por ello se hace rollback.

8. Por analogía crea las siguientes transacciones dentro de un stored procedure:

Una transacción que registre el retiro de una cajero. nombre del store procedure REGISTRAR_RETIRO_CAJERO que recibe 2 parámetros en NoCuenta y el monto a retirar.

```
CREATE PROCEDURE REGISTRAR_RETIRO_CAJERO
    @NoCuenta VARCHAR(5),
    @Monto NUMERIC(10,2)
AS
    BEGIN TRANSACTION T1
        INSERT INTO MOVIMIENTOS VALUES(@NoCuenta,'A',GETDATE(),@Monto);
        UPDATE CLIENTES_BANCA SET SALDO = SALDO - @Monto
        WHERE NoCuenta = @NoCuenta
    IF @@ERROR = 0
        COMMIT TRANSACTION T1
    ELSE
        BEGIN
            PRINT 'El retiro se tuvo que cancelar.'
        ROLLBACK TRANSACTION T1
    END
GO
```

Una transacción que registre el deposito en ventanilla. Nombre del store procedure REGISTRAR_DEPOSITO_VENTANILLA que recibe 2 parámetros en NoCuenta y el monto a depositar.

```
CREATE PROCEDURE REGISTRAR_DEPOSITO_VENTANILLA
    @NoCuenta VARCHAR(5),
    @Monto NUMERIC(10,2)
AS
    BEGIN TRANSACTION T2
        UPDATE CLIENTES_BANCA SET SALDO = SALDO + @Monto
        WHERE NoCuenta = @NoCuenta;
        INSERT INTO MOVIMIENTOS
VALUES(@NoCuenta,'B',GETDATE(),@Monto)
    IF @@ERROR = 0
        COMMIT TRANSACTION T2
    ELSE
        BEGIN
            PRINT 'El deposito se tuvo que cancelar.'
        ROLLBACK TRANSACTION T2
    END
GO
```