ADMINISTRACIÓN DE BASE DE DATOS

Unidad II

TRANSACCIONES

Ana E. Congacha

Definición

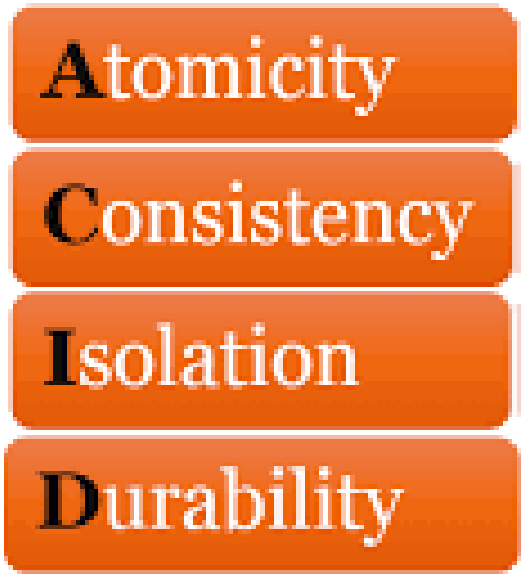
Una transacción es una unidad única de trabajo. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una ***parte permanente de la base de datos.*** *Si una transacción encuentra errores y debe cancelarse o revertirse, se* ***borran todas las modificaciones de los datos.***

EJEMPLO

A menudo, desde el punto de vista del usuario de una base de datos, se considera a un conjunto de varias operaciones sobre una base de datos como una única operación. Por ejemplo, una transferencia de fondos desde una cuenta corriente a una cuenta de ahorros es una operación simple desde el punto de vista del cliente; sin embargo, en el sistema de base de datos, está compuesta internamente por varias operaciones. Evidentemente es esencial que tengan lugar todas las operaciones o que, en caso de fallo, ninguna de ellas se produzca. Sería inaceptable efectuar el cargo de la transferencia en la cuenta corriente y que no se abonase en la cuenta de ahorros.

Propiedades de las transacciones

- **Atomicidad**. Todas las operaciones de la transacción o ninguna de ellas.
- **Consistencia**. La ejecución de la transacción conserva la consistencia de la base de datos. Cuando finaliza, una transacción debe dejar todos los datos en un estado coherente.
- **Aislamiento**. Aunque se ejecuten varias transacciones concurrentemente, el sistema garantiza que la realización de dos transacciones sobre la misma información nunca generará ningún tipo de error.
- **Durabilidad**. Tras la finalización con éxito de una transacción, los cambios realizados en la base de datos permanecen.



Transacciones de SQL Server

- En una transacción **implícita**, cada instrucción Transact-SQL, como INSERT, UPDATE o DELETE, se ejecuta como una transacción.
- En una transacción **explícita** o definida por el usuario, las instrucciones de la transacción se agrupan entre las cláusulas BEGIN TRANSACTION y COMMIT TRANSACTION.

```
BEGIN TRAN fund_transfer
   EXEC debit_checking 100, 'account1'
   EXEC credit_savings 100, 'account1'
COMMIT TRAN fund_transfer
```

Punto clave
Una transacción confirmada no se puede deshacer.

Consideraciones para el uso de transacciones

- **Recomendaciones para las transacciones**
  - Las transacciones deben ser lo más cortas posible
  - INSERT, UPDATE y DELETE deben ser las instrucciones principales de una transacción.
  - La transacción debe completarse explícitamente con COMMIT o ROLLBACK TRANSACTION

INSTRUCCIONES EQUIVALENTES

En SQL Server las instrucciones equivalentes a las genéricas son:

- **BEGIN TRANSACTION** o **BEGIN TRAN**: marca el inicio de una transacción. TRAN es un sinónimo de TRANSACTION y se suele usar más a menudo por abreviar.
- **ROLLBACK TRANSATION** o **ROLLBACK TRAN**: fuerza que se deshaga la transacción en caso de haber un problema o querer abandonarla. Cierra la transacción.
- **COMMIT TRANSACTION** O **COMMIT TRAN**: confirma el conjunto de operaciones convirtiendo los datos en definitivos. Marca el éxito de la operación de bloque y cierra la transacción.

Ejemplo:

```
CREATE TABLE ValueTable (id int)
BEGIN TRANSACTION
INSERT INTO ValueTable VALUES(1)
INSERT INTO ValueTable VALUES(2)
ROLLBACK
```

Ejemplo

```
USE Ejercicio
CREATE TABLE ValueTable (id int)
GO
---
DECLARE @TransactionName varchar(20) = 'Transaction1'
BEGIN TRAN @TransactionName
      INSERT INTO ValueTable VALUES(1), (2)
ROLLBACK TRAN @TransactionName
---
INSERT INTO ValueTable VALUES(3),(4);
---
SELECT id FROM ValueTable;
DROP TABLE ValueTable;
```

Ejemplo

```
USE Northwind
BEGIN TRAN
    SELECT ContactName
     FROM Customers
     WHERE CustomerID = 'GREAL‘


 UPDATE Customers
 SET ContactName = 'Howard prueba Snyder_Updated‘
  WHERE CustomerID ='GREAL'

 COMMIT TRAN


SELECT ContactName FROM Customers WHERE CustomerID = 'GREAL'
```

TRY ... CATCH

```
BEGIN TRY
     { sql_statement | statement_block }
END TRY
BEGIN CATCH
     [ { sql_statement | statement_block } ]
END CATCH
```

- Una construcción TRY ... CATCH captura todos los errores de ejecución que tienen una gravedad superior a 10 que no cierran la conexión de la base de datos.
- Un bloque TRY debe ser seguido inmediatamente por un bloque CATCH asociado. La inclusión de cualquier otra instrucción entre las declaraciones END TRY y BEGIN CATCH genera un error de sintaxis.

Recuperando información de error

En el ámbito de un bloque CATCH, las siguientes funciones del sistema se pueden usar para obtener información sobre el error que causó la ejecución del bloque CATCH:

- ERROR_NUMBER () devuelve el número del error.
- ERROR_SEVERITY () devuelve la gravedad.
- ERROR_STATE () devuelve el número de estado del error.
- ERROR_PROCEDURE () devuelve el nombre del procedimiento almacenado o disparador donde ocurrió el error.
- ERROR_LINE () devuelve el número de línea dentro de la rutina que provocó el error.
- ERROR_MESSAGE () devuelve el texto completo del mensaje de error.

```
IF OBJECT_ID ( 'usp_GetErrorInfo', 'P' ) IS NOT NULL
   DROP PROCEDURE usp_GetErrorInfo;
GO

CREATE PROCEDURE usp_GetErrorInfo
AS
SELECT
   ERROR_NUMBER() AS ErrorNumber
   ,ERROR_SEVERITY() AS ErrorSeverity
   ,ERROR_STATE() AS ErrorState
   ,ERROR_PROCEDURE() AS ErrorProcedure
   ,ERROR_LINE() AS ErrorLine
   ,ERROR_MESSAGE() AS ErrorMessage;
GO

BEGIN TRY
  SELECT 1/0;
END TRY

BEGIN CATCH
    EXECUTE usp_GetErrorInfo;
END CATCH;
```

Ejemplo

```
BEGIN TRY
    BEGIN TRANSACTION
        UPDATE cuentas SET balance = balance - 1250 WHERE nombreCliente = 'Antonio';
        UPDATE cuentas SET balance = balance + 1250 WHERE nombreCliente = 'Claudio';
    COMMIT TRANSACTION
    PRINT 'Transacción completada'
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION
    PRINT 'Transacción cancelada'
END CATCH
```

Puntos de recuperacion

Los puntos de recuperación (SavePoints) permiten manejar las transacciones por pasos, pudiendo hacer rollbacks hasta un punto marcado por el savepoint y no por toda la transacción.

EJEMPLO1

En cuanto al SAVE TRAN podemos recordarlo con el siguiente ejemplo:

```
CREATE TABLE Tabla1 (Columna1 varchar(50))
GO

BEGIN TRAN
INSERT INTO Tabla1 VALUES ('Primer valor')
        SAVE TRAN Punto1
        INSERT INTO Tabla1 VALUES ('Segundo valor')
        ROLLBACK TRAN Punto1
        INSERT INTO Tabla1 VALUES ('Tercer valor')
COMMIT TRAN

SELECT * FROM Tabla1
Columna1
----------------------------------------------
Primer valor
Tercer valor
(2 filas afectadas)
```

EJEMPLO2

```
BEGIN TRAN

UPDATE Clientes SET Estado = 'Inactivo' WHERE iddepartamento = 1020

UPDATE Clientes SET Estado = 'DeBaja' WHERE iddepartamento=7025

SAVE TRANSACTION Punto1 -- Guardamos la transaccion (Savepoint)

UPDATE Clientes SET Descripcion = 'Ninguna' WHERE idcliente=5896

-- Este ROLLBACK afecta solo a las instrucciones posteriores al savepoint Punto1.

ROLLBACK TRANSACTION Punto1

-- Confirmamos la transaccion

COMMIT
```