IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
Módulo Base de datos

UNIDAD 7: TRANSACCIONES

Índice

1. TRANSACCIONES EN TRANSACT SQL

Una transacción **es un conjunto de operaciones Transact SQL que se ejecutan como un único bloque**, es decir, si falla **una operación Transact SQL fallan todas**. 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.

El ejemplo clásico de transacción es una transferencia bancaria, en la que quitamos saldo a una cuenta y lo añadimos en otra. Si no somos capaces de abonar el dinero en la cuenta de destino, no debemos quitarlo de la cuenta de origen.

SQL Server funciona por defecto con Transacciones de confirmación automática, es decir, cada **instrucción individual es una transacción y se confirma automáticamente**.

Sobre el ejemplo anterior de la transferencia bancaria, un script debería realizar algo parecido a los siguiente:

```
DECLARE @importe DECIMAL(18,2),
        @CuentaOrigen VARCHAR(12),
        @CuentaDestino VARCHAR(12)
/* Asignamos el importe de la transferencia y las cuentas de origen y destino
*/
SET @importe = 50
SET @CuentaOrigen  = '200700000001'
SET @CuentaDestino = '200700000002'

/* Descontamos el importe de la cuenta origen */
UPDATE CUENTAS
SET SALDO = SALDO - @importe
WHERE NUMCUENTA = @CuentaOrigen

/* Registramos el movimiento */
INSERT INTO MOVIMIENTOS
(IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR, IMPORTE, FXMOVIMIENTO)
SELECT IDCUENTA, SALDO + @importe, SALDO, @importe, getdate()
FROM CUENTAS
WHERE NUMCUENTA = @CuentaOrigen

/* Incrementamos el importe de la cuenta destino */
UPDATE CUENTAS
SET SALDO = SALDO + @importe
WHERE NUMCUENTA = @CuentaDestino

/* Registramos el movimiento */
INSERT INTO MOVIMIENTOS
(IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR, IMPORTE, FXMOVIMIENTO)
SELECT IDCUENTA, SALDO - @importe, SALDO, @importe, getdate()
FROM CUENTAS
WHERE NUMCUENTA = @CuentaDestino
```

Esta forma de actuar seria errónea, ya que **cada instrucción se ejecutaría y confirmaría de forma independiente**, por lo que un error dejaría los **datos erróneos en la base de datos** (¡y ese es el peor error que nos podemos encontrar!)

2. Transacciones implícitas y explícitas

Para agrupar varias sentencias Transact SQL en una única transacción, disponemos de los siguientes métodos:

- **Transacciones explícitas**

Cada transacción se inicia explícitamente con la instrucción **BEGIN TRANSACTION** y se termina explícitamente con una instrucción **COMMIT o ROLLBACK**.

- **Transacciones implícitas**

Se inicia automáticamente una nueva transacción cuando se ejecuta una instrucción que realiza modificaciones en los datos, pero cada transacción se completa explícitamente con una instrucción COMMIT o ROLLBACK.

Para **activar-desactivar el modo de transacciones implícitas** debemos ejecutar la siguiente instrucción.

```
--Activamos el modo de transacciones implicitas
SET IMPLICIT_TRANSACTIONS ON
--Desactivamos el modo de transacciones implícitas
SET IMPLICIT_TRANSACTIONS OFF
```

Cuando la opción ANSI_DEFAULTS está establecida en ON, IMPLICIT_TRANSACTIONS también se establece en ON.

El siguiente ejemplo muestra el script anterior **haciendo uso de transacciones explicitas**:

```
DECLARE @importe DECIMAL(18,2),
              @CuentaOrigen VARCHAR(12),
              @CuentaDestino VARCHAR(12)
/* Asignamos el importe de la transferencia y las cuentas de origen y destino
*/
SET @importe = 50
SET @CuentaOrigen = '200700000002'
SET @CuentaDestino = '200700000001'
BEGIN TRANSACTION -- O solo BEGIN TRAN
 BEGIN TRY
 /* Descontamos el importe de la cuenta origen */
  UPDATE CUENTAS
  SET SALDO = SALDO - @importe
  WHERE NUMCUENTA = @CuentaOrigen
/* Registramos el movimiento */
  INSERT INTO MOVIMIENTOS (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
                           IMPORTE, FXMOVIMIENTO)
  SELECT IDCUENTA, SALDO + @importe, SALDO, @importe, getdate()
  FROM CUENTAS
  WHERE NUMCUENTA = @CuentaOrigen
/* Incrementamos el importe de la cuenta destino */
  UPDATE CUENTAS
  SET SALDO = SALDO + @importe
  WHERE NUMCUENTA = @CuentaDestino
/* Registramos el movimiento */
  INSERT INTO MOVIMIENTOS (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
                          IMPORTE, FXMOVIMIENTO)
  SELECT IDCUENTA, SALDO - @importe, SALDO, @importe, getdate()
  FROM CUENTAS
  WHERE NUMCUENTA = @CuentaDestino
/* Confirmamos la transaccion*/
 COMMIT TRANSACTION          - - O solo COMMIT
END TRY
BEGIN CATCH
/* Hay un error, deshacemos los cambios*/
```

```
   ROLLBACK TRANSACTION -- O solo ROLLBACK
   PRINT 'Se ha producido un error!'
  END CATCH

  --El siguiente ejemplo muestra el mismo script con transacciones implícitas:
  SET IMPLICIT_TRANSACTIONS ON
  DECLARE @importe DECIMAL(18,2),
          @CuentaOrigen VARCHAR(12),
          @CuentaDestino VARCHAR(12)
  /* Asignamos el importe de la transferencia y las cuentas de origen y destino
*/
  SET @importe = 50
  SET @CuentaOrigen = '200700000002'
  SET @CuentaDestino = '200700000001'
  BEGIN TRY
  /* Descontamos el importe de la cuenta origen */
    UPDATE CUENTAS
    SET SALDO = SALDO - @importe
    WHERE NUMCUENTA = @CuentaOrigen
  /* Registramos el movimiento */
    INSERT INTO MOVIMIENTOS (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
                             IMPORTE, FXMOVIMIENTO)
    SELECT IDCUENTA, SALDO + @importe, SALDO, @importe, getdate()
    FROM CUENTAS
    WHERE NUMCUENTA = @CuentaOrigen
  /* Incrementamos el importe de la cuenta destino */
    UPDATE CUENTAS
    SET SALDO = SALDO + @importe
    WHERE NUMCUENTA = @CuentaDestino
  /* Registramos el movimiento */
    INSERT INTO MOVIMIENTOS (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
                            IMPORTE, FXMOVIMIENTO)
    SELECT IDCUENTA, SALDO - @importe, SALDO, @importe, getdate()
    FROM CUENTAS
    WHERE NUMCUENTA = @CuentaDestino
  /* Confirmamos la transaccion*/
   COMMIT TRANSACTION          - - O solo COMMIT
  END TRY
  BEGIN CATCH
  /* Hay un error, deshacemos los cambios*/
   ROLLBACK TRANSACTION -- O solo ROLLBACK
   PRINT 'Se ha producido un error!'
  END CATCH
```

La transacción **sigue activa hasta que emita una instrucción COMMIT o ROLLBACK**. Una vez que la primera transacción se ha confirmado o revertido, se inicia automáticamente una nueva transacción la siguiente vez que la conexión ejecuta una instrucción para modificar datos.

La conexión continúa generando transacciones implícitas hasta que se desactiva el modo de transacciones implícitas.

Podemos **verificar el número de transacciones activas a través de @@TRANCOUNT**.

```
  SET IMPLICIT_TRANSACTIONS ON
  BEGIN TRY
    UPDATE CUENTAS SET FXALTA = FXALTA - 1
    PRINT @@TRANCOUNT
    COMMIT
  END TRY
  BEGIN CATCH
    ROLLBACK
    PRINT 'Error'
  END CATCH
```

3. Transacciones anidadas

Podemos **anidar varias transacciones**. Cuando anidamos varias transacciones **la instrucción COMMIT afectará a la última transacción abierta**, pero **ROLLBACK afectará a todas las transacciones abiertas**.

Un hecho a tener en cuenta, es que, si hacemos ROLLBACK de la transacción superior se desharán también los cambios de todas las transacciones internas, aunque hayamos realizado COMMIT de ellas.

```
BEGIN TRAN
  UPDATE EMPLEADO  SET NOMBRE = 'Pepe'
  WHERE ID=10
BEGIN TRAN
  UPDATE EMPLEADO  SET APELLIDO1 = 'Villa'
  WHERE ID=10
-- Este COMMIT solo afecta a la segunda transacción.
COMMIT
-- Este ROLLBACK afecta a las dos transacciones.
ROLLBACK
```

4. Puntos de recuperación (SavePoint).

Una consideración a tener en cuanta cuando trabajamos con transacciones anidadas es la posibilidad de **utilizar puntos de guardado o SAVEPOINTs**.

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.

El siguiente ejemplo muestra como trabajar con puntos de recuperación:

```
BEGIN TRAN
 UPDATE EMPLEADOS SET NOMBRE = 'Pepiño'
 WHERE ID=11
 UPDATE EMPLEADOS SET APELLIDO1 = 'Vila'
 WHERE ID=11
 SAVE TRANSACTION P1 -- Guardamos la transacción (Savepoint)
 UPDATE EMPLEADOS  SET APELLIDO1 = 'García'
 WHERE ID=11
-- Este ROLLBACK afecta solo a las instrucciones posteriores al savepoint P1.
ROLLBACK TRANSACTION P1
-- Confirmamos la transacción
COMMIT
```