SQL Programming

Módulo 7 - Transacciones

Transacciones explícitas

BEGIN TRANSACTION

Marca el punto de inicio de una transacción local explícita.

Las transacciones explícitas empiezan con la instrucción BEGIN TRANSACTION y acaban con la instrucción COMMIT o ROLLBACK.

Transacciones: nivel de aislamiento

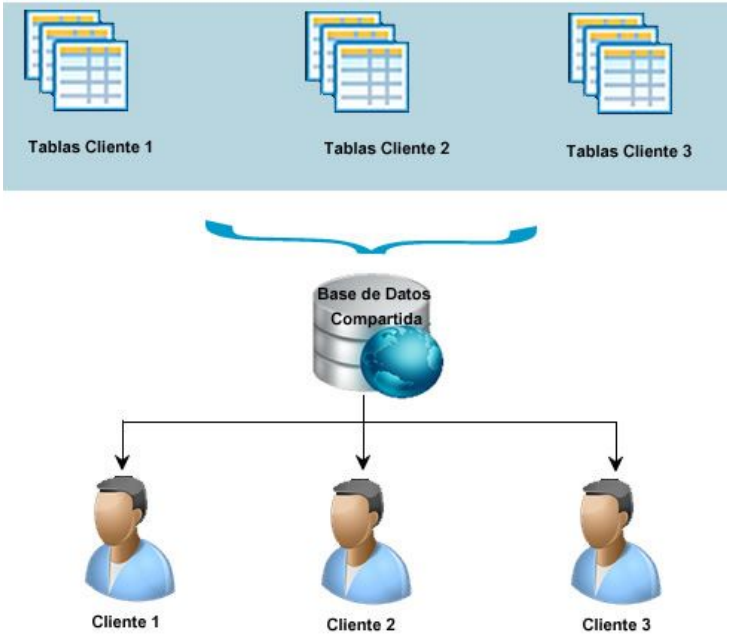
El nivel de aislamiento con el que se ejecuta una instrucción Transact-SQL determina su comportamiento de bloqueo y de versión de fila.

- **READ UNCOMMITTED** es el nivel de aislamiento menos restrictivo porque omite los bloqueos realizados por otras transacciones. Las transacciones que se ejecutan con READ UNCOMMITTED pueden leer valores de datos modificados que aún no han confirmado otras transacciones; éstos se conocen como lecturas no confirmadas.
- **READ COMMITTED** es el nivel de aislamiento predeterminado en SQL Server. Impide las lecturas no confirmadas al especificar que las instrucciones no pueden leer valores de datos modificados que aún no hayan confirmado otras transacciones.
- **REPEATABLE READ** es un nivel de aislamiento más restrictivo que READ COMMITTED. Incluye READ COMMITTED y además especifica que ninguna otra transacción puede modificar ni eliminar datos que la transacción actual haya leído hasta que ésta no se confirme. La simultaneidad es menor que en READ COMMITTED porque durante la transacción se mantienen bloqueos compartidos en los datos leídos en lugar de liberarlos al final de cada instrucción.
-

Transacciones: nivel de aislamiento

- **SNAPSHOT.** La transacción únicamente puede reconocer las modificaciones de datos confirmadas antes del comienzo de la misma. Las instrucciones que se ejecuten en la transacción actual no verán las modificaciones de datos efectuadas por otras transacciones después del inicio de la transacción actual. El efecto es el mismo que se obtendría si las instrucciones de una transacción obtuviese una instantánea de los datos confirmados tal como se encontraban al comienzo de la transacción.
- **SERIALIZABLE** es el nivel de aislamiento más restrictivo, dado que bloquea intervalos enteros de claves y mantiene los bloqueos hasta que la transacción finaliza. Incluye REPEATABLE READ y agrega la restricción de que otras transacciones no pueden insertar filas nuevas en intervalos que haya leído la transacción hasta que ésta no finalice.

Propiedades (ACID)



En el mundo de las bases de datos es muy común escuchar hablar del concepto ACID.

ACID es un grupo de 4 propiedades que garantizan que las transacciones en las bases de datos se realicen de forma confiable. Veamos en detalle este interesante concepto.

Para empezar a definir ACID en el ámbito de las bases de datos, es necesario comprender el concepto de transacción. En las bases de datos, se denomina transacción a una única operación lógica ("de negocio"). Por ejemplo, es una sola transacción la acción de transferir fondos de una cuenta bancaria a otra, aún cuando involucra varios cambios en distintas tablas.

Propiedades (ACID)

- **ATOMICIDAD**

  La **atomicidad** requiere que cada transacción sea "todo o nada": si una parte de la transacción falla, todas las operaciones de la transacción fallan, y por lo tanto la base de datos no sufre cambios.

  Un sistema atómico tiene que garantizar la atomicidad en cualquier operación y situación, incluyendo fallas de alimentación eléctrica, errores y caídas del sistema.

- **CONSISTENCIA**

  La propiedad de **consistencia** se asegura que cualquier transacción llevará a la base de datos de un estado válido a otro estado válido. Cualquier dato que se escriba en la base de datos tiene que ser válido de acuerdo a todas las reglas definidas, incluyendo (pero no limitado a) los constraints, los triggers, y cualquier combinación de estos.

Propiedades (ACID)

- **AISLAMIENTO**

  El **aislamiento** (*"Isolation"* en inglés) se asegura que la ejecución concurrente de las transacciones resulte en un estado del sistema que se obtendría si estas transacciones fueran ejecutadas una detrás de otra. Cada transacción debe ejecutarse en aislamiento total; por ejemplo, si T1 y T2 se ejecutan concurrentemente, luego cada una debe mantenerse independiente de la otra.

- **DURABILIDAD**

  La **durabilidad** significa que una vez que se confirmó una transacción (commit), quedará persistida, incluso ante eventos como pérdida de alimentación eléctrica, errores y caídas del sistema. Por ejemplo, en las bases de datos relacionales, una vez que se ejecuta un grupo de sentencias SQL, los resultados tienen que almacenarse inmediatamente (incluso si la base de datos se cae inmediatamente luego).

Propiedades (ACID)

El siguiente ejemplo abre una transacción explícita y confirma la operación DELETE.

Sintaxis

```
BEGIN TRANSACTION;
DELETE FROM HumanResources.JobCandidate WHERE Job Candidate ID = 13;
COMMIT;
```

El siguiente ejemplo abre una transacción explícita y revierte la operación INSERT.

Sintaxis

```
CREATE TABLE ValueTable (id INT);
BEGIN TRANSACTION;
INSERT INTO ValueTable VALUES(1);
ROLLBACK;
```

Propiedades (ACID)

El siguiente ejemplo abre una transacción explícita con el nombre Candidatos y confirma la operación DELETE.

Sintaxis

```
BEGIN TRANSACTION Candidatos;
DELETE FROM HumanResources.JobCandidate WHERE JobCandidateID = 13;
COMMIT TRANSACTION Candidatos;
```

Propiedades (ACID)

El siguiente ejemplo se nos solicita aumentar el precio de un producto, con la condición que su nuevo valor debe superar al promedio. Si el nuevo valor cumple con la condición se confirm de lo contrario se revierte la transacción

Sintaxis

```
BEGIN TRANSACTION
DECLARE @Promedio Money;
SELECT @Promedio=AVG(ListPrice) FROM Production.Product;
UPDATE Production.Product
SET ListPrice = ListPrice * 1.5 WHERE ProductId = 707;
    IF (SELECT ListPrice FROM Production.Product WHERE ProductId = 707) > @Promedio
        COMMIT TRANSACTION;
    ELSE
        ROLLBACK TRANSACTION;
GO
```

IMPLICIT TRANSACTIONS

Establece el modo BEGIN TRANSACTION en implícito para la conexión.

Notas

Cuando está en ON, el sistema está en modo de transacción implícito. Esto significa que si @@TRANCOUNT = 0, cualquiera de las instrucciones de Transact-SQL siguientes inicia una transacción nueva. Es equivalente a una instrucción BEGIN TRANSACTION oculta que se ejecuta en primer lugar:

Cuando está en OFF, cada una de las instrucciones T-SQL anteriores está limitada por las instrucciones ocultas BEGIN TRANSACTION y COMMIT TRANSACTION. Cuando está en OFF, decimos que el modo de transacción es de confirmación automática. Si el código de T-SQL emite visiblemente una instrucción BEGIN TRANSACTION, se dice que el modo de transacción es explícito.

IMPLICIT TRANSACTIONS

Sintaxis

```
SET IMPLICIT_TRANSACTIONS ON
-- En este caso no es necesario abrir una transacción explícita y se confirma el UPDATE.
UPDATE Production.Product SET ListPrice = 50 WHERE ProductId = 707;
COMMIT;

SELECT ListPrice FROM Production.Product  WHERE ProductId = 707;
----------
50
-- Es necesario abrir una transacción de manera explícita.
SET IMPLICIT_TRANSACTIONS OFF
-- En este caso no se abre ninguna transacción por lo que genera un error.
UPDATE Production.Product SET ListPrice = 0 WHERE ProductId = 707;
ROLLBACK;
----------
La solicitud ROLLBACK TRANSACTION no tiene la correspondiente BEGIN TRANSACTION.
SET IMPLICIT_TRANSACTIONS OFF
-- En este caso se abre una transacción explícita y se confirma el UPDATE.
BEGIN TRAN
UPDATE Production.Product SET ListPrice = 1000 WHERE ProductId = 707;
COMMIT;
SELECT ListPrice FROM Production.Product WHERE ProductId = 707;
----------
1000
```

SAVE TRANSACTION

Establece un punto de retorno dentro de una transacción.

Un usuario puede establecer un punto de retorno, o marcador, dentro de una transacción. El punto de retorno define una ubicación a la que la transacción puede volver si se cancela parte de la transacción de forma condicional. Si se revierte una transacción hasta un punto de retorno, se debe continuar hasta su finalización con más instrucciones Transact-SQL si es necesario y una instrucción COMMIT TRANSACTION o se debe cancelar completamente al revertir la transacción hasta su inicio.

Para cancelar una transacción completa, use el formato ROLLBACK TRANSACTION transaction_name. Se deshacen todas las instrucciones o procedimientos de la transacción.

En una transacción se permiten nombres de puntos de retorno duplicados, pero una instrucción ROLLBACK TRANSACTION que especifique el nombre de un punto de retorno sólo revertirá la transacción hasta la instrucción SAVE TRANSACTION más reciente que también utilice ese nombre.

SAVE TRANSACTION

El siguiente ejemplo se intenta ingresar un producto y luego establecer su precio y de ser posible guardar el usuario que lo ingresó:

```
-- Entidades necesarias
CREATE TABLE Productos (ProductoID INT, Producto VARCHAR(25), Precio MONEY);
CREATE TABLE ProductosLog (ProductoID INT, Usuario VARCHAR(25));

-- Abro una transacción
BEGIN TRAN
     -- Inserta el producto
       INSERT INTO Productos VALUES (1,'Televisor',0);
     -- Registra la operación y el usuario que la realizó
       BEGIN TRAN Log
            INSERT INTO ProductosLog VALUES (1, 'Juan Perez');
       ROLLBACK; -- Algo malo ha ocurrido y se revierte la operación
     -- Actualiza el precio
       UPDATE Productos SET Precio = 100 WHERE PRODUCTO = 1
COMMIT -- Confirma el alta del producto
La solicitud COMMIT TRANSACTION no tiene la correspondiente BEGIN TRANSACTION.

SELECT * FROM Productos WHERE ProductoID = 1;
------------
(0 rows affected)
SELECT * FROM ProductosLog WHERE ProductoID = 1;
------------
(0 rows affected)
-- Abro una transacción
BEGIN TRAN
     -- Inserta el producto
       INSERT INTO Productos VALUES (1,'Televisor',0);
     -- Registra la operación y el usuario que la realizó
       SAVE TRAN registro
            INSERT INTO ProductosLog VALUES (1, 'Juan Perez');
       ROLLBACK TRAN registro -- Algo malo ha ocurrido y se revierte la operación
     -- Actualiza el precio
       UPDATE Productos SET Precio = 100 WHERE ProductoID = 1;
COMMIT -- Confirma el alta del producto

SELECT * FROM Productos WHERE ProductoID = 1;
------------
(1 rows affected)

SELECT * FROM ProductosLog WHERE ProductoID = 1;
------------
(0 rows affected)
```

¡Muchas gracias!

¡Sigamos trabajando!