UNIVERSIDAD TECNOLOGICA
IZUCAR DE MATAMOROS
Organismo Público Descentralizado del Estado de Puebla

PROGRAMA ACADÉMICO DE TECNOLOGÍAS DE LA INFORMACIÓN

Actividad

Tutorial de comandos para transacciones

Como requerimiento parcial para
Acreditar la asignatura de

Base de Datos para Aplicaciones

Que presentan:

Alumno	No. Control
Juana Vivas Villanueva	**09292204**
Greysi Martínez Arce	**09292086**
Antonio Aguilar Galicia	**09292241**

Asesor
Mtro. Gonzalo Rosas Cabrera

Izúcar de Matamoros, Pué., 21 de febrero de 2012

RESUMEN

Una transacción es una unidad lógica de trabajo solicitado por un usuario que se aplicará a los objetos de base de datos SQL Server.

Las transacciones aportan una fiabilidad superior a las bases de datos. Si disponemos de una serie de consultas SQL Server que deben ejecutarse en conjunto, con el uso de transacciones podemos tener la certeza de que nunca nos quedaremos a medio camino de su ejecución. De hecho, podríamos decir que las transacciones aportan una característica de "deshacer" a las aplicaciones de bases de datos.

En este manual mostramos los comandos: start transaction, commit y rollback, su utilidad así también como ejemplos que utilizarán para administrar una base de datos de forma más segura. También muestra como desactivar el modo autocommit que por defecto, SQL Server se ejecuta con el modo autocommit activado. Esto significa que en cuanto ejecute un comando que actualice (modifique) una tabla, SQL Server almacena la actualización en disco.

DESARROLLO

Definición

El concepto transacción proporciona un mecanismo para describir las unidades lógicas del procesamiento de una base de datos. Los sistemas de procesamiento de transacciones son sistemas con grandes base de datos y cientos de usuarios concurrentes que están ejecutando transacciones de crédito, mercado de valores, cajas de supermercados y otros sistemas similares. Estos sistemas requieren una alta disponibilidad y buen tiempo de repuesta para cientos de usuarios concurrentes.

Estados de transacciones y operaciones

Una transacción es una unidad atómica de trabajo que se realiza por completo o bien no se efectúa en absoluto. Para fines de recuperación el sistema necesita mantenerse al tanto de cuando la transacción se inicia, termina y se confirma o aborta. Así pues, el gestor de recuperación se mantiene al tanto de las siguientes operaciones:

BEGIN_TRANSACTION (Inicio_de_transacción): Esta marca el principio de la ejecución de la transacción.

READ (leer) o WRITE (escribir): Éstas especifican operaciones de lectura o escritura de elementos de la base de datos que se ejecutan como parte de la transacción.

END_TRANSACTION (fin_de_transacción): Ésta especifica que las operaciones de LEER o ESCRIBIR de la transacción han terminado y marca el fin de la ejecución de la transacción.

COMMIT_TRANSACTION (confirmar_transacción): Ésta señala que la transacción terminó con éxito y que cualquier cambio (actualizaciones) ejecutado por ella se puede confirmar sin peligro en la base de datos y que no se deshará.

ROLLBACK (restaurar) o ABORT (abortar): Éstas señales indican que la transacción termino sin éxito y que cualquier cambio o efecto que pueda haberse aplicado a la base de datos se debe deshacer.

Iniciar transacciones

Puede iniciar las transacciones del Database Engine (Motor de base de datos) de SQL Server como explícita o de confirmación automática.

- Transacciones explícitas

Inicie explícitamente una transacción ejecutando una instrucción **BEGIN TRANSACTION**.

- Transacciones auto confirmadas

Se trata del modo predeterminado en SQL Server Compact Edition. Una transacción auto confirmado se inicia cuando se inicia la instrucción de la operación y se confirma cuando finaliza la instrucción.

Finalizar transacciones

Puede finalizar las transacciones con una instrucción **COMMIT** o **ROLLBACK**.

- **COMMIT**

Una instrucción **COMMIT** garantiza que todas las modificaciones de la transacción se convierten en una parte permanente de la base de datos. Una instrucción **COMMIT** libera recursos utilizados por la transacción, por ejemplo bloqueos.

- **ROLLBACK**

Si se produce un error en una transacción, o bien si el usuario decide cancelar la transacción, una instrucción **ROLLBACK** deshace la transacción. Una instrucción **ROLLBACK** deshace todas las modificaciones realizadas en la transacción devolviendo los datos al estado en el que se encontraban al iniciar la transacción. Una instrucción **ROLLBACK** libera algunos recursos retenidos por la transacción.

Puntos de recuperación (SavePoint).

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 = 'Devjoker'

WHERE ID=101


UPDATE EMPLEADOS

SET APELLIDO1 = 'Devjoker.COM'

WHERE ID=101


SAVE TRANSACTION P1 -- Guardamos la transaccion (Savepoint)


UPDATE EMPLEADOS

SET APELLIDO1 = 'Otra cosa!'

WHERE ID=101


-- Este ROLLBACK afecta solo a las instrucciones

-- posteriores al savepoint P1.

ROLLBACK TRANSACTION P1


-- Confirmamos la transaccion

COMMIT
```

Ejemplo 1

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.

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

```
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
```

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
```

CONCLUSIONES Y RECOMENDACIONES

La realización de este trabajo nos deja un aprendizaje extenso sobre el manejo de las transacciones de bases de datos. Es de suma importancia conocer cómo se realizan las transacciones, ya que de no hacerlo podría causar serios problemas en los datos, como también conocer cómo se inicia un bloque de instrucciones o como se deshace, como se mencionó anteriormente es recomendable deshabilitar el auto commit.

REFERENCIAS

Elmasri, R., & Navathe, S. B. (2002). Fundamentos de sistemas de bases de datos (3a ed.). México: Pearson.

Connolly, T., & Begg, C. (2005). Sistemas de bases de datos (4a. ed.). México: Pearson.

LuAuF. (27 de Mayo de 2009).
http://luauf.com/2008/10/13/commit-de-transacciones-en-mysql/

Server, S. (18 de 01 de 2008). *Microsoft Technet*:
http://technet.microsoft.com/es-es/library/ms174377.aspx

MySQL. (20 de Junio de 1997):
http://dev.mysql.com/doc/refman/5.0/es/savepoints.html

MySQL. (30 de Enero de 1997):
http://dev.mysql.com/doc/refman/5.0/es/innodb-transactions-with-different-apis.html