Crea las siguientes tablas sobre una bd de prueba

USE PruebaBD;

GO

IF OBJECT\_ID('t2') IS NOT NULL

DROP TABLE t2;

GO

IF OBJECT\_ID('t1') IS NOT NULL

DROP TABLE t1;

GO

CREATE TABLE t1

(col1 INT NOT NULL PRIMARY KEY,

Col2 int);

CREATE TABLE t2

(cola INT NOT NULL Primary KEY REFERENCES t1(col1),

colb int);

1. Ejecuta lo siguiente. Atento a las transacciones. ¿Se revierte la BD al estado anterior al inicio de la transacción?. Fíjate como queda la BD t2

USE PruebaBD;

GO

INSERT INTO t1 VALUES (1,0)

INSERT INTO t1 VALUES (2,0);

INSERT INTO t1 VALUES (3,0);

INSERT INTO t1 VALUES (4,0);

GO

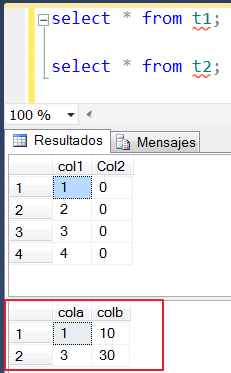
BEGIN TRANSACTION;

INSERT INTO t2 VALUES (1,10);

INSERT INTO t2 VALUES (20,20); -- Foreign key error.

INSERT INTO t2 VALUES (3,30);

COMMIT TRANSACTION;



No se revierte la transacción, porque no hay control de errores. Se insertan todos los valores excepto:

INSERT INTO t2 VALUES (20,20); -- Foreign key error.

Al no tener la referencia 20 creada no puede insertar el valor.

1. Ejecuta lo siguiente.

USE PruebaBD;

DELETE FROM t2;

GO

SET XACT\_ABORT ON;

GO

BEGIN TRANSACTION;

INSERT INTO t2 VALUES (1,10);

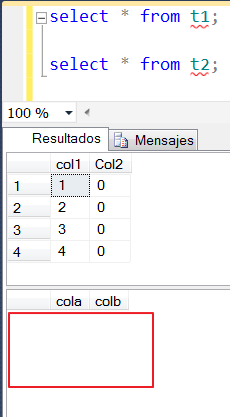
INSERT INTO t2 VALUES (20,20); -- Foreign key error.

INSERT INTO t2 VALUES (3,30);

COMMIT TRANSACTION;

¿Y ahora?, ¿se ha insertado algo en la tabla t2? ¿qué ha cambiado?

No se ha insertado nada en la tabla 2 por la instrucción "SET XACT\_ABORT ON"



Cuando SET XACT\_ABORT es ON, si una instrucción Transact-SQL genera un error en tiempo de ejecución, se termina toda la transacción y se revierte.

1. Usa la bd Prueba. Crea una tabla alumnos:

USE Prueba  
GO

IF OBJECT\_ID(‘ALUMNOS’) IS NOT NULL  
   DROP TABLE ALUMNOS  
GO

CREATE TABLE ALUMNOS (ID INT PRIMARY KEY, NOMBRE VARCHAR(30),  
                       FECHA DATETIME,EDAD INT  
                       CHECK (EDAD >=11 AND EDAD <=18))  
GO

Utilizando TRY and CATCH Realiza un control de errores al insertar el siguiente registro:

   INSERT INTO ALUMNOS VALUES (2,’JULIAN’,’20100101′,3)

Se debe mostrar: el número del error, el número de severidad, el estado y el mensaje de error. (usa las funciones del sistema de ERROR).

USE PruebaDB;

GO

IF OBJECT\_ID('ALUMNOS') IS NOT NULL

DROP TABLE ALUMNOS

GO

CREATE TABLE ALUMNOS (ID INT PRIMARY KEY, NOMBRE VARCHAR(30),

FECHA DATETIME,EDAD INT

CHECK (EDAD >=11 AND EDAD <=18))

GO

BEGIN TRY;

BEGIN TRANSACTION;

INSERT INTO ALUMNOS VALUES (2,'JULIAN','20100101',3);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH;

PRINT ERROR\_NUMBER();

PRINT ERROR\_SEVERITY();

PRINT ERROR\_STATE();

PRINT ERROR\_MESSAGE();

END CATCH

Salida

(0 filas afectadas)

547

16

0

Instrucción INSERT en conflicto con la restricción CHECK "CK\_\_ALUMNOS\_\_EDAD\_\_182C9B23". El conflicto ha aparecido en la base de datos "PruebaDB", tabla "dbo.ALUMNOS", column 'EDAD'.

1. Sobre la tabla anterior crea una transacción con control de errores para la inserción de los siguientes valores en la tabla alumnos:

   INSERT INTO ALUMNOS VALUES (2,’ALDO’,’20100101′,17)                         
   INSERT INTO ALUMNOS VALUES (1,’VICTOR’,’20100101′,18)                         
   INSERT INTO ALUMNOS VALUES (3,’JUAN’,’20100101′,3)

USE PruebaDB;

GO

IF OBJECT\_ID('ALUMNOS') IS NOT NULL

DROP TABLE ALUMNOS

GO

CREATE TABLE ALUMNOS (ID INT PRIMARY KEY, NOMBRE VARCHAR(30),

FECHA DATETIME,EDAD INT

CHECK (EDAD >=11 AND EDAD <=18))

GO

BEGIN TRY;

BEGIN TRANSACTION;

INSERT INTO ALUMNOS VALUES (2,'ALDO','20100101',17);

INSERT INTO ALUMNOS VALUES (1,'VICTOR','20100101',18);

INSERT INTO ALUMNOS VALUES (3,'JUAN','20100101',3);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH;

PRINT ERROR\_NUMBER();

PRINT ERROR\_SEVERITY();

PRINT ERROR\_STATE();

PRINT ERROR\_MESSAGE();

END CATCH

ROLLBACK TRANSACTION;

Salida

(1 filas afectadas)

(1 filas afectadas)

(0 filas afectadas)

547

16

0

Instrucción INSERT en conflicto con la restricción CHECK "CK\_\_ALUMNOS\_\_EDAD\_\_1B0907CE". El conflicto ha aparecido en la base de datos "PruebaDB", tabla "dbo.ALUMNOS", column 'EDAD'.

1. Cree una transacción con control de errores, con las siguientes instrucciones:

* Active la base de datos Banco.
* Aumentar el saldo de la cuenta 3 un 10% (usar un valor concreto pe 300€)
* Descontar el importe correspondiente de la cuenta 4
* Generar los registros correspondientes en la tabla movimientos
* Terminar.

Durante la transacción anterior (antes de su confirmación), ¿Qué información ven los usuarios que consulten las cuentas afectadas?, ¿está actualizada?, explícalo.

De forma predeterminada el nivel de aislamiento de SQl Server es READ COMMITTED (lectura confirmada). Permite que una transacción lea los datos previamente leídos (no modificados) por otra transacción, sin tener que esperar a que la primera transacción finalice. El Motor de base de datos mantiene los bloqueos de lectura (adquiridos en datos seleccionados) hasta el final de la transacción, pero los bloqueos de lectura se liberan tan pronto se efectúa la operación SELECT.

Esto quiere decir que los usuarios que consulten al mismo tiempo que la transacción ven los datos originales (sin modificar). No estarán actualizados.

SET IMPLICIT\_TRANSACTIONS ON

Go

Use banco;

Go

BEGIN TRY;

/\* Descontamos el importe de la cuenta origen \*/

UPDATE cuenta SET saldo=saldo-300 WHERE cod\_cuenta=4;

/\* Registramos el movimiento \*/

INSERT INTO movimiento VALUES (GETDATE(), -300, 117, 4, 9);

/\* Mostramos por pantalla el movimiento realizado \*/

SELECT tipo, getdate() fecha, saldo - 300 saldo, cod\_cuenta

FROM cuenta

WHERE cod\_cuenta = 4;

/\* Incrementamos el importe de la cuenta destino \*/

UPDATE cuenta SET saldo=saldo+300 WHERE cod\_cuenta=3;

/\* Registramos el movimiento \*/

INSERT INTO movimiento VALUES (GETDATE(), 300, 117, 3, 10);

/\* Mostramos por pantalla el movimiento realizado \*/

SELECT tipo, getdate() fecha, saldo + 300 saldo, cod\_cuenta

FROM cuenta

WHERE cod\_cuenta = 3;

/\* Confirmamos la transaccion\*/

COMMIT;

END TRY

BEGIN CATCH;

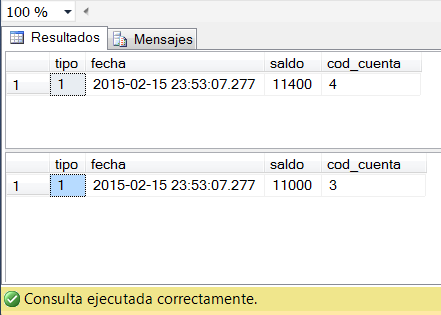
/\* Hay un error, deshacemos los cambios\*/

ROLLBACK TRANSACTION;

PRINT 'Se ha producido un error!';

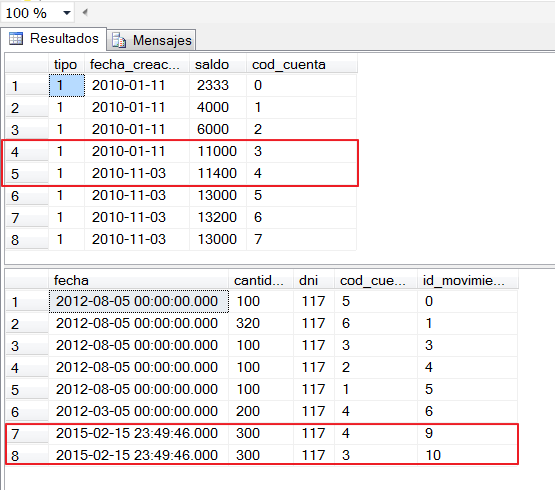
END CATCH;

|  |  |
| --- | --- |
|  |  |



select \* from cuenta;

select \* from movimiento;



Para conocer el nivel de aislamiento de la base de datos podemos ejecutar:

SELECT CASE transaction\_isolation\_level

WHEN 0 THEN 'Unspecified'

WHEN 1 THEN 'ReadUncommitted'

WHEN 2 THEN 'ReadCommitted'

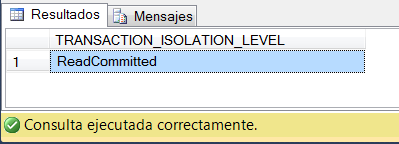
WHEN 3 THEN 'Repeatable'

WHEN 4 THEN 'Serializable'

WHEN 5 THEN 'Snapshot' END AS TRANSACTION\_ISOLATION\_LEVEL

FROM sys.dm\_exec\_sessions

where session\_id = @@SPID



Fuente:

<https://technet.microsoft.com/es-es/library/jj856598%28v=sql.110%29.aspx>

<http://www.devjoker.com/contenidos/Articulos/210/Bloqueos-y-nivel-de-aislamiento-en-SQL-Server-2005.aspx>