lsi Departamento de Lenguajes y Sistemas Informáticos
Universitat d'Alacant
Universidad de Alicante

Programación básica de servidor de base de datos (2): transacciones

Esta sesión se centra en la implementación de transacciones en SQL Server aunque, nuevamente, desde la óptica del refresco más que de la introducción de conceptos novedosos o técnicas avanzadas.

Objetivos

1. ¿Qué es una transacción y para qué se necesita? Es indispensable entender el ámbito de aplicación y su motivación.
2. ¿Cómo se implementa una transacción? Alguna orden de T-SQL será la que determina qué parte del código es transacción y cuál no. Especialmente importante es entender
3. ¿cómo se para una transacción? Y, una vez detenida, dependiendo de si ha habido errores o no, qué hace el sistema a continuación. Por último
4. ¿qué son los niveles de aislamiento? Es uno de los aspectos más importantes de toda la sesión, determina el comportamiento de los procesos afectados.

Conceptos necesarios para la asignatura

Pseudocódigo

```
BEGIN TRAN
      Operación 1...
      Si fallo: ROLLBACK TRAN

       Operación 2....
       Si fallo: ROLLBACK TRAN
...

      Operación N....
      Si fallo: ROLLBACK TRAN
COMMIT TRAN
```

Tratamiento de errores

```
BEGIN TRAN
DECLARE @ErrorVar INT;
DECLARE @RowCountVar INT;

-- Ejemplo con UPDATE
UPDATE ...
    SET ...
    WHERE ...;
-- Save the @@ERROR and @@ROWCOUNT values in local
-- variables before they are cleared.
SELECT @ErrorVar = @@ERROR
    ,@RowCountVar = @@ROWCOUNT;

-- Check for errors
-- Por ejemplo, un intento de violación de integridad
-- referencial error #547.
IF @ErrorVar <> 0
   ROLLBACK

COMMIT TRAN
```

Presentación de la segunda sesión: Transacciones en SQL Server

Extractos del manual

- Transacciones
- BEGIN TRANSACTION
- @@TRANCOUNT
- @@ERROR
- Usar TRY...CATCH en T-SQL
- COMMIT TRANSACTION

- ROLLBACK TRANSACTION
  - SAVE TRANSACTION
- Descripción de los niveles de aislamiento
  - Ajustar los niveles de aislamiento de transacción
  - Aislamiento (ACID), efectos de lectura
- Guía de versiones de fila y bloqueo de transacciones de SQL Server (2014)

¿Qué has aprendido?

Contesta a estas preguntas en Moodle (cuestionario "Transacciones información"):

- ¿Qué es una transacción?
- ¿Para qué se necesitan transacciones?
- ¿Cómo puedo conocer el número de transacciones activas?
- Para especificar y exigir transacciones, ¿qué proporciona el motor de base de datos?
- Cuando una transacción está operando, ¿qué puede bloquear el motor de la base de datos puede bloquear?
- Cuanto menor es la granularidad del bloqueo, ¿aumenta o disminuye la simultaneidad? ¿Y la sobrecarga del motor?
- ¿Qué es un dato sucio?
- ¿Qué es una lectura fantasma?
- ¿Qué es una lectura no repetible?

¿Cómo funcionan las transacciones en SQL Server?

Cómo trabajar con los ejemplos de transacciones

Versión en texto

- Locking, Blocking and Deadlocking, Wayne Sheffield, SQLServerCentral, 2013
- Managing Transaction Logs, Gail Shaw, SQLServerCentral, 2008

- Transaction Isolation Levels, Wayne Sheffield, SQLServerCentral, 2014
- El registro de transacciones
  - Una guía para principiantes acerca de los registros de transacciones SQL Server, diciembre 4, 2015 by Ivan Stankovic
- Realizar copias de seguridad y restaurar bases de datos de SQL Server
  - Aplicar copias de seguridad de registros de transacción
  - Backup / Restore
  - Modelos de recuperación
  - How to schedule and automate backups of SQL Server databases in SQL Server Express

Ejemplos de transacciones

Preparación de estructuras y datos para los ejercicios

Estas son las tablas sobre las que vamos a trabajar. Se necesita haber creado, previamente, una base de datos llamada "ventas".

```
USE master
GO
IF DB_ID (N'ventas') IS NOT NULL
DROP DATABASE ventas;
GO
CREATE DATABASE ventas
GO
use ventas
go
if OBJECT_ID ('trantest2','U') is not null DROP table trantest2
if OBJECT_ID ('trantest','U') is not null DROP table trantest
go
CREATE TABLE trantest (Col1 int primary key, Col2 int)

CREATE TABLE trantest2 (ColA int primary key references trantest, ColB int)

CREATE TABLE cuenta (id integer primary key, saldo decimal(6,2))
```

Ejemplo de algo que necesita transacciones

En este ejemplo, mediante la instrucción *waitfor*, simulamos el caso de un proceso que se cuela mientras otro no ha terminado. Prepara 2 consultas en paralelo y ejecuta A; y mientras A se demora, ejecuta B. ¿Qué ves de raro con el resultado final, en el estado final de la base de datos?

conexión A	conexión B
use ventas go delete from cuenta; insert into cuenta values (1,**6000.00**); select saldo 'inicial' from cuenta declare @x decimal(6,2); select @x=saldo from cuenta where id = 1; **set @x = @x - 2000** select @x 'variableX' **waitfor delay '00:00:10'** update cuenta set saldo = @x where id = 1 select saldo 'final' from cuenta	use ventas go select saldo 'inicial' from cuenta declare @x decimal(6,2); select @x=saldo from cuenta where id = 1; **set @x = @x + 3000** select @x 'variableX' update cuenta set saldo = @x where id = 1 select saldo 'final' from cuenta

Resultados de las transacciones

Anota en un documento de texto las respuestas a las preguntas numeradas que acompañan al código a ejecutar y entrégalo en Moodle, preferiblemente en PDF.

Comportamiento esperado de una transacción

T1a

Ejecuta lo siguiente. Atento a la transacción ¿se revierte la base de datos al estado anterior al inicio de la misma?

```
use ventas;
delete from trantest2;
delete from trantest;
insert into trantest values (1,0);
insert into trantest values (2,0);
insert into trantest values (3,0);
insert into trantest values (4,0);
SET XACT_ABORT OFF
-- solo se revierte la instrucción Transact-SQL que generó el error y la transacción continúa procesándose
BEGIN TRAN
insert into trantest2 values (1,10);
insert into trantest2 values (20,20);
insert into trantest2 values (3,30);
COMMIT
SELECT * FROM trantest2;
```

1. ¿Qué ha ocurrido?

T1b

Ejecuta lo siguiente.

```
use ventas;
delete from trantest2;
SET XACT_ABORT ON
-- si una instrucción Transact-SQL genera un error en tiempo de ejecución,
-- se termina toda la transacción y se revierte
BEGIN TRAN
insert into trantest2 values (1,10);
insert into trantest2 values (20,20);
insert into trantest2 values (3,30);
COMMIT
SELECT * FROM trantest2;
```

2. ¿Y ahora? ¿Se ha insertado algo? ¿Qué ha cambiado?

Control de errores

Ejecuta lo siguiente.

```
use ventas;
delete from trantest2;
SET XACT_ABORT ON
BEGIN TRY
   BEGIN TRANSACTION;
         insert into trantest2 values (1,10);
         insert into trantest2 values (20,20);
         insert into trantest2 values (3,30);
  COMMIT
END TRY
BEGIN CATCH
  select 'trancount', @@TRANCOUNT; -- informativo, verificar cuántas transacciones activas
   IF @@TRANCOUNT > 0
      ROLLBACK;
  select 'trancount', @@TRANCOUNT; -- meramente informativo, para comprobar
  SELECT ERROR_NUMBER() AS ErrorNumber;
  SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
SELECT * FROM trantest2;
```

3. ¿Qué ha ocurrido?

Transacciones implícitas y autoconfirmadas

T1c

Tipos de transacciones: implícitas y autoconfirmadas. Son otros modos de tratar transacciones, simplemente comprobar su comportamiento. Cuando SET IMPLICIT_TRANSACTIONS está configurado como OFF, **si no usamos BEGIN TRAN**, ciertas órdenes se ejecutan en modo de CONFIRMACIÓN AUTOMÁTICA:

- ALTER TABLE
- BEGIN TRANSACTION
- CREATE
- DELETE
- DROP
- FETCH
- GRANT
- INSERT
- OPEN
- REVOKE
- SELECT (excepto cuando no se utilizan tablas, SELECT GETDATE(), SELECT 1)
- TRUNCATE TABLE
- UPDATE

Cuando usamos **SET IMPLICIT_TRANSACTIONS ON** es como si cada instrucción tuviera un BEGIN TRAN invisible, pero necesita confirmación. Esa confirmación se la puede dar una siguiente instrucción o una orden COMMIT.

Ejecuta lo siguiente en dos ventanas de consulta, y en el orden que te mostramos:

Consulta A	Consulta B
use ventas; -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback; delete from trantest2; go **SET IMPLICIT_TRANSACTIONS ON** -- inicia transacción insert into trantest2 values (1,10); **select** 'insert 1', @@TRANCOUNT; -- confirma la anterior transacción e inicia otra **insert** into trantest2 values (2,20); select 'insert 2', @@TRANCOUNT;	
	use ventas; SET TRANSACTION ISOLATION LEVEL **READ COMMITTED** go **SELECT** * FROM trantest2;
-- esto confirma la anterior transacción e inicia otra **insert** into trantest2 values (3,20),(4,40); select 'insert 3', @@TRANCOUNT; COMMIT -- la última necesita confirmación select 'commit', @@TRANCOUNT;	
	SET IMPLICIT_TRANSACTIONS OFF -- como no iniciamos con BEGIN TRAN, las -- transacciones son de CONFIRMACIÓN AUTOMÁTICA --(cada orden se confirma automáticamente) delete from trantest2; **update** trantest set col2=5 WHERE Col2 = 0 select '¿cuántas trans?', @@TRANCOUNT; SELECT * FROM trantest;

4. ¿Qué ha ocurrido?

Interbloqueos

Se producen entre dos transacciones que bloquean los mismos recursos en orden inverso. Por ejemplo:

Preparación de datos

```
use ventas
go
if OBJECT_ID ('trantest2','U') is not null DROP table trantest2
if OBJECT_ID ('trantest','U') is not null DROP table trantest
go
CREATE TABLE trantest (Col1 int primary key, Col2 int)
CREATE TABLE trantest2 (ColA int primary key, ColB int)
go
insert into trantest values (1,0);
insert into trantest2 values (4,40);
```

Ejecuta en paralelo estos dos lotes

Consulta A	Consulta B
SET XACT_ABORT ON use ventas go if @@TRANCOUNT>0 rollback go BEGIN TRAN UPDATE **trantest** SET col2 = 1000 where col1=1; **waitfor delay '00:00:05'** UPDATE **trantest2** SET colB = 1000 where colA=4; COMMIT SELECT * FROM trantest SELECT * FROM trantest2	SET XACT_ABORT ON use ventas go if @@TRANCOUNT>0 rollback go BEGIN TRAN UPDATE **trantest2** SET colB = 4000 where colA=4; **waitfor delay '00:00:05'** UPDATE **trantest** SET col2 = 4000 where col1=1; COMMIT SELECT * FROM trantest SELECT * FROM trantest2

5. Revisa los mensajes de error para saber qué ha pasado.

De los niveles de aislamiento

En esta serie de ejercicios, se va a simular un entorno concurrente en el que dos transacciones acceden a los mismos datos en diferentes condiciones de aislamiento. Como se explica en "Entorno práctico para transacciones", las tablas siguientes han de interpretarse como ejecuciones ordenadas de trozos de código. El objetivo es comprobar cuándo una transacción se detiene por esperar un recurso de la otra, y el resultado final que se obtiene en las tablas afectadas.

T2a1

Seleccionando parte del texto y ejecutando (recuerda que SQL Server permite lanzar solo aquello que esté marcado) debes realizar la siguiente secuencia de ejecuciones:

Consulta A	Consulta B
use ventas; SET IMPLICIT_TRANSACTIONS OFF WHILE @@TRANCOUNT>0 rollback SET TRANSACTION ISOLATION LEVEL SERIALIZABLE BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); insert into trantest2 values (4,40); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go BEGIN TRAN INSERT INTO trantest (Col1,Col2) VALUES (5,3)	
	use ventas; SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO -- como no iniciamos con BEGIN TRAN -- las transacciones son de -- CONFIRMACIÓN AUTOMÁTICA --(cada orden se confirma automáticamente) SELECT * FROM trantest WHERE Col2 = 3
ROLLBACK	
	SELECT * FROM trantest WHERE Col2 = 3

6. ¿Quién espera a quién? (nadie/la consulta A/la consulta B)
7. ¿Qué resultados obtienen las *select* de la consulta B y por qué?

T2a1B

Seleccionando parte del texto y ejecutando (recuerda que SQL Server permite lanzar solo aquello que esté marcado) debes realizar la siguiente secuencia de ejecuciones:

Consulta A	Consulta B
use ventas; while @@TRANCOUNT>0 rollback SET IMPLICIT_TRANSACTIONS OFF SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); insert into trantest2 values (4,40); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** BEGIN TRAN **UPDATE** trantest set Col2=10 where **col1=2**;	
	while @@TRANCOUNT>0 rollback SET IMPLICIT_TRANSACTIONS OFF SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go BEGIN TRAN **UPDATE** trantest set col2=20 where **col1=2**; **COMMIT** select * from trantest; --consulta B final
COMMIT select * from trantest; - consulta A inicial select * from trantest; -- consulta A final	

8. ¿Quién espera a quién? (nadie/la consulta A/la consulta B)
9. ¿Qué resultados obtiene la *consulta final* de la consulta A y por qué?

Ten en cuenta que cualquier escritura implica un bloqueo exclusivo. Los niveles de aislamiento sirven para que las lecturas de datos sean más o menos "fiables", pero escribir siempre bloquea a cualquier otra transacción que intente la misma operación, aunque sea READ UNCOMMITED.

T2a2

Seleccionando parte del texto y ejecutando (recuerda que SQL Server permite lanzar solo aquello que esté marcado) debes realizar la siguiente secuencia de ejecuciones:

Consulta A	Consulta B
use ventas; if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); insert into trantest2 values (4,40); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go BEGIN TRAN **INSERT** INTO trantest (Col1,Col2) VALUES (5,3)	
	use ventas go SET TRANSACTION ISOLATION LEVEL **READ COMMITTED** GO -- como no iniciamos con BEGIN TRAN -- las transacciones son de -- CONFIRMACIÓN AUTOMÁTICA --(cada orden se confirma automáticamente) **SELECT** * FROM trantest WHERE Col2 = 3
ROLLBACK	
	SELECT * FROM trantest WHERE Col2 = 3

10. ¿Quién espera a quién? (nadie/la consulta A/la consulta B)
11. ¿Qué resultados obtienen las *select* de la consulta B y por qué?

T2b

Ejecuta:

Consulta A	Consulta B
use ventas; if @@TRANCOUNT>0 rollback; SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
use ventas; SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ** GO BEGIN TRAN -- select 1 SELECT * FROM trantest WHERE Col2 = 2	
	use ventas; SET TRANSACTION ISOLATION LEVEL READ COMMITTED go -- como no iniciamos con BEGIN TRAN -- las transacciones son de -- CONFIRMACIÓN AUTOMÁTICA --(cada orden se confirma automáticamente) update trantest set col2 = 1 where col2=2;
-- select 2 SELECT * FROM trantest WHERE Col2 = 2 COMMIT -- select fin SELECT * FROM trantest	

12. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
13. ¿Es el resultado de "select 1" igual al de "select 2"?
14. ¿Es diferente el resultado de "select fin?
15. Prueba a realizar el mismo ejercicio pero, en la consulta B, con "update trantest set col2=1 where col2=0;"; ¿qué pasa?
16. Prueba a realizar el mismo ejercicio pero, en la consulta B, con "update trantest set col2=1;"; ¿qué pasa?

T2c

Consulta A	Consulta B
use ventas go if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
use ventas go **update trantest set col2 = 2;** SET TRANSACTION ISOLATION LEVEL **READ COMMITTED** GO BEGIN TRAN SELECT * FROM trantest WHERE Col2 = 2 -- select 1	
	use ventas go SET TRANSACTION ISOLATION LEVEL **READ COMMITTED** go select * from trantest; update trantest set col2 = 1 where col2=2; select * from trantest;
SELECT * FROM trantest WHERE Col2 = 2 -- select 2 COMMIT	
-- select fin SELECT * FROM trantest	

La transacción A, en realidad, no escribe nada, simplemente consulta. Lo que queremos ver es si, a pesar del nivel de aislamiento, A lee lo mismo en las dos consultas o no. La que decide si alguien espera o no es el nivel de la transacción B.

17. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
18. ¿Es el resultado de "select 1" igual al de "select 2"?

19. ¿Es diferente el resultado de "select fin"?
20. ¿Qué diferencia hay entonces entre las transacciones de la consulta A en este ejercicio y el anterior?

T2c2

Ejecuta

Consulta A	Consulta B
use ventas go if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
	use ventas go SET TRANSACTION ISOLATION LEVEL READ COMMITTED go BEGIN TRAN select * from trantest; update trantest set col2 = 1 where col2=2; select * from trantest;
use ventas go SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ** GO BEGIN TRAN SELECT * FROM trantest WHERE Col2 = 2 -- select 1	
	COMMIT
SELECT * FROM trantest WHERE Col2 = 2 -- select 2 COMMIT	
SELECT * FROM trantest -- select fin	

21. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
22. ¿Es el resultado de "select 1" igual al de "select 2"?
23. ¿Es diferente el resultado de "select fin"?

T2c22

Ejecuta

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
	use ventas go SET TRANSACTION ISOLATION LEVEL READ COMMITTED go BEGIN TRAN select * from trantest; update trantest set col2 = 1 where col2=2; select * from trantest;
SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ** GO BEGIN TRAN update trantest set col2 = 2;	
	COMMIT
SELECT * FROM trantest WHERE Col2 = 2 -- select 2 COMMIT	
SELECT * FROM trantest -- select fin	

24. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)

T2c22a

Ejecuta

Consulta A	Consulta B
use ventas; SET IMPLICIT_TRANSACTIONS OFF WHILE @@TRANCOUNT>0 rollback SET TRANSACTION ISOLATION LEVEL SERIALIZABLE BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); insert into trantest2 values (4,40); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go BEGIN TRAN INSERT INTO trantest2 SELECT (select sum(Col1) from trantest where Col2=2), 100;	
	use ventas; SET IMPLICIT_TRANSACTIONS OFF WHILE @@TRANCOUNT>0 rollback SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO BEGIN TRAN UPDATE trantest SET Col2=2 where Col1=2;
UPDATE trantest2 SET ColB=ColB+10 where ColA=(select sum(Col1) from trantest where Col2=2); --el cálculo depende de las filas recuperadas COMMIT; select * from trantest; select * from trantest2;	
	ROLLBACK; --algo malo ha pasado select * from trantest; select * from trantest2;

25. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
26. ¿Qué ha ocurrido?

T2c22b

Ejecuta

Consulta A	Consulta B
use ventas; SET IMPLICIT_TRANSACTIONS OFF WHILE @@TRANCOUNT>0 rollback SET TRANSACTION ISOLATION LEVEL SERIALIZABLE BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); insert into trantest2 values (4,40); COMMIT	
SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ** go BEGIN TRAN INSERT INTO trantest2 SELECT (select sum(Col1) from trantest where Col2=2), 100;	
	use ventas; SET IMPLICIT_TRANSACTIONS OFF WHILE @@TRANCOUNT>0 rollback SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO BEGIN TRAN UPDATE trantest SET Col2=2 where Col1=2;
UPDATE trantest2 SET ColB=ColB+10 where ColA=(select sum(Col1) from trantest where Col2=2); --el cálculo depende de las filas recuperadas COMMIT; select * from trantest; select * from trantest2;	
	ROLLBACK; --algo malo ha pasado select * from trantest; select * from trantest2;

27. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
28. ¿Qué diferencias observas entre los resultados de esta ejecución y la anterior T2c22a?

T2d1

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
update trantest set col2 = 2; COMMIT SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ** GO BEGIN TRAN SELECT * FROM trantest WHERE Col2 = 2 -- select 1	
	use ventas go SET TRANSACTION ISOLATION LEVEL READ COMMITTED go -- como no iniciamos con BEGIN TRAN -- las transacciones son de -- CONFIRMACIÓN AUTOMÁTICA --(cada orden se confirma automáticamente) **insert into trantest values (100,2);**
SELECT * FROM trantest WHERE Col2 = 2 -- select 2 COMMIT	
SELECT * FROM trantest -- select fin	

29. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
30. ¿Es el resultado de "select 1" igual al de "select 2"?
31. ¿Es diferente el resultado de "select fin"?

T2d2

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
use ventas go **update trantest set col2 = 2;** SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE** GO BEGIN TRAN SELECT * FROM trantest WHERE Col2 = 2 -- select 1	
	use ventas go SET TRANSACTION ISOLATION LEVEL READ COMMITTED go **insert into trantest values (100,2);**
SELECT * FROM trantest WHERE Col2 = 2 -- select 2 COMMIT	
SELECT * FROM trantest -- select fin	

32. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
33. ¿Es el resultado de "select 1" igual al de "select 2"?
34. ¿Es diferente el resultado de "select fin"?
35. ¿Qué ha ocurrido entre las dos transacciones y por qué?

T2e

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO BEGIN TRAN	
	use ventas go SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go BEGIN TRAN insert into trantest values (100,2); SELECT * from trantest -- select B1
SELECT * FROM trantest -- select 1 delete from trantest where col2 = 2; SELECT * FROM trantest -- select 2	
	COMMIT SELECT * from trantest -- select fin B1
ROLLBACK SELECT * FROM trantest -- select fin	

36. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
37. Si alguien ha esperado ¿para qué sirve READ UNCOMMITTED?

T2f

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO BEGIN TRAN SELECT * FROM trantest -- select 1 delete from trantest where col2 = 2; SELECT * FROM trantest -- select 2	
	use ventas go SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** go SELECT sum(col1) FROM trantest; -- select B1
ROLLBACK SELECT * FROM trantest -- select fin	
	SELECT sum(col1) FROM trantest; -- select finB

38. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
39. No obtienen el mismo resultado "select B1" y "select finB ¿por qué?

T2f2

Consulta A	Consulta B
use ventas go -- por si se nos ha quedado algo por ahí if @@TRANCOUNT>0 rollback go -- preparar el ejercicio SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED** GO BEGIN TRAN SELECT * FROM trantest -- select 1 delete from trantest where col2 = 2; SELECT * FROM trantest -- select 2	
	use ventas go SET TRANSACTION ISOLATION LEVEL **READ COMMITTED** go SELECT sum(col1) FROM trantest; -- select B1
ROLLBACK SELECT * FROM trantest -- select fin	
	SELECT sum(col1) FROM trantest; -- select finB

40. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
41. "Select B1" y "select finB" ¿obtienen el mismo resultado?

Instantáneas

Cierra todas las consultas y abre una nueva. Ejecuta lo siguiente:

ALTER DATABASE ventas SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE ventas SET READ_COMMITTED_SNAPSHOT ON

Hemos habilitado el aislamiento de instantáneas. Esto hace que podamos elegir el nivel de aislamiento correspondiente.

T2c-S

Consulta A	Consulta B
use ventas; if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); **update trantest set col2 = 2;** COMMIT	
SET TRANSACTION ISOLATION LEVEL **SNAPSHOT** GO BEGIN TRAN SELECT * FROM trantest WHERE Col2 = 2 -- select 1	
	use ventas go SET TRANSACTION ISOLATION LEVEL READ COMMITTED go update trantest set **col2 = 1** where **col2=2**;
SELECT * FROM trantest -- select 2 COMMIT SELECT * FROM trantest-- select fin	

42. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
43. ¿Es el resultado de "select 1" igual al de "select 2"?
44. ¿Es diferente el resultado de "select fin"?
45. ¿Qué diferencia hay entonces entre las transacciones de la consulta A en este ejercicio y los anteriores?

T2c-S2

Consulta A	Consulta B
use ventas go if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
use ventas go SET TRANSACTION ISOLATION LEVEL **SNAPSHOT** GO BEGIN TRAN update trantest set **col2 = (select col2 from trantest where col1=1)** where **col2=2**; SELECT * FROM trantest -- select 2	
	use ventas go SET TRANSACTION ISOLATION LEVEL **SNAPSHOT** GO go update trantest set **col2 = 2** where **col2=0**; SELECT * FROM trantest-- select fin2
COMMIT SELECT * FROM trantest-- select fin1	

46. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
47. ¿Qué ha pasado?

T2c-S3

Consulta A	Consulta B
use ventas go if @@TRANCOUNT>0 rollback go SET TRANSACTION ISOLATION LEVEL SERIALIZABLE go BEGIN TRAN delete from trantest2; delete from trantest; insert into trantest values (1,0),(2,0),(3,2),(4,2); COMMIT	
use ventas go SET TRANSACTION ISOLATION LEVEL **SNAPSHOT** GO BEGIN TRAN update trantest set **col2 = 3** where **col2=2**; SELECT * FROM trantest -- select 2	
	use ventas go SET TRANSACTION ISOLATION LEVEL **SNAPSHOT** GO go update trantest set **col2 = 2** where **col2=2**; SELECT * FROM trantest-- select fin
COMMIT SELECT * FROM trantest-- select fin	

48. ¿Quién ha esperado por quién? (nadie/la consulta A/la consulta B)
49. ¿Qué ha pasado?