Para ilustrar los ejemplos,  usaremos la tabla **“prueba transacciones”** la cual se compone de un id (campo clave) y una descripción generada con el siguiente script:

CREATE TABLE [dbo].[pruebatransacciones](

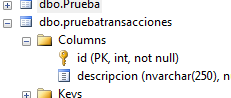
[id] [int] NOT NULL,

[descripcion] [nvarchar](250) NULL,

CONSTRAINT [PK\_pruebatransacciones] PRIMARY KEY (id)

)

GO



1-Explorador de Objetos de SQLSERVER

**READ COMMITED:**

Es el nivel de aislamiento por defecto en SQL SERVER,  una transacción debe esperar que finalice otra si quiere leer un recurso modificado por la primera:

Ejemplo 1:

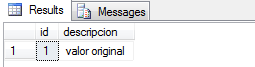
1. En una nueva instancia de SQL SERVER (instancia 1):
   1. Borramos los datos de la tabla de prueba:

DELETE FROM pruebatransacciones

1. Insertamos un valor inicial original:

Insert dbo.pruebatransacciones(id,descripcion) values('1','valor original');

Los datos de la tabla  pruebatransacciones quedarían de la siguente manera:

  
2- Vista parcial en instancia uno.

2. El paso siguiente será iniciar una transacción y realizar un update sobre el valor previamente creado (valor original):

begin transaction

update dbo.pruebatransacciones

set descripcion = 'cambio'

**nota: Estas instrucciones dejarán la transacción abierta a la espera del cierre** (commit transaction)

 commit;

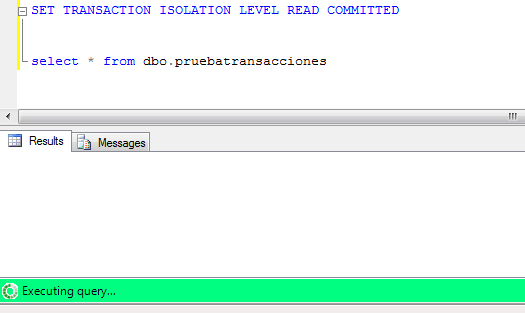
3. Luego, desde otra instancia (consulta) de SQL SERVER (instancia 2):

a. Establecemos el nivel de aislamiento (recordar que es el valor por defecto)

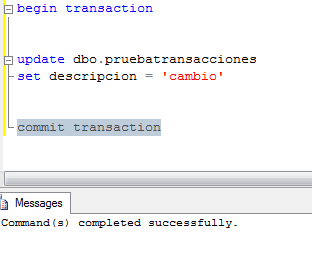
SET TRANSACTION ISOLATION LEVEL READ COMMITTED

b. Luego consultamos la tabla:

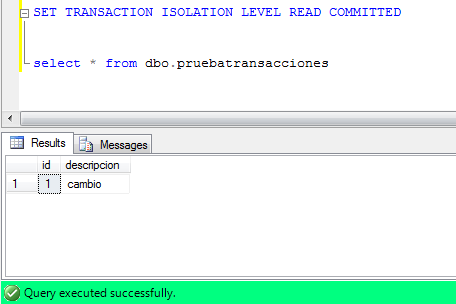
select \* from dbo.pruebatransacciones

  
3- La consulta no puede terminar.

No es posible obtener los resultados de la tabla, ya que otra transacción ha sido iniciada sobre la misma tabla. Al terminarla en la instancia (consulta) 1 (solo ejecuto el commit transaction):

  
4- Se ejecuta el commit pendiente de la instancia uno

Inmediatamente, al ver la segunda instancia la consulta realizada ya tendrá sus resultados:



5- Solo al terminar la transacción uno, la instancia dos muestra los resultados

**READ UNCOMMITED:**

También llamado “Lectura sucia”. La segunda transacción puede leer aunque la primera no haya sido completada:

Ejemplo 2:

1. En la instancia 1:

1. Borramos los datos de la tabla de prueba:

DELETE FROM pruebatransacciones

1. Insertamos un valor inicial original:

Insert dbo.pruebatransacciones(id,descripcion) values('1','valor original');

2. Nuevamente iniciamos una transacción y realizamos un update sobre el valor previamente creado (valor original):

begin transaction

update dbo.pruebatransacciones

set descripcion = 'cambio'

**nota: Estas instrucciones dejarán la transacción abierta a la espera del cierre** (commit transaction)

 rollback

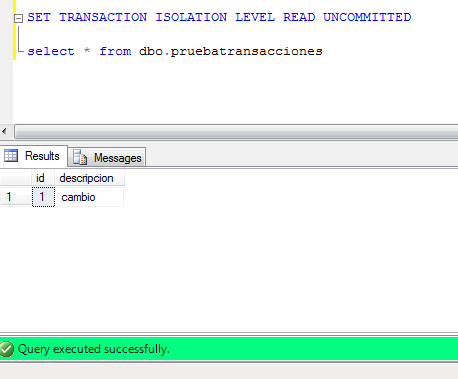
3. Luego, desde otra instancia de SQL SERVER (instancia 2):

1. Establecemos el nivel de aislamiento a testear:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

1. Luego consultamos la tabla pudiendo ver el cambio parcial realizado en la instancia uno:

select \* from dbo.pruebatransacciones

  
  6- Se obtiene el valor “parcial” en la instancia dos.  
  
El nombre lectura sucia, se debe a que si la transacción uno realiza un rollback no modificándose el valor, se habrá leído un valor erróneo.

**SNAPSHOT:**

En este nivel de aislamiento, los cambios que produce una primera transacción sobre un recurso serán ignorados si se comienza una nueva transacción antes de finalizar la primera. Es decir,  cuando  se leen filas modificadas por otras transacciones, se recupera la versión de la fila que existía cuando empezó la transacción  
 NOTA: Hay que permitir en la BD ese nivel de aislamiento.

ALTER DATABASE pruebaBD

SET ALLOW\_SNAPSHOT\_ISOLATION ON

Ejemplo 3:

1. En la instancia 1:

a. Borramos los datos de la tabla de prueba:

DELETE FROM pruebatransacciones

b. Insertamos un valor inicial original:

Insert dbo.pruebatransacciones(id,descripcion) values('1','valor original');

2. Nuevamente iniciamos una transacción, realizamos un update sobre el valor previamente cargado e insertamos un segundo valor:

begin transaction

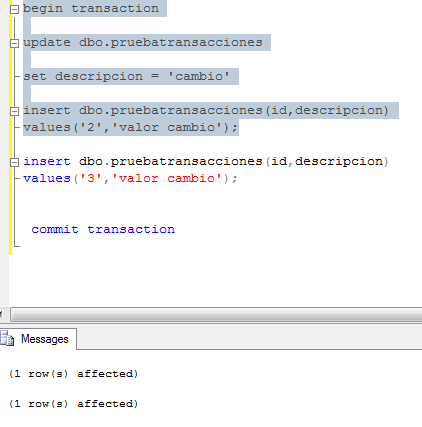
update dbo.pruebatransacciones

set descripcion = 'cambio'

insert dbo.pruebatransacciones(id,descripcion)

   values('2','valor cambio');

**nota: Estas instrucciones dejarán la transacción abierta a la espera del cierre** (commit transaction)

  
7- Ejecución de las primeras tres instrucciones.

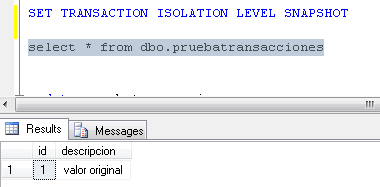
3. Luego, desde otra instancia de SQL SERVER (instancia 2):

a. Establecemos el nivel de aislamiento:

SET TRANSACTION ISOLATION LEVEL SNAPSHOT

b. Luego consultamos la tabla y obtenemos una foto de la tabla antes de empezar la transacción:

select \* from dbo.pruebatransacciones

  
8- La consulta arroja  el valor  original.

El valor arrojado corresponde al existente en la tabla antes que la consulta de la instancia uno lo modifique.

Además, si desde la instancia dos se desea una tupla específica, solo se podrá hacerlo en aquellas filas cuyo valor no haya sido modificado por la instancia uno.