In this video we will discuss the differences between snapshot isolation and read committed snapshot isolation in sql server. This is continuation to [Parts 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) and [76](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html). Please watch [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) and [76](http://csharp-video-tutorials.blogspot.com/2015/08/read-committed-snapshot-isolation-level.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
 

|  |  |
| --- | --- |
| **Read Committed Snapshot Isolation** | **Snapshot Isolation** |
| No update conflicts | Vulnerable to update conflicts |
| Works with existing applications without requiring any change to the application | Application change may be required to use with an existing application |
| Can be used with distributed transactions | Cannot be used with distributed transactions |
| Provides statement-level read consistency | Provides transaction-level read consistency |

**Update conflicts :** Snapshot isolation is vulnerable to update conflicts where as Read Committed Snapshot Isolation is not. When a transaction running under snapshot isolation triess to update data that an another transaction is already updating at the sametime, an update conflict occurs and the transaction terminates and rolls back with an error.   
  
**We will use the following table tblInventory in this demo**   
Read committed snapshot isolation level example   
  
Enable Snapshot Isolation for the SampleDB database using the following command

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed. When Transaction 1 completes, Transaction 2 raises an update conflict and the transaction terminates and rolls back with an error. 

--Transaction 1

Set transaction isolation level snapshot

Begin Transaction

Update tblInventory set ItemsInStock = 8 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level snapshot

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

Commit Transaction

Now let's try the same thing using **Read Committed Sanpshot Isolation**   
  
**Step 1 :** Disable Snapshot Isolation for the SampleDB database using the following command

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION OFF

**Step 2 :**Enable Read Committed Sanpshot Isolation at the database level using the following command

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT ON

**Step 3 :**Open 2 instances of SQL Server Management studio. From the first window execute Transaction 1 code and from the second window execute Transaction 2 code. Notice that Transaction 2 is blocked until Transaction 1 is completed. When Transaction 1 completes, Transaction 2 also completes successfully without any update conflict. 

--Transaction 1

Set transaction isolation level read committed

Begin Transaction

Update tblInventory set ItemsInStock = 8 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

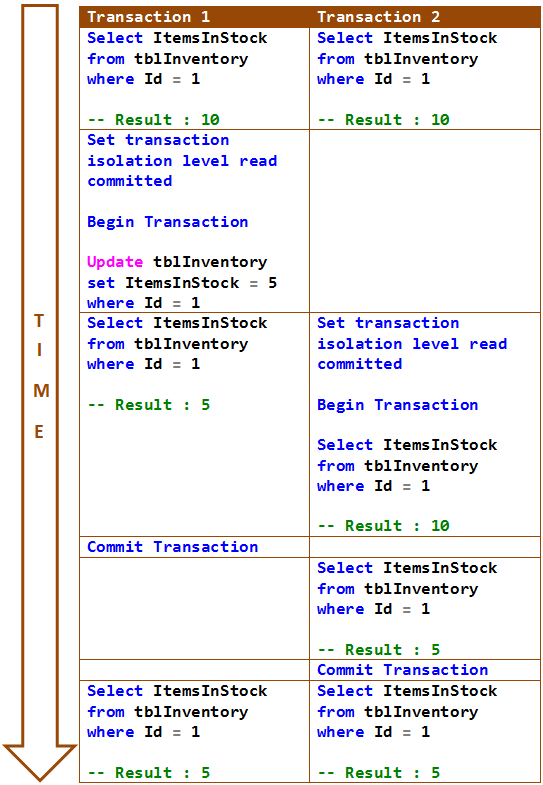
-- Transaction 2

Set transaction isolation level read committed

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

Commit Transaction

**Existing application :** If your application is using the default Read Committed isolation level, you can very easily make the application to use Read Committed Snapshot Isolation without requiring any change to the application at all. All you need to do is turn on READ\_COMMITTED\_SNAPSHOT option in the database, which will change read committed isolation to use row versioning when reading the committed data.  
  
**Distributed transactions :** Read Committed Snapshot Isolation works with distributed transactions, whereas snapshot isolation does not.  
  
**Read consistency :**Read Committed Snapshot Isolation provides statement-level read consistency where as Snapshot Isolation provides transaction-level read consistency. The following diagrams explain this.  
  
Transaction 2 has 2 select statements. Notice that both of these select statements return different data. This is because Read Committed Snapshot Isolation returns the last committed data before the select statement began and not the last committed data before the transaction began.   
   
  
In the following example, both the select statements of Transaction 2 return same data. This is because Snapshot Isolation returns the last committed data before the transaction began and not the last committed data before the select statement began.   
