In this video we will discuss **Read committed snapshot isolation level** in sql server. This is continuation [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html). Please watch [Part 75](http://csharp-video-tutorials.blogspot.com/2015/08/snapshot-isolation-level-in-sql-server.html) from [SQL Server tutorial](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB) before proceeding.   
  
   
  
**We will use the following table tblInventory in this demo**   
Read committed snapshot isolation level example   
  
Read committed snapshot isolation level is not a different isolation level. It is a different way of implementing Read committed isolation level. One problem we have with Read Committed isloation level is that, it blocks the transaction if it is trying to read the data, that another transaction is updating at the same time.   
  
The following example demonstrates the above point. 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.  

--Transaction 1

Set transaction isolation level Read Committed

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level read committed

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

We can make Transaction 2 to use row versioning technique instead of locks by enabling Read committed snapshot isolation at the database level. Use the following command to enable READ\_COMMITTED\_SNAPSHOT isolation

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT ON

**Please note :** For the above statement to execute successfully all the other database connections should be closed.   
  
After enabling READ\_COMMITTED\_SNAPSHOT, execute Transaction 1 first and then Transaction 2 simultaneously. Notice that the Transaction 2 is not blocked. It immediately returns the committed data that is in the database before Transaction 1 started. This is because Transaction 2 is now using Read committed snapshot isolation level.   
  
Let's see if we can achieve the same thing using snapshot isolation level instead of read committed snapshot isolation level.    
  
**Step 1 :** Turn off READ\_COMMITTED\_SNAPSHOT

Alter database SampleDB SET READ\_COMMITTED\_SNAPSHOT OFF

**Step 2 :** Enable snapshot isolation level at the database level

Alter database SampleDB SET ALLOW\_SNAPSHOT\_ISOLATION ON

**Step 3 :**Execute Transaction 1 first and then Transaction 2 simultaneously. Just like in the previous example, notice that the Transaction 2 is not blocked. It immediately returns the committed data that is in the database before Transaction 1 started.

--Transaction 1

Set transaction isolation level Read Committed

Begin Transaction

Update tblInventory set ItemsInStock = 5 where Id = 1

waitfor delay '00:00:10'

Commit Transaction

-- Transaction 2

Set transaction isolation level snapshot

Begin Transaction

Select ItemsInStock from tblInventory where Id = 1

Commit Transaction

**So what is the point in using read committed snapshot isolation level over snapshot isolation level?**  
There are some differences between read committed snapshot isolation level and snapshot isolation level. We will discuss these in our next video.