In this video we will discuss, **dirty read concurrency problem** with an example. This is continuation to [Part 70](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html). Please watch [Part 70](http://csharp-video-tutorials.blogspot.com/2015/08/sql-server-concurrent-transactions.html) from [SQL Server tutorial for beginners](https://www.youtube.com/playlist?list=PL08903FB7ACA1C2FB).   
  
   
  
A dirty read happens when one transaction is permitted to read data that has been modified by another transaction that has not yet been committed. In most cases this would not cause a problem. However, if the first transaction is rolled back after the second reads the data, the second transaction has dirty data that does not exist anymore.   
  
SQL script to create table tblInventory 

Create table tblInventory

(

    Id int identity primary key,

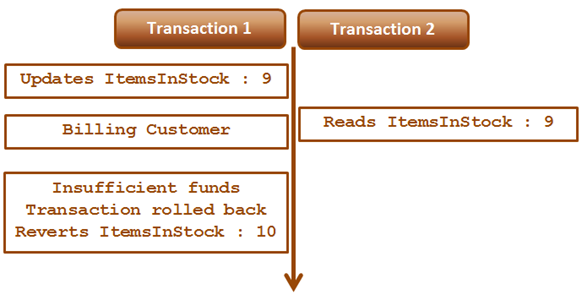
    Product nvarchar(100),

    ItemsInStock int

)

Go

Insert into tblInventory values ('iPhone', 10)

**Table tblInventory**   
dirty read problem in concurrency control   
  
**Dirty Read Example :** In the example below, Transaction 1, updates the value of ItemsInStock to 9. Then it starts to bill the customer. While Transaction 1 is still in progress, Transaction 2 starts and reads ItemsInStock value which is 9 at the moment. At this point, Transaction 1 fails because of insufficient funds and is rolled back. The ItemsInStock is reverted to the original value of 10, but Transaction 2 is working with a different value (i.e 10).   
  
  
  
**Transaction 1 :**  
Begin Tran

Update tblInventory set ItemsInStock = 9 where Id=1

-- Billing the customer

Waitfor Delay '00:00:15'

-- Insufficient Funds. Rollback transaction

Rollback Transaction

**Transaction 2 :**

Set Transaction Isolation Level Read Uncommitted

Select \* from tblInventory where Id=1

Read Uncommitted transaction isolation level is the only isolation level that has dirty read side effect. This is the least restrictive of all the isolation levels. When this transaction isolation level is set, it is possible to read uncommitted or dirty data. Another option to read dirty data is by using NOLOCK table hint. The query below is equivalent to the query in Transaction 2. 

Select \* from tblInventory (NOLOCK) where Id=1