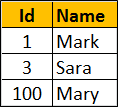
In this video we will discuss **phantom read concurrency problem** with examples.   
  
   
  
Phantom read happens when one transaction executes a query twice and it gets a different number of rows in the result set each time. This happens when a second transaction inserts a new row that matches the WHERE clause of the query executed by the first transaction.    
  
We will use the following table tblEmployees in this demo  
   
  
**Scrip to create the table tblEmployees**

Create table tblEmployees

(

    Id int primary key,

    Name nvarchar(50)

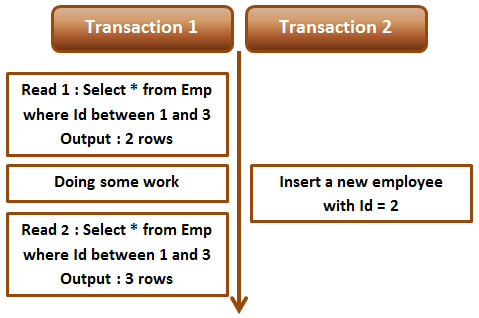
)

Go

Insert into tblEmployees values(1,'Mark')

Insert into tblEmployees values(3, 'Sara')

Insert into tblEmployees values(100, 'Mary')

**The following diagram explains the problem :** Transaction 1 starts first. Reads from Emp table where Id between 1 and 3. 2 rows retrieved for first read. Transaction 1 is doing some work and at this point Transaction 2 starts and inserts a new employee with Id = 2. Transaction 1 then makes a second read. 3 rows retrieved for second read, reulting in phantom read problem.   
  
   
  
**Phantom read example :** 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 when Transaction 1 completes, it gets different number of rows for read 1 and read 2, resulting in phantom read. 

-- Transaction 1

Begin Transaction

Select \* from tblEmployees where Id between 1 and 3

-- Do Some work

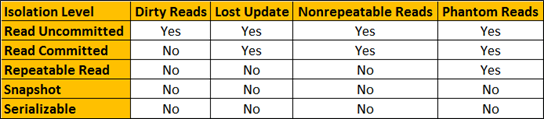
waitfor delay '00:00:10'

Select \* from tblEmployees where Id between 1 and 3

Commit Transaction

-- Transaction 2

Insert into tblEmployees values(2, 'Marcus')

Serializable or any other higher isolation level should solve the phantom read problem.  
   
  
**Fixing phantom read concurrency problem :** To fix the phantom read problem, set transaction isolation level of Transaction 1 to serializable. This will place a range lock on the rows between 1 and 3, which prevents any other transaction from inserting new rows with in that range. This solves the phantom read problem.    
  
When you execute Transaction 1 and 2 from 2 different instances of SQL Server management studio, Transaction 2 is blocked until Transaction 1 completes and at the end of Transaction 1, both the reads get the same number of rows. 

-- Transaction 1

Set transaction isolation level serializable

Begin Transaction

Select \* from tblEmployees where Id between 1 and 3

-- Do Some work

waitfor delay '00:00:10'

Select \* from tblEmployees where Id between 1 and 3

Commit Transaction

-- Transaction 2

Insert into tblEmployees values(2, 'Marcus')

**Difference between repeatable read and serializable**  
**Repeatable read prevents only non-repeatable read.** Repeatable read isolation level ensures that the data that one transaction has read, will be prevented from being updated or deleted by any other transaction, but it doe not prevent new rows from being inserted by other transactions resulting in phantom read concurrency problem.   
  
**Serializable prevents both non-repeatable read and phantom read problems.**Serializable isolation level ensures that the data that one transaction has read, will be prevented from being updated or deleted by any other transaction. It also prevents new rows from being inserted by other transactions, so this isolation level prevents both non-repeatable read and phantom read problems.