Concurrency

ISM 6218

Due on November 12th

The Avengers Team

"We will avenge every problem on our way"

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

1. Requirements_____3
2. Business Process Explained_____4
3. User Story_____5
4. Isolation Levels_____6
5. Hints_____10
6. Locks_____13
7. DM_EXEC_QUERIES_____15

Requirements

This series of experiments must cover the following areas:

1. Serializable Transactions

2. Isolation Levels.

3. Hints: no lock, read past, no wait

4. Shared locks, exclusive locks, deadlock resolution.

5. DM_Exec queries.

Must include the following:

- All 4 isolation levels.
- Compare with row versioning, snapshot isolation (SI).
- 3 locking options.
- 4 table hints.
- Must support a user story for each isolation level and locking option choice explaining what advantage the choice provides the user.
- You must demonstrate effective use of Dynamic Management View evaluating the waits and blocks.

Business Process Supported

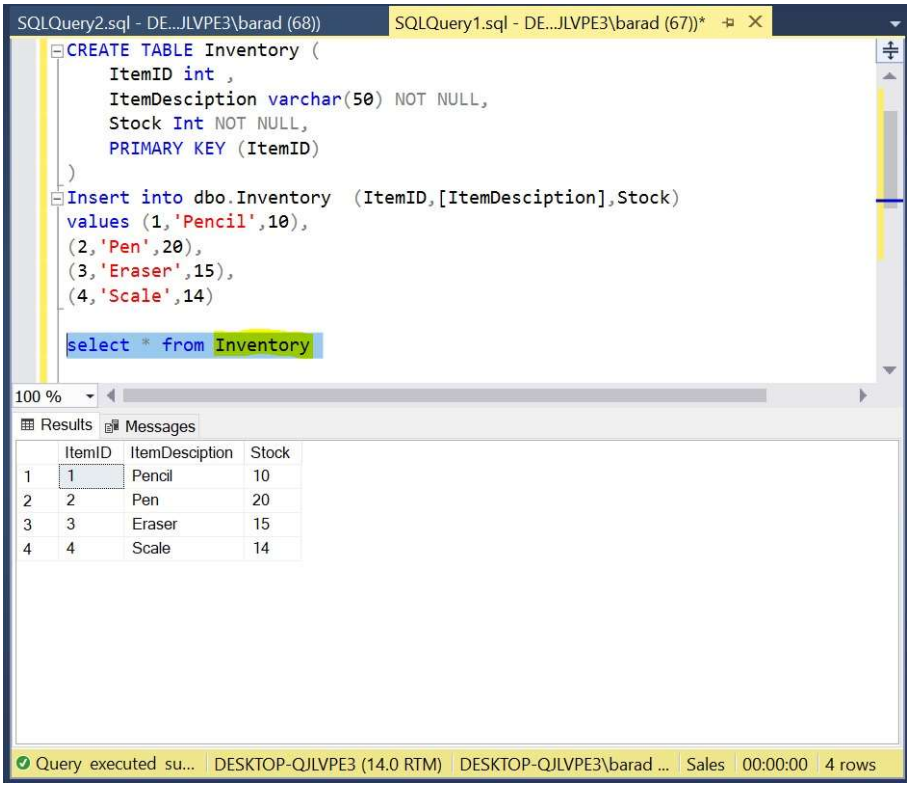
We have created the Inventory Table which consists of the Item Id, Item description and it stock at a point of time.

USER STORY:

Store manager wants to maintain a Database of stocks which is administered by more than one Database administrators from different locations without facing any Concurrency Issues.

Create a table and add rows.

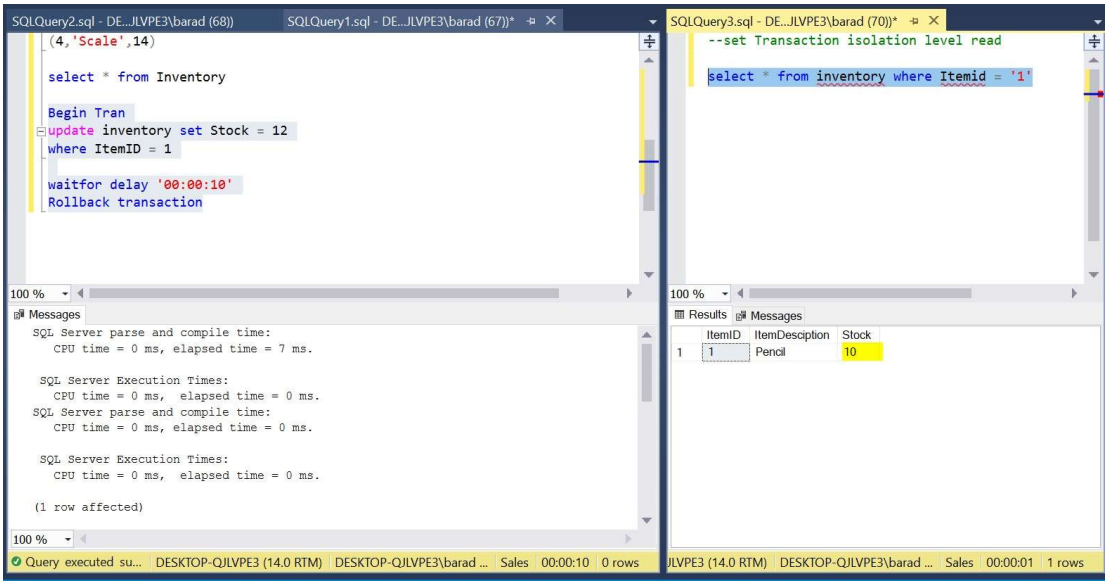
We have created a Inventory Table which consists of Item Id, Item Description, Stock and inserted some dummy records into the table.



Isolation levels

- **Read Committed**

  This isolation level guarantees that any data read is committed at the moment it is read. Thus, it does not allow dirty read. The transaction holds a read or write lock on the current row, and thus prevent other transactions from reading, updating or deleting it.

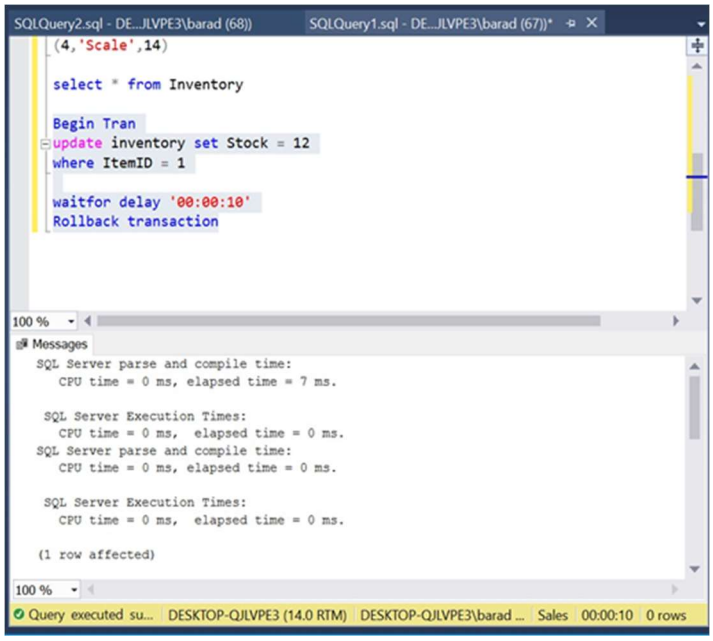


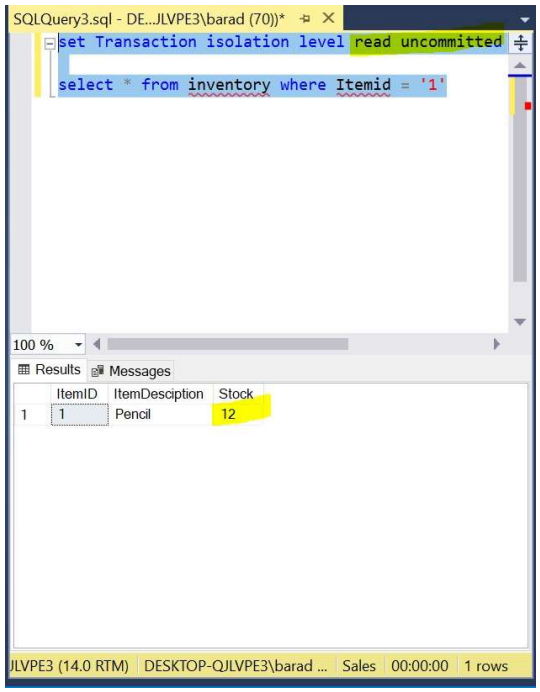
  In the above screenshot we can see the stock of the product pencil is updated to '12' in query window1 with a delay time and in that delay time query in the window2 is run. Query in window2 will wait until the query in window 1 is done. So it didn't allow dirty read as '12'.

- **Read Uncommitted**

  Read Uncommitted is the lowest isolation level. In this level, one transaction may read not yet committed changes made by other transaction, thereby allowing dirty reads. In this level, transactions are not isolated from each other.

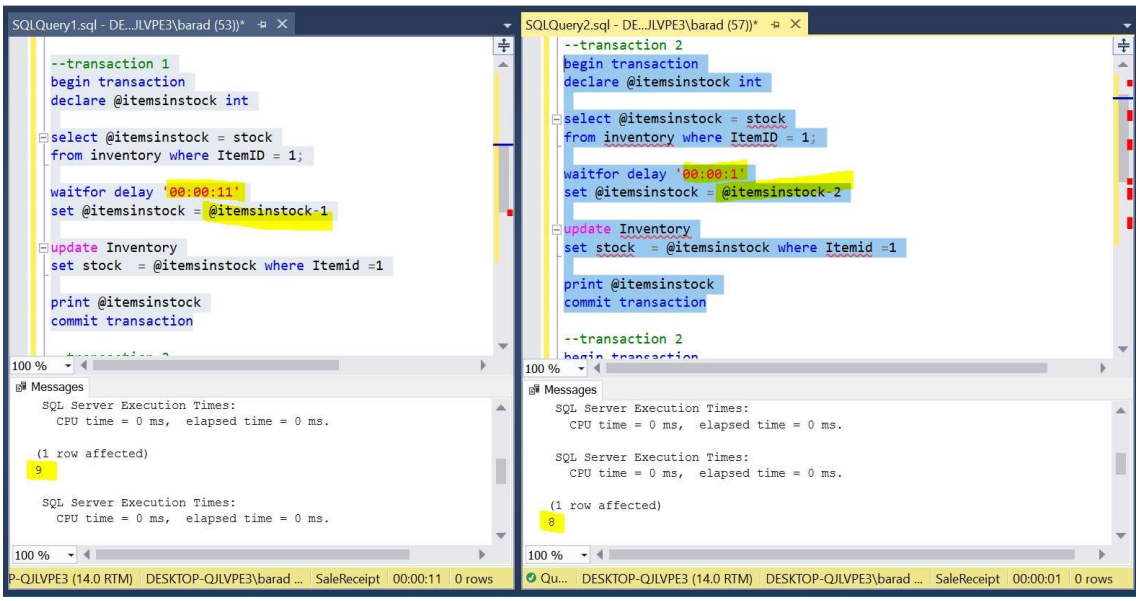
  In the below screenshot we can see the stock of the product pencil is updated to '12' in query window1 with a delay time and in that delay time query in the window2 is run. Query in window2 will not wait until the query in window 1 is done and outputs dirty data.

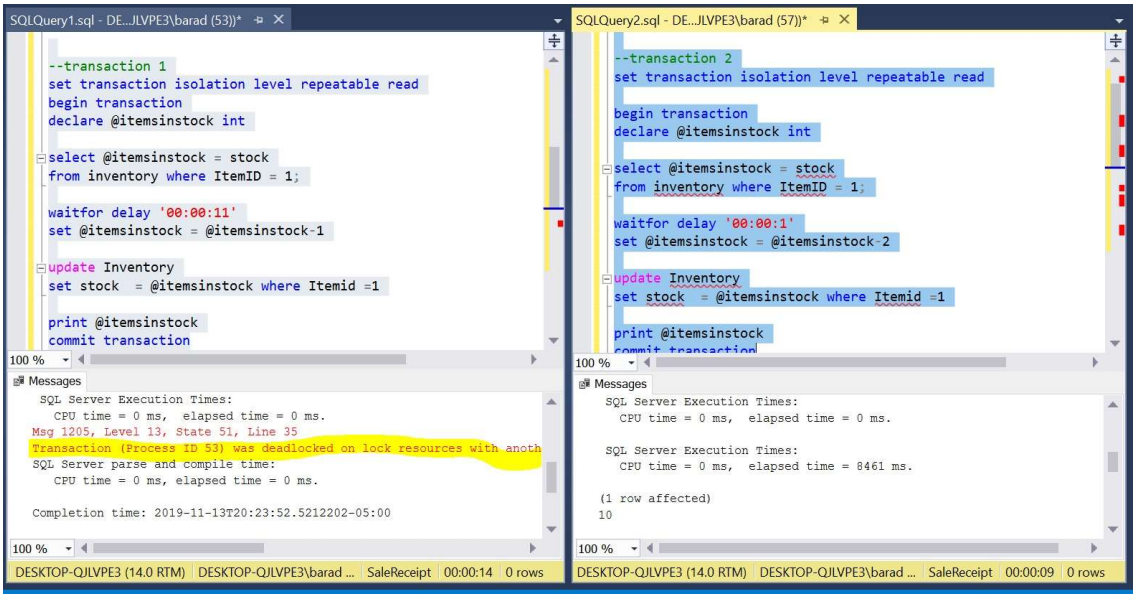




- **Repeatable Read**

  This is the most restrictive isolation level. The transaction holds read locks on all rows it references and writes locks on all rows it inserts, updates, or deletes. Since other transaction cannot read, update or delete these rows, consequently it avoids non-repeatable read.

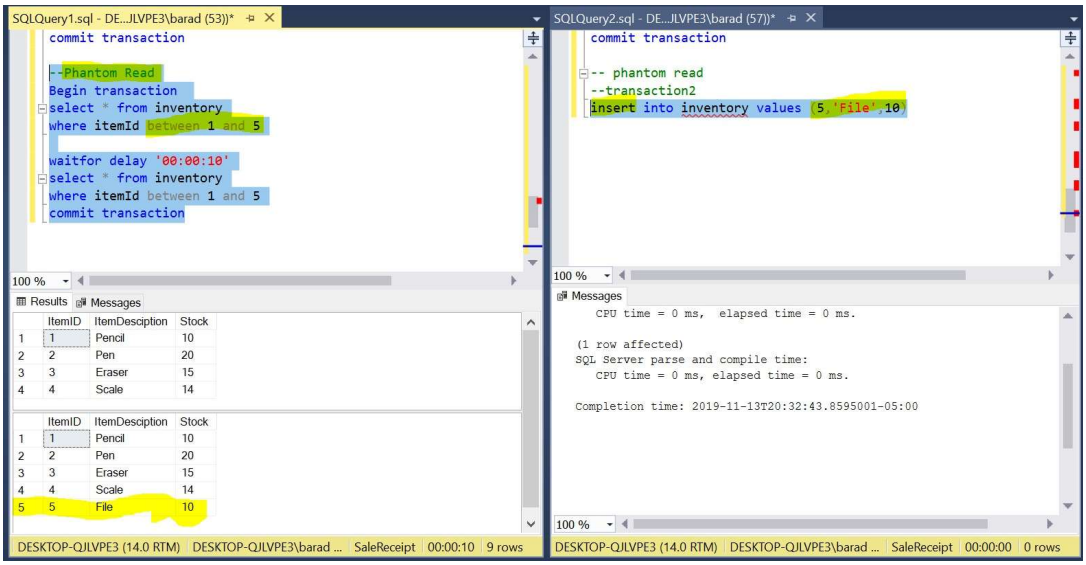


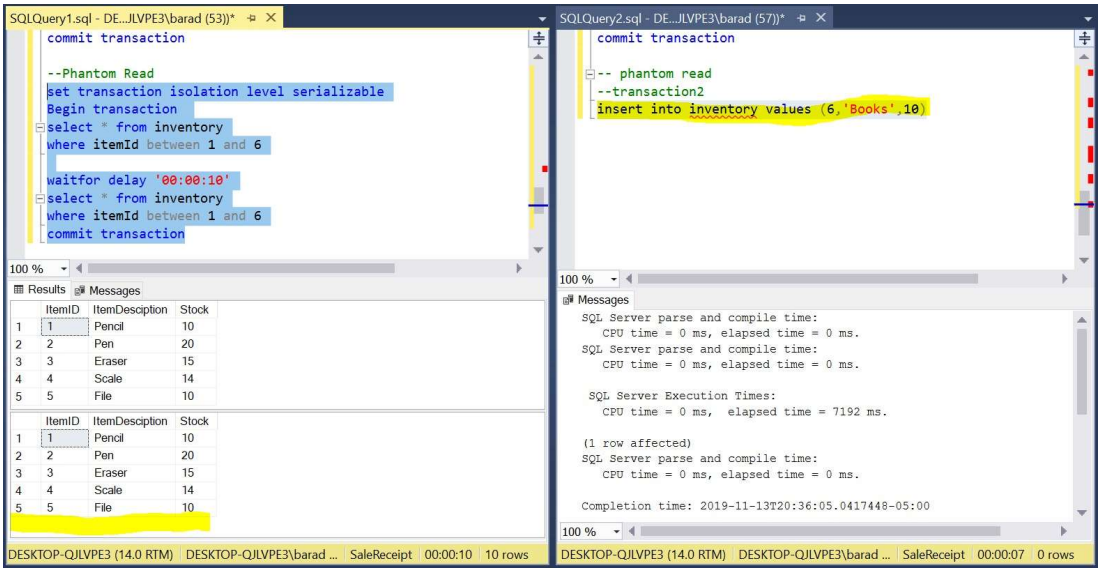


In the above screenshot1 we can see the stock of the product pencil is reduced by 1 from 10 to 9 in query window1 with a delay time and in that delay time query in the window2 which will reduce the product by 2 from 10 to 8 is run. Query in window2 will return 8 and query in window 1 will return 8 and produces data loss. We can avoid this by introducing isolation level to Repeatable data which will throw an error “Error: Transaction (Process ID 53) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.” Refer screenshot 2.

- **Serializable**

  This is the Highest isolation level. A serializable execution is guaranteed to be serializable. Serializable execution is defined to be an execution of operations in which concurrently executing transactions appears to be serially executing.





In the above screenshot1 we can see the new product file is added into table at the same time of delay between two queries of reading the table in window 1, which will return the updated table. Ideally it should not allow insertion when the query 2 is in progress, which is also called Phantom error. To avoid this set isolation level to serializable, refer screenshot2, which avoided insertion of new product book into table.

Hints

- **No Lock**
  Items in the Inventory table.

```
select * from Inventory
```

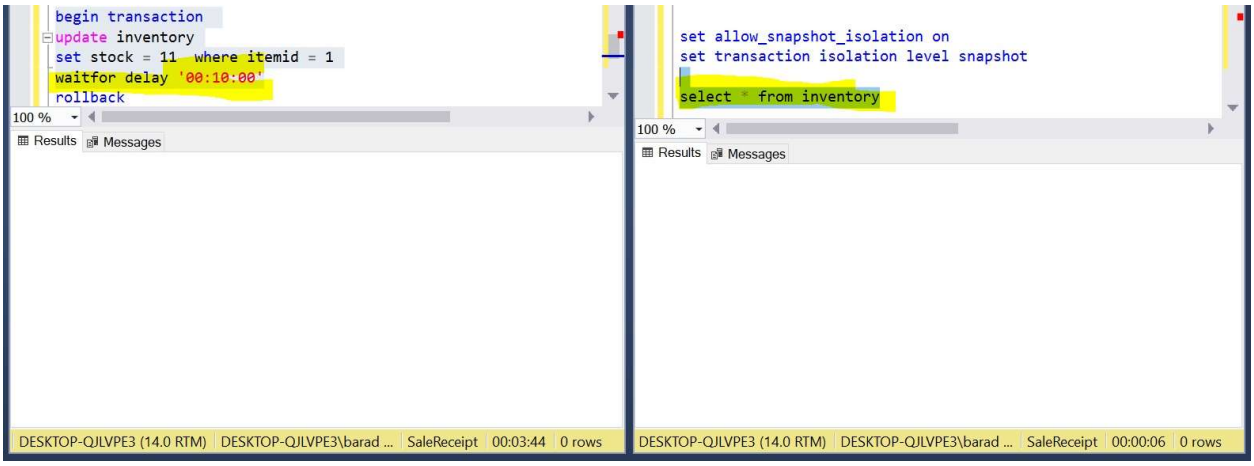
100 %

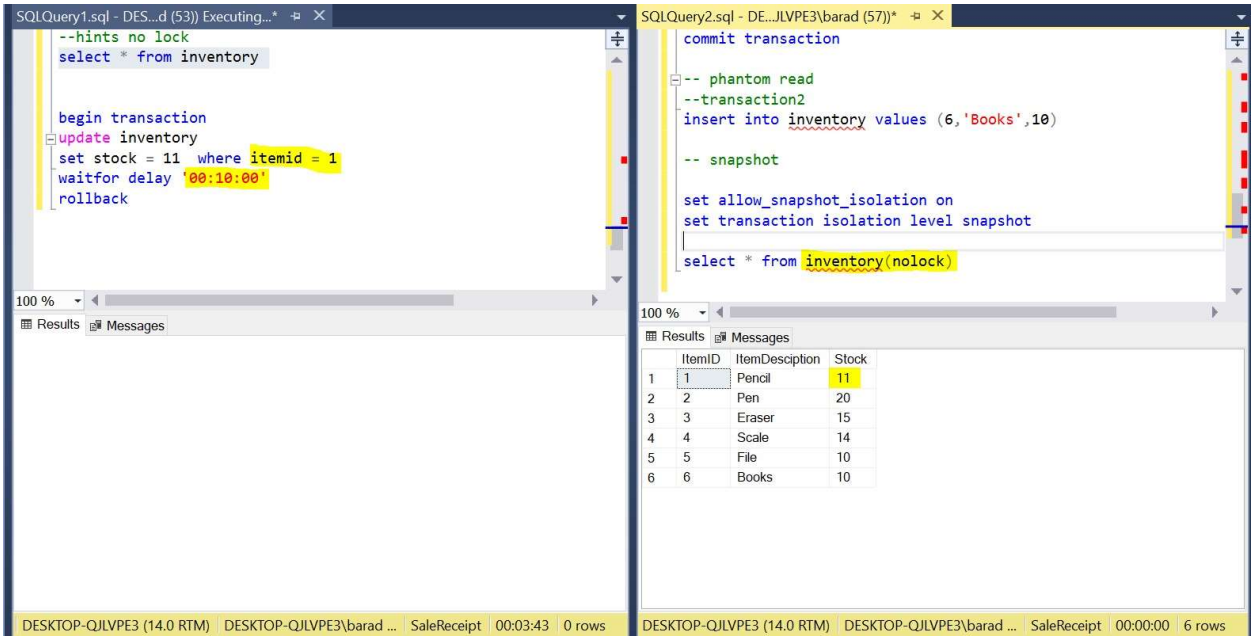
Results | Messages

	ItemID	ItemDesciption	Stock
1	1	Pencil	10
2	2	Pen	20
3	3	Eraser	15
4	4	Scale	14
5	5	File	10
6	6	Books	10

DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:00 | 6 rows

In the below screenshot we can see without no lock, we cannot read table query, whereas with using no lock (refer screenshot2) the list gets updated even before the completion of transaction giving dirty reads.





- **Read past**

  In the below screenshot without using anything, we cannot execute query, whereas with Read past we can access the table except one which we are updating.

```
begin transaction
update inventory
set stock = 11  where itemid = 1
waitfor delay '00:10:00'
rollback
```

```
set allow_snapshot_isolation on
set transaction isolation level snapshot

select * from inventory
```

```
begin transaction
update inventory
set stock = 11  where itemid = 1
waitfor delay '00:00:15'
rollback
```

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)

 SQL Server Execution Times:
```

```
select * from inventory

select * from inventory(readpast)
```

	ItemID	ItemDesciption	Stock
1	2	Pen	20
2	3	Eraser	15
3	4	Scale	14
4	5	File	10
5	6	Books	10

- **No wait**

In the below screenshot with use of no wait, SQL server throws error and we'll be not able to access the table.

```
begin transaction
update inventory
set stock = 11  where itemid = 1
waitfor delay '00:00:15'
rollback
```

```
Messages
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)

 SQL Server Execution Times:
```

DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:15 | 0 rows

```
select * from inventory(readpast)

select * from inventory(nowait)
```

```
Results | Messages
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Msg 1222, Level 16, State 51, Line 52
Lock request time out period exceeded.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-11-13T22:47:33.0907245-05:00
```

DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:00 | 0 rows

Locks

- **Shared Lock**

  Shared (S) locks allow concurrent transactions to read (SELECT) a resource under pessimistic concurrency control. No other transactions can modify the data while shared (S) locks exist on the resource. Shared (S) locks on a resource are released as soon as the read operation completes, unless the transaction isolation level is set to repeatable read or higher, or a locking hint is used to retain the shared (S) locks for the duration of the

Exclusive (X) locks prevent access to a resource by concurrent transactions. With an exclusive (X) lock, no other transactions can modify data; read operations can take place only with the use of the NOLOCK hint or read uncommitted isolation level.

```
set Transaction isolation level read committed
begin tran
select * from Inventory with(HOLDLOCK)

SELECT resource_type, request_mode, resource_description
FROM   sys.dm_tran_locks
WHERE  resource_type <> 'DATABASE'
rollback
```

100 %

Results | Messages

	ItemID	ItemDesciption	Stock
1	1	Pencil	10
2	2	Pen	20
3	3	Eraser	15
4	4	Scale	14
5	5	File	10
6	6	Books	10

	resource_type	request_mode	resource_description
1	PAGE	IS	1:392
2	KEY	RangeS-S	(ffffffffffff)
3	KEY	RangeS-S	(98ec012aa510)
4	KEY	RangeS-S	(a0c936a3c965)
5	OBJECT	IS	

DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:00 | 15 rows

- **Exclusive Lock**

  Exclusive (X) locks prevent access to a resource by concurrent transactions. With an exclusive (X) lock, no other transactions can modify data; read operations can take place only with the use of the NOLOCK hint or read uncommitted isolation level.

```
begin tran
select * from Inventory with(XLOCK)

SELECT resource_type, request_mode, resource_description
FROM   sys.dm_tran_locks
WHERE  resource_type <> 'DATABASE'
rollback
```

100 %

Results | Messages

	ItemID	ItemDesciption	Stock
1	1	Pencil	15
2	2	Pen	20
3	3	Eraser	15
4	4	Scale	14
5	5	File	10
6	6	Books	10

	resource_type	request_mode	resource_description
1	PAGE	IX	1:392
2	KEY	X	(98ec012aa510)
3	KEY	X	(a0c936a3c965)
4	OBJECT	IX	
5	KEY	X	(8194443284a0)

DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:00 | 14 rows

- **DeadLock**

  A deadlock occurs when 2 processes are competing for exclusive access to a resource but is unable to obtain exclusive access to it, because the other process is preventing it. This results in a standoff where neither process can proceed.

```
-- Transaction 1
Begin Tran
Update Inventory set Stock = '1' where itemid = 1

-- From Transaction 2 window execute the first up
waitfor delay '00:00:15'
Update Inventory set Stock = '2' where itemid = 2

-- From Transaction 2 window execute the second u
Commit Transaction
```

100 %

Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)
```

100 %

3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:26 | 0 rows

```
-- Transaction 2
Begin Tran
Update Inventory set Stock = '1' where itemid = 2
waitfor delay '00:00:15'

-- From Transaction 1 window execute the second update statement

Update Inventory set Stock = '2' where itemid = 1

-- After a few seconds notice that one of the transactions complete
-- successfully while the other transaction is made the deadlock victim

Commit Transaction
```

100 %

Messages

```
   CPU time = 0 ms,  elapsed time = 15002 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Msg 1205, Level 13, State 51, Line 104
Transaction (Process ID 52) was deadlocked on lock resources with another process and
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-11-13T23:59:10.9234288-05:00
```

100 %

Query com... | DESKTOP-QJLVPE3 (14.0 RTM) | DESKTOP-QJLVPE3\barad ... | SaleReceipt | 00:00:16 | 0 rows