Database Design & Applications

Concurrency Control

Objectives

- Transaction Properties
- What is Concurrency
- Concurrency Problems:
    - Dirty Reads
    - Lost Updates
    - Non-repeatable Reads
    - Phantom Reads
- Isolation Levels
- Setting Isolation using TSQL

What is a Transaction?

- A transaction is a unit of program execution that accesses and possibly updates various data items.
- A transaction must see a consistent database.
- During transaction execution the database may be inconsistent.
- When the transaction is committed, the database must be consistent.
- Two main issues to deal with:
    - Failures of various kinds, such as hardware failures and system crashes
    - Concurrent execution of multiple transactions

ACID Properties

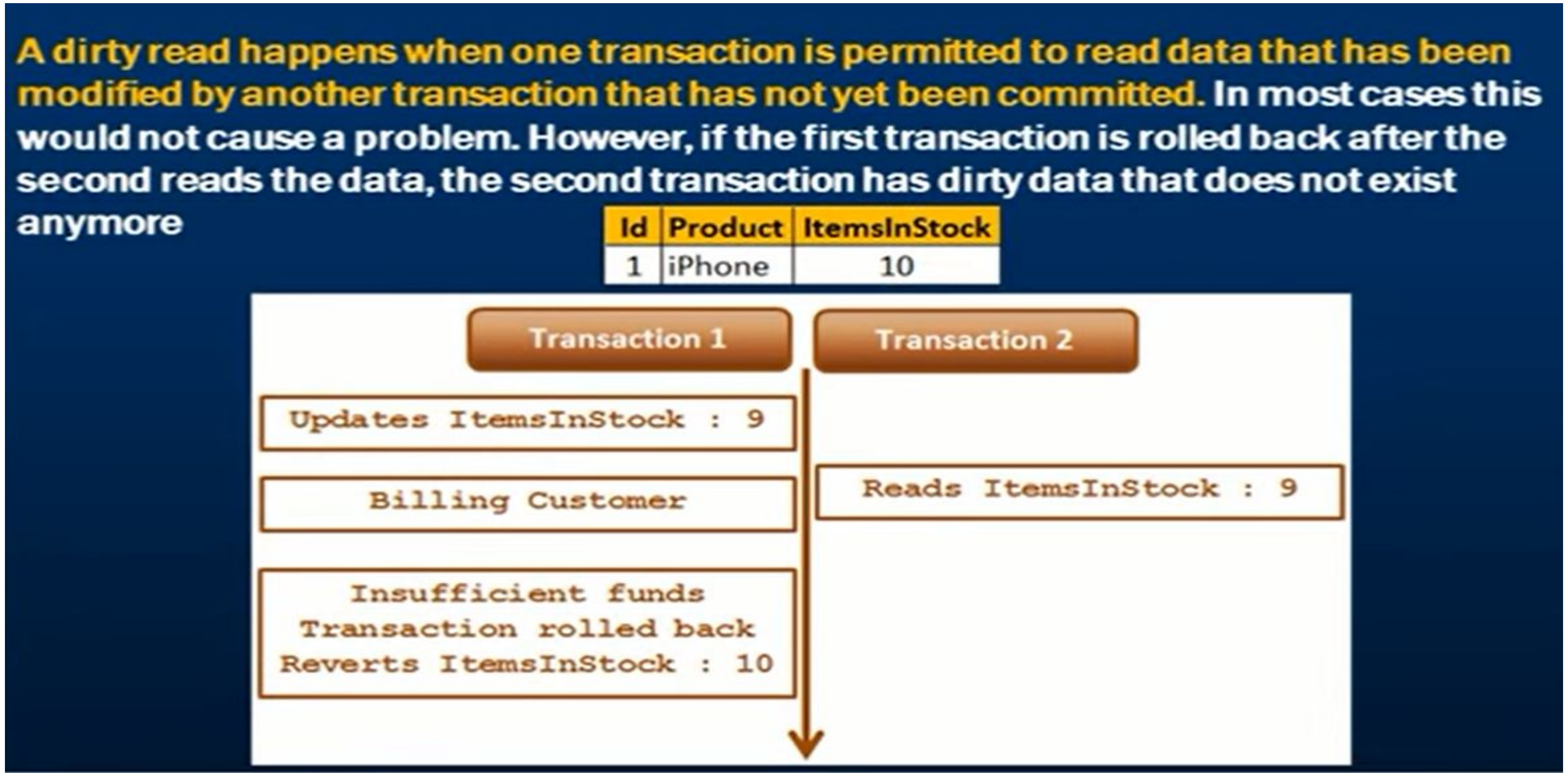
To preserve integrity of data, the database system must ensure:

- **Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency:** Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
    - That is, for every pair of transactions T_i and T_j, it appears to T_i that either T_j, finished execution before T_i started, or T_j started execution after T_i finished.
- **Durability:** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

Concurrency in SQL Server

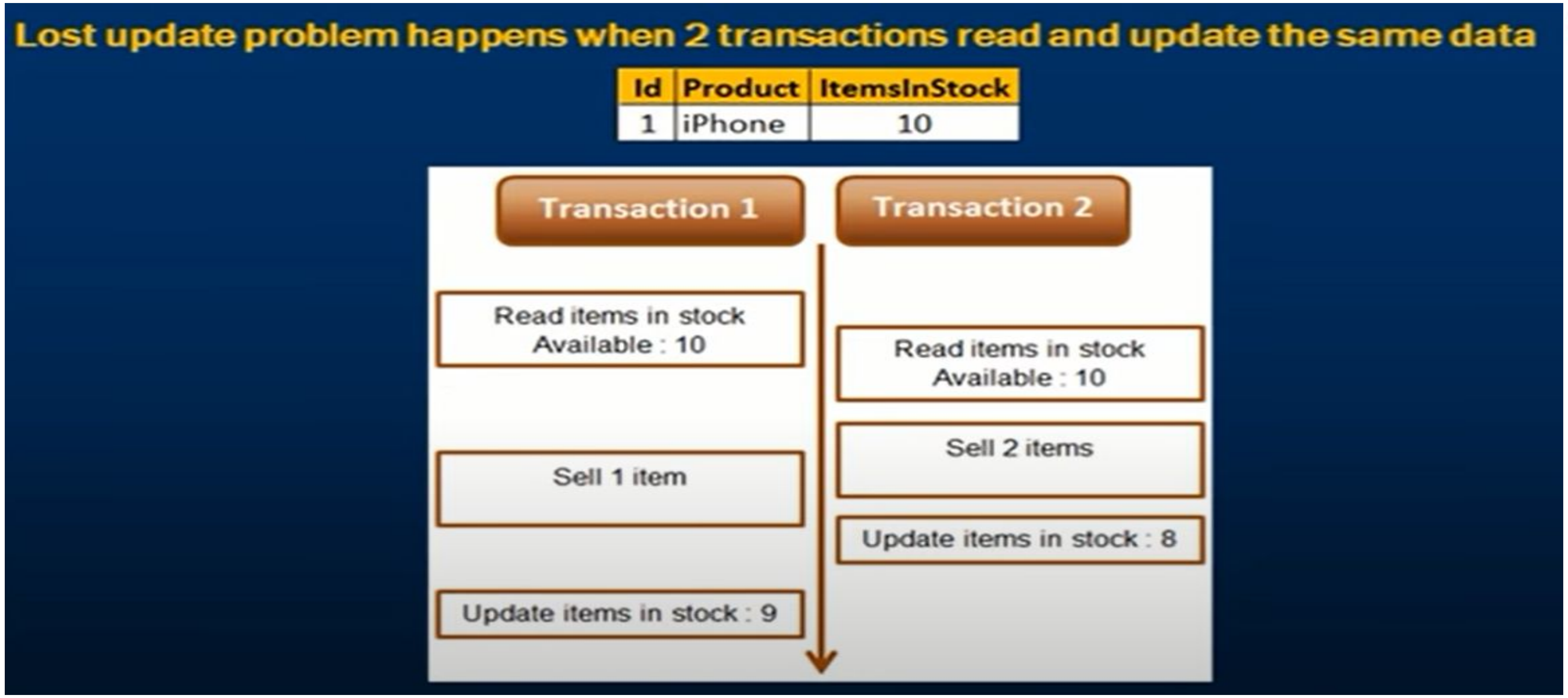
- Concurrency is a situation that arises in a database due to the transaction process.
- Concurrency occurs when two or more than two users are trying to access the same data or information.
- The concurrency problem mostly arises when both the users try to write the same data, or when one is writing and the other is reading
- **Concurrency Problem Types:**
    - Dirty Reads
    - Lost Updates
    - Non-repeatable Reads
    - Phantom Reads

Dirty Reads


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

Id	Product	ItemsInStock
1	iPhone	10

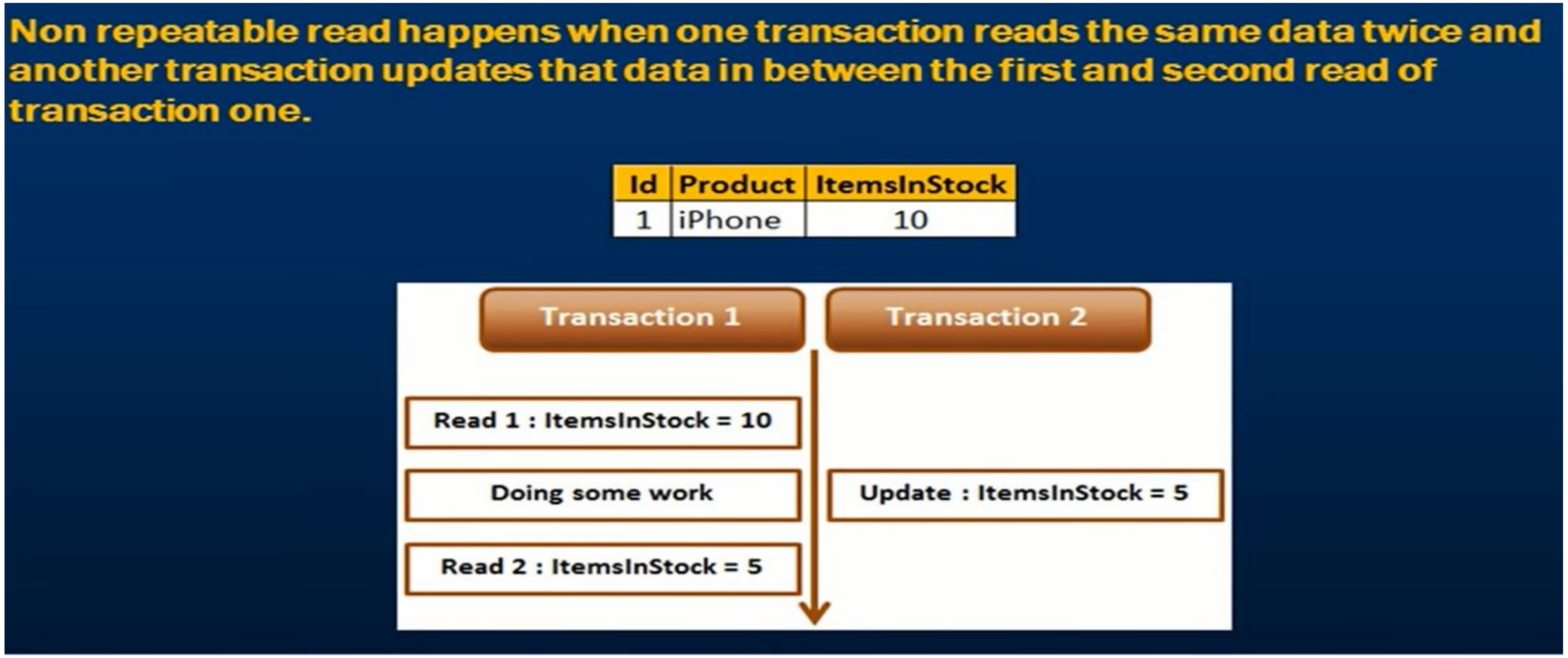
Lost Updates


Lost update problem happens when 2 transactions read and update the same data

Id	Product	ItemsInStock
1	iPhone	10

Transaction 1
Transaction 2
Read items in stock Available : 10
Read items in stock Available : 10
Sell 1 item
Sell 2 items
Update items in stock : 8
Update items in stock : 9

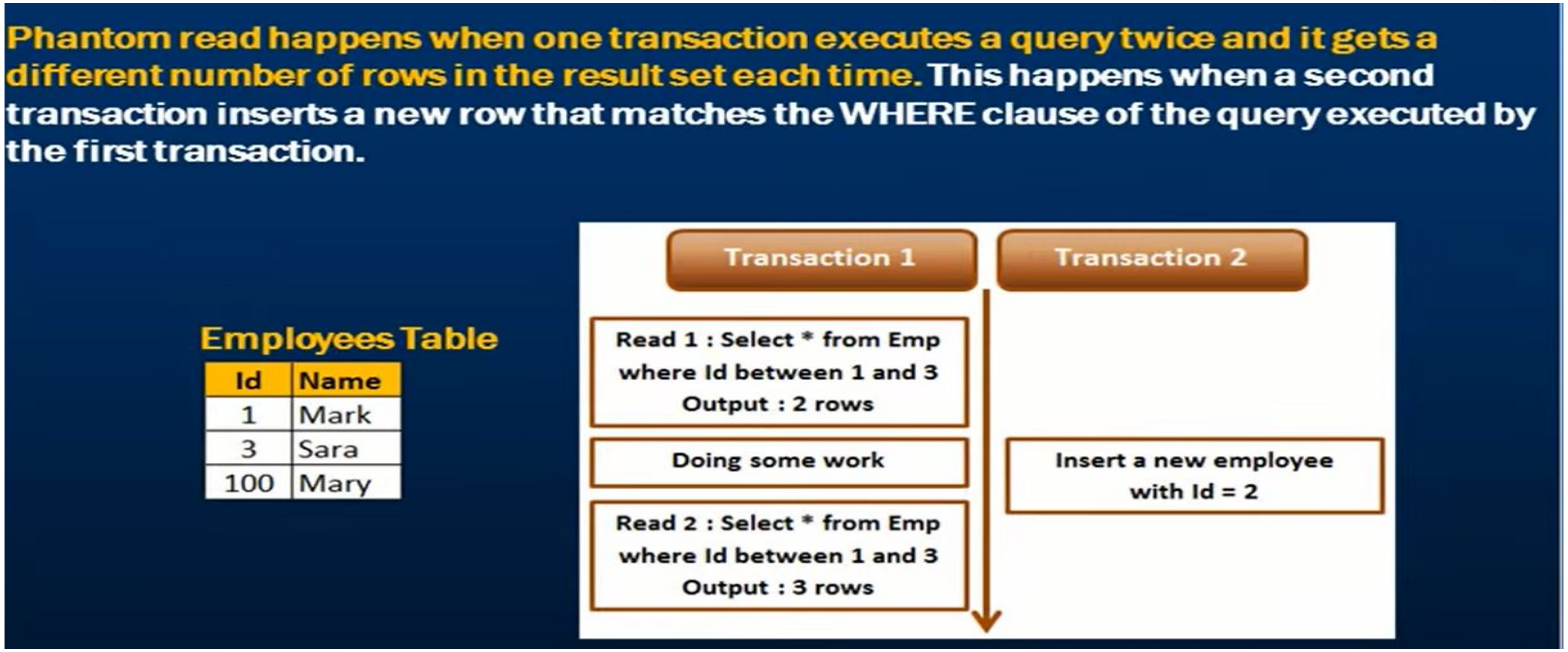
Non-repeatable Read


Non repeatable read happens when one transaction reads the same data twice and another transaction updates that data in between the first and second read of transaction one.

Id	Product	ItemsInStock
1	iPhone	10

Transaction 1
Transaction 2
Read 1 : ItemsInStock = 10
Doing some work
Update : ItemsInStock = 5
Read 2 : ItemsInStock = 5

Phantom Reads



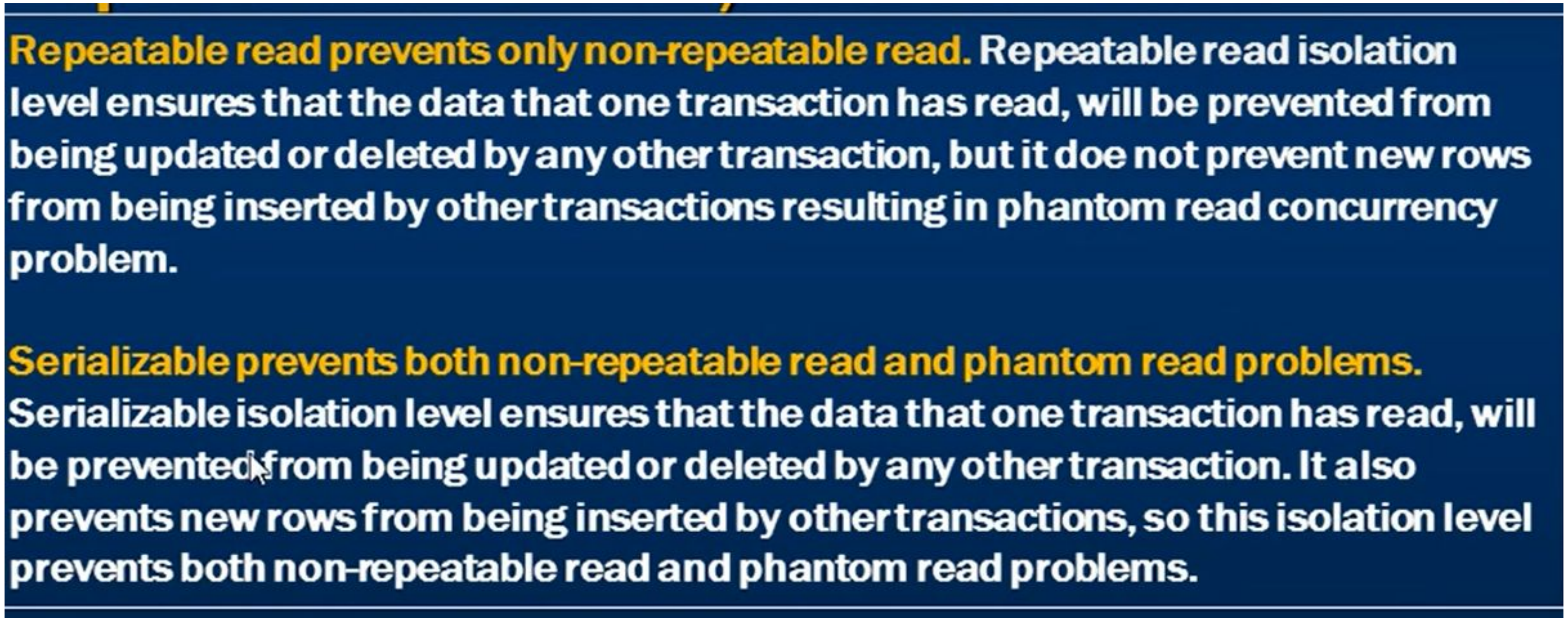
Phantom read happens when one transaction executes a query twice and it gets a different number of rows in the result set each time. This happens when a second transaction inserts a new row that matches the WHERE clause of the query executed by the first transaction.

Employees Table

Id	Name
1	Mark
3	Sara
100	Mary

Transaction 1
Transaction 2
Read 1 : Select * from Emp where Id between 1 and 3 Output : 2 rows
Doing some work
Insert a new employee with Id = 2
Read 2 : Select * from Emp where Id between 1 and 3 Output : 3 rows

Repeatable Read v/s 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.

Solve Concurrency Problems

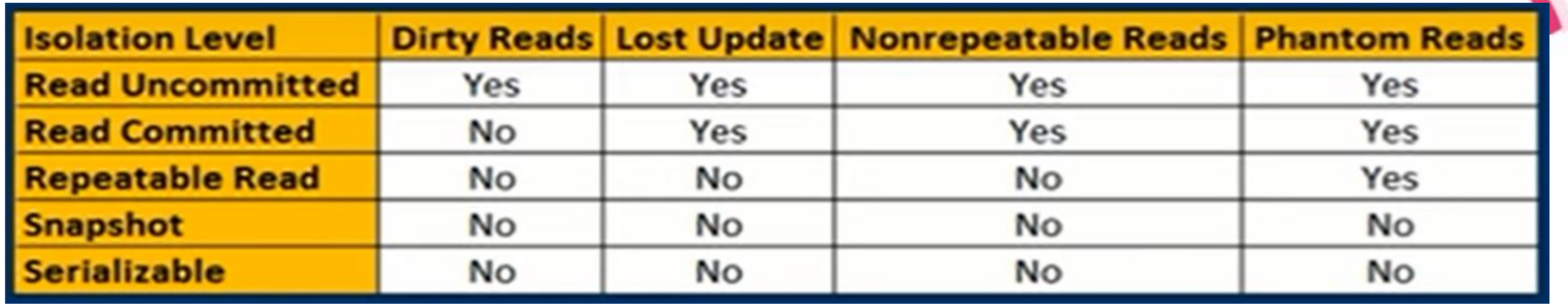
- SQL Server provides 5 different levels of transaction isolation to overcome these Concurrency problems. These 5 isolation levels work on two major concurrency models:
- **Pessimistic model** - In the pessimistic model of managing concurrent data access, the readers can block writers, and the writers can block readers.
- **Optimistic model** - In the optimistic model of managing concurrent data access, the readers cannot block writers, and the writers cannot block readers, but the writer can block another writer.

Note that readers are users are performing the SELECT operations. Writers are users are performing INSERT, ALTER, UPDATE, S.E.T. operations.

Isolation Levels

- When we connect to a SQL server database, the application can submit queries to the database with one of five different isolation levels. These levels are:
  1. Read Uncommitted
  2. Read Committed
  3. Repeatable Read
  4. Serializable
  5. Snapshot
- Out of these five isolation levels, 1-4 come under the pessimistic concurrency model. 5th level i.e. Snapshot comes under the optimistic concurrency model.

Isolation Levels



Isolation Level	Dirty Reads	Lost Update	Nonrepeatable Reads	Phantom Reads
Read Uncommitted	Yes	Yes	Yes	Yes
Read Committed	No	Yes	Yes	Yes
Repeatable Read	No	No	No	Yes
Snapshot	No	No	No	No
Serializable	No	No	No	No

- **Read Uncommitted**
  - This is the first level of isolation.
  - In Read Uncommitted, one transaction is allowed to read the data that is about to be changed by the commit of another process.
  - Read Uncommitted allows the dirty read problem.
- **Read Committed**
  - This is the second level of isolation.
  - In the Read Committed isolation level, we are only allowed to read data that is committed, which means this level eliminates the dirty read problem.
  - In this level, if you are reading data then the concurrent transactions that can delete or write data, some work is blocked until other work is complete.

Isolation Levels

- **Repeatable Read**
  - It eliminates the Non-Repeatable Read problem.
  - In this level, the transaction has to wait till another transaction's update or read query is complete. But if there is an insert transaction, it does not wait for anyone. This can lead to the Phantom Read problem.
- **Serializable**
  - This is the highest level of isolation in the pessimistic model.
  - By implementing this level of isolation, we can prevent the Phantom Read problem.
  - In this level of isolation, we can ask any transaction to wait until the current transaction completes.
- **Snapshot**
  - Snapshot follows the optimistic model of concurrency
  - In this level of isolation takes a snapshot of the current data and uses it as a copy for the different transactions.
  - Here each transaction has its copy of data, so if a user tries to perform a transaction like an update or insert, it asks him to re-verify all the operation before the process gets started executing.

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)

- **Syntax:**

```
SET TRANSACTION ISOLATION LEVEL
  { READ UNCOMMITTED
  | READ COMMITTED
  | REPEATABLE READ
  | SNAPSHOT
  | SERIALIZABLE
  }
```

- **Example :**

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

THANK YOU!