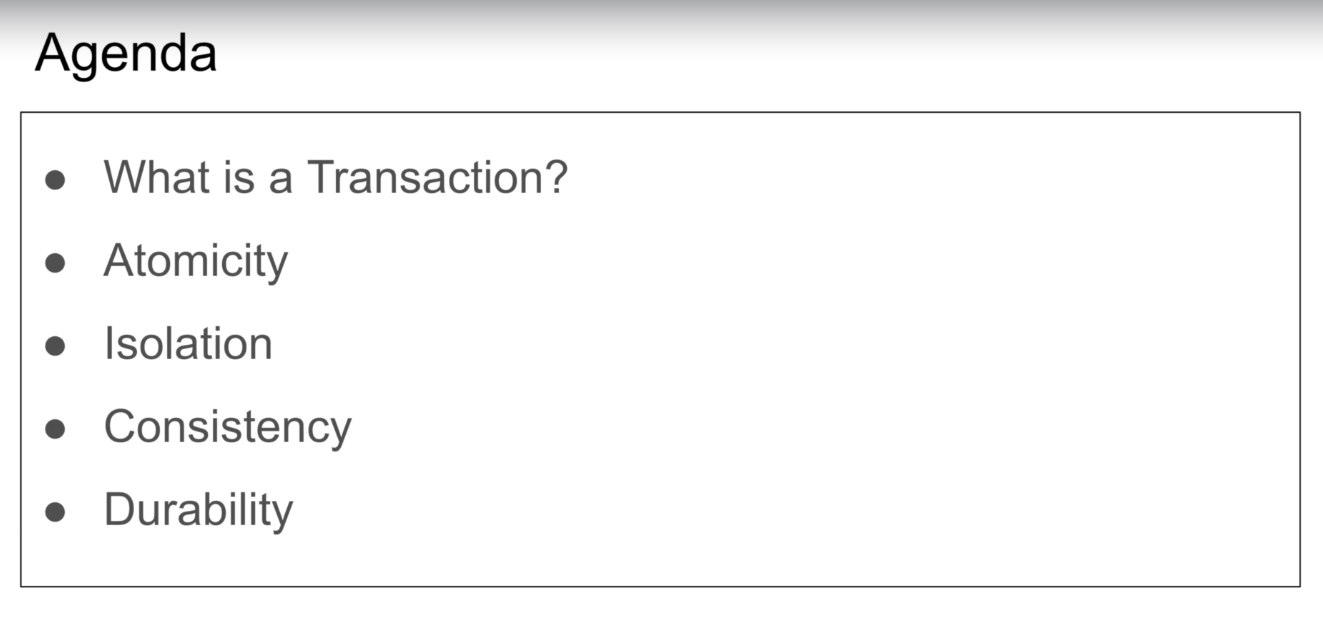
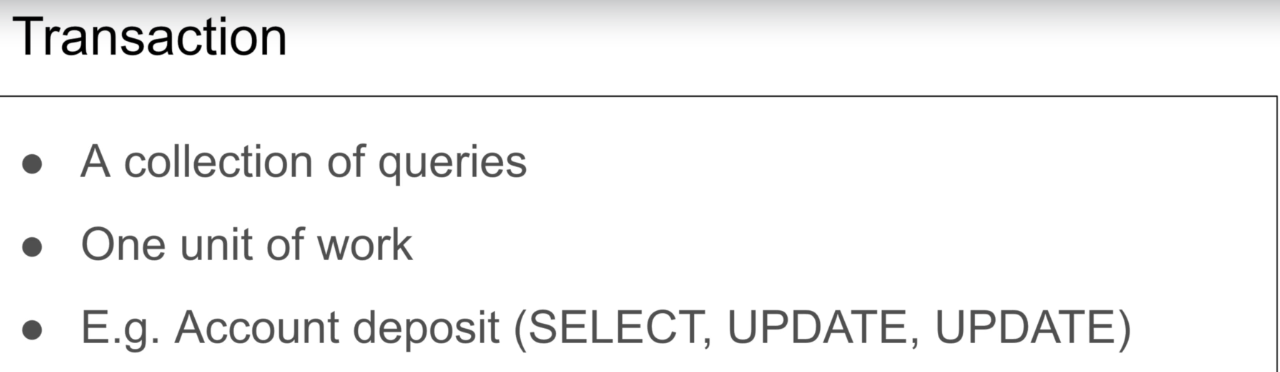
Organise your learnings at COMPANY NAME

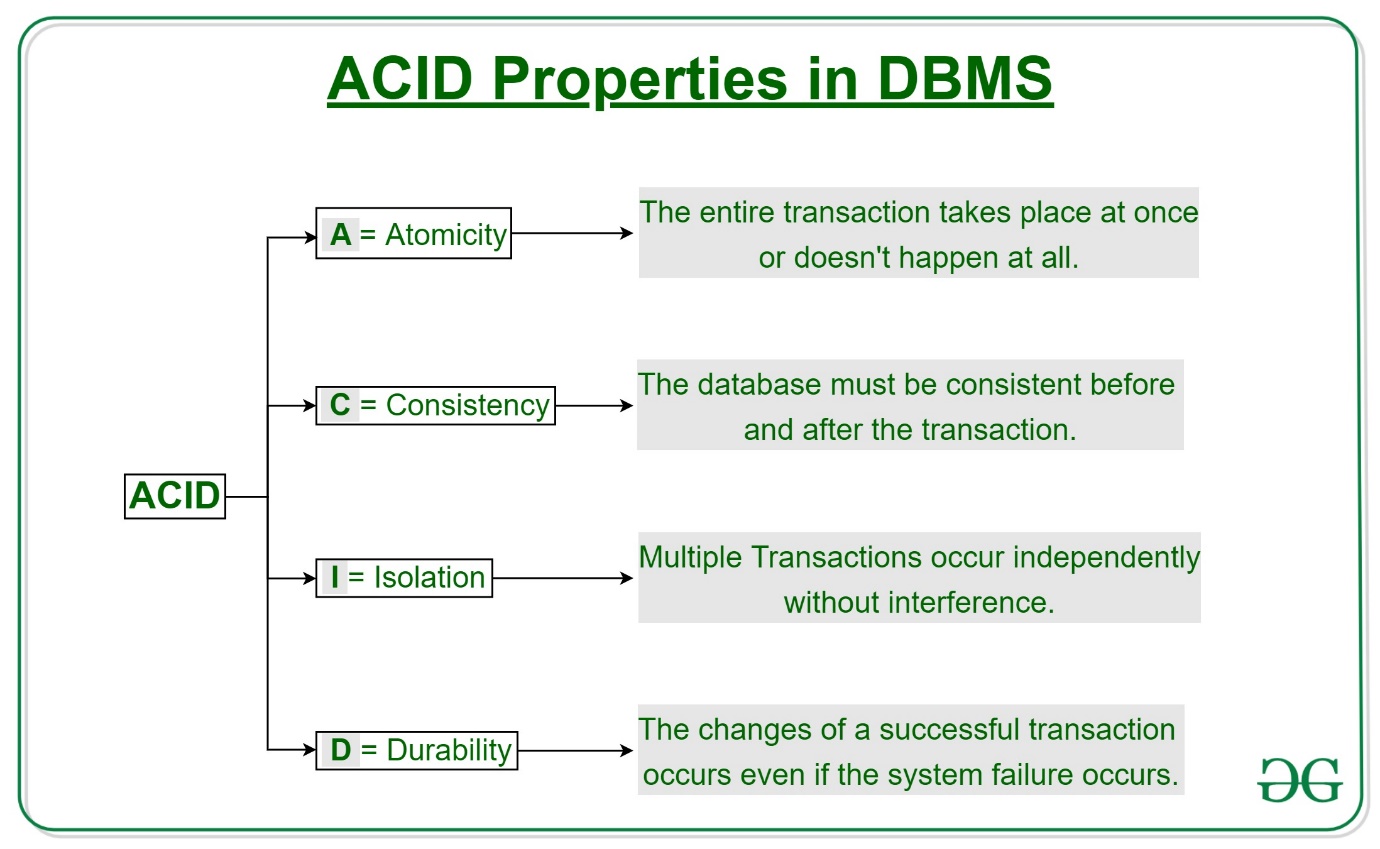
RELATIONAL DATABASE



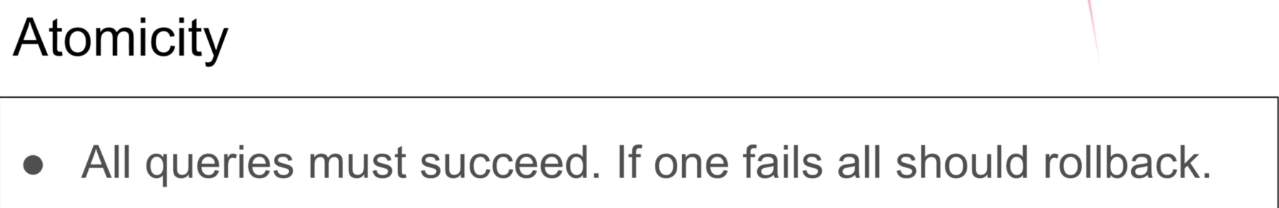


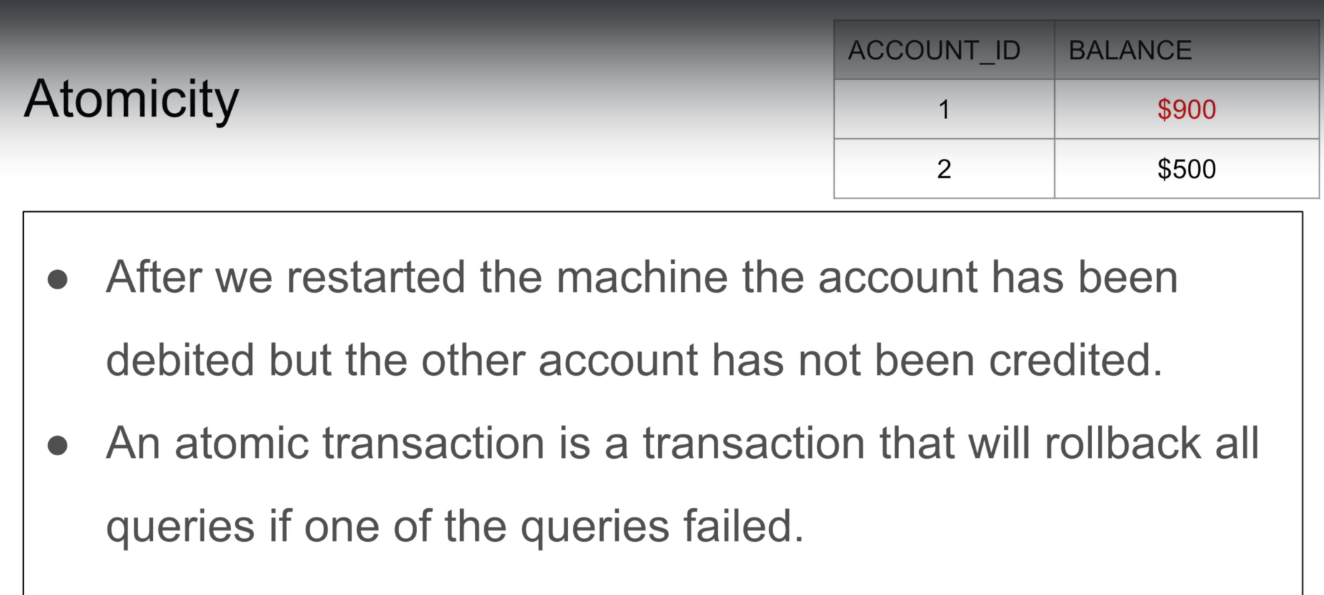
A **transaction** is a logical, atomic unit of work that contains one or more SQL statements. A **transaction** groups SQL statements so that they are either all committed, which means they are applied to the **database**, or all rolled back, which means they are undone from the **database.**

[**https://youtu.be/4EajrPgJAk0**](https://youtu.be/4EajrPgJAk0)



**Note:** A transaction should be atomic

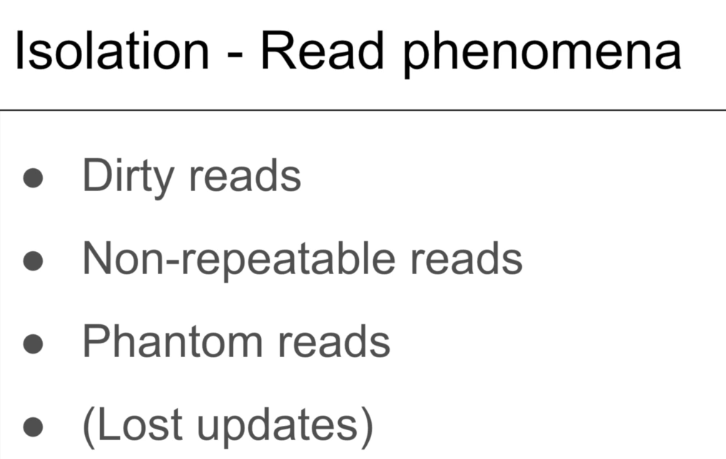


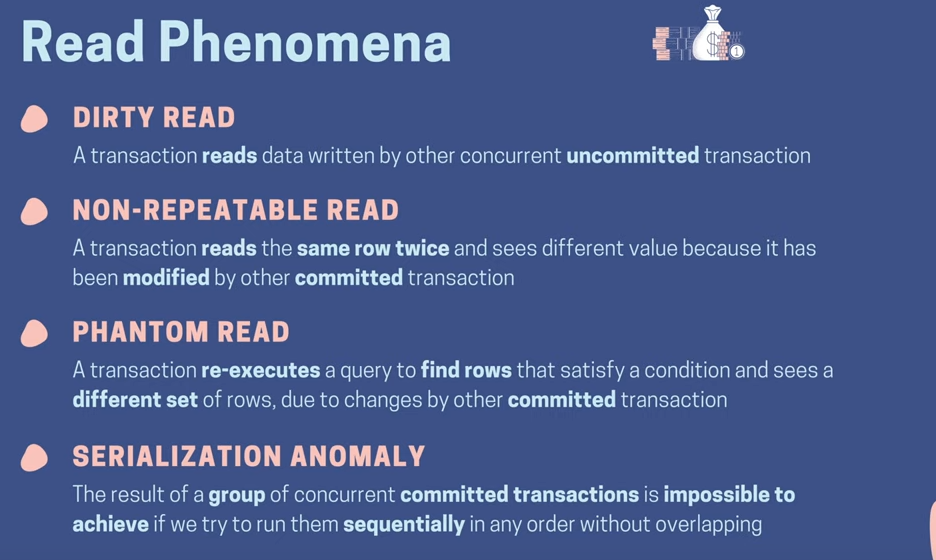




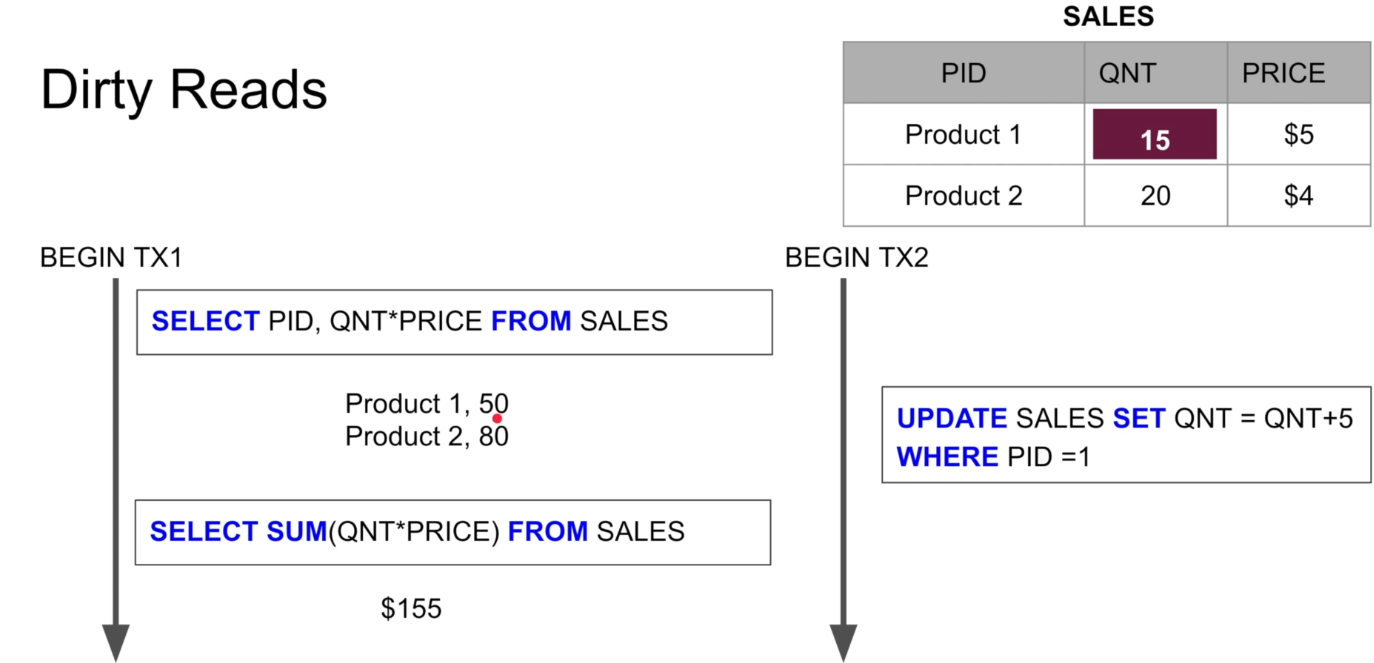
Isolation depends on how you did your implementation.

Without isolation we get a bunch of read phenomena, and to fix those read phenomen we use isolation levels.





Dirty read



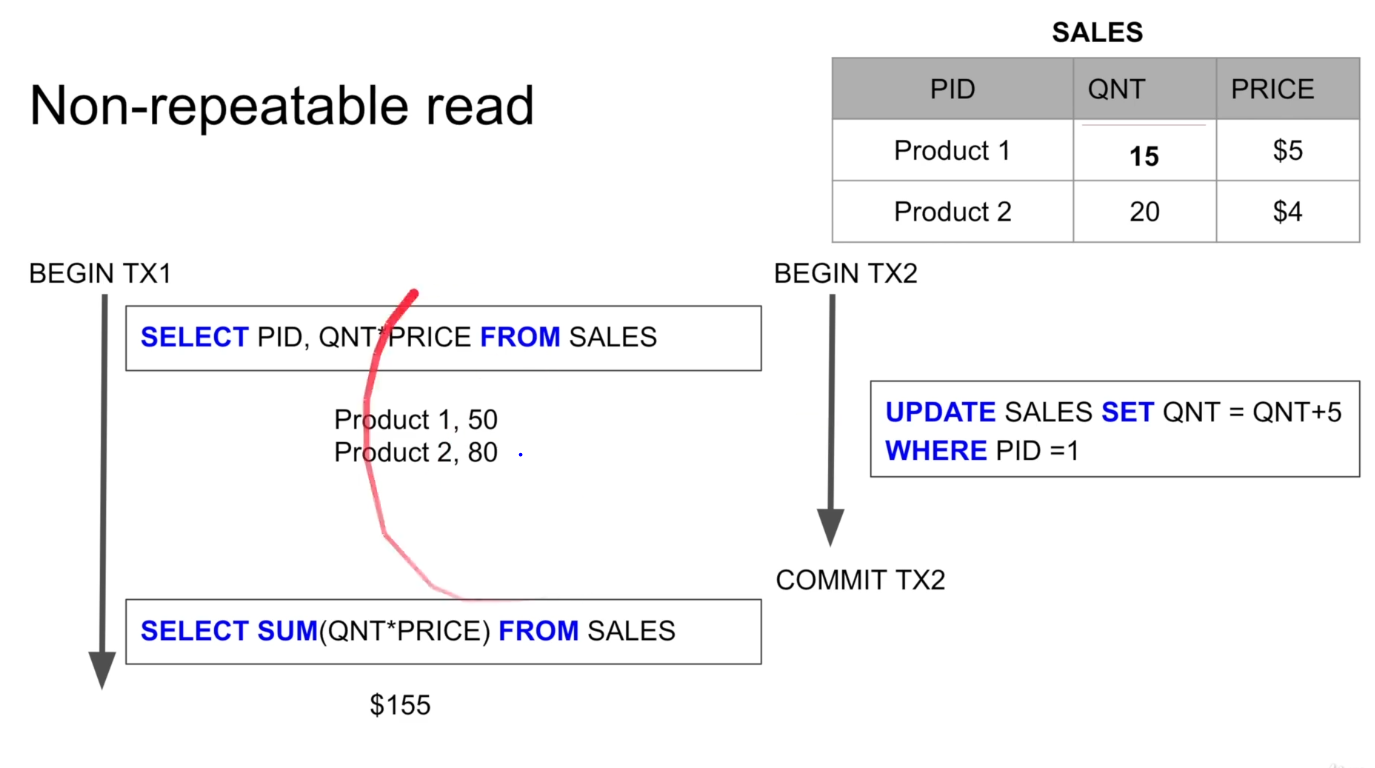
If you can tolerate dirty read then it’s not a problem, it depends on the application.

Eg-likes in Instagram (you don’t care)

The QNT 15 is not committed yet, that’s why it’s a dirty read.

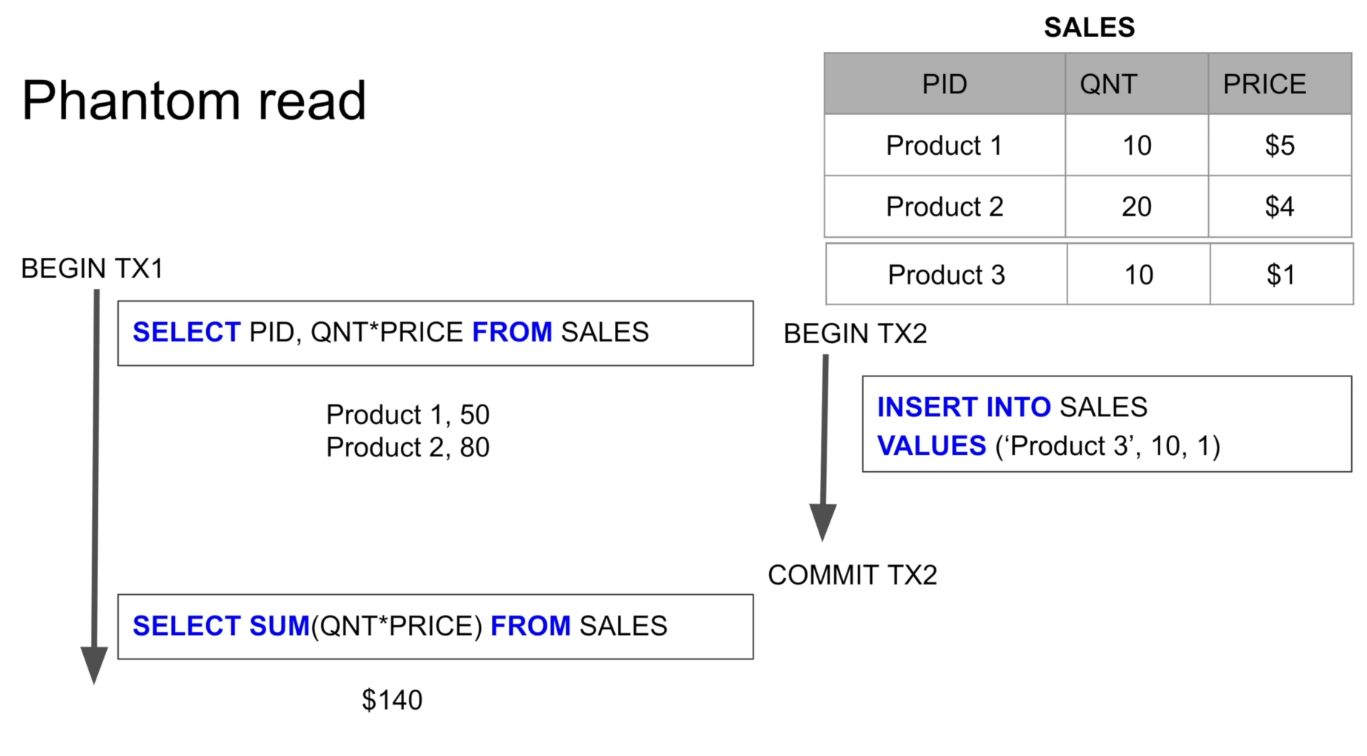
Non-repeatable read

You will get different result if you read same value multiple times.



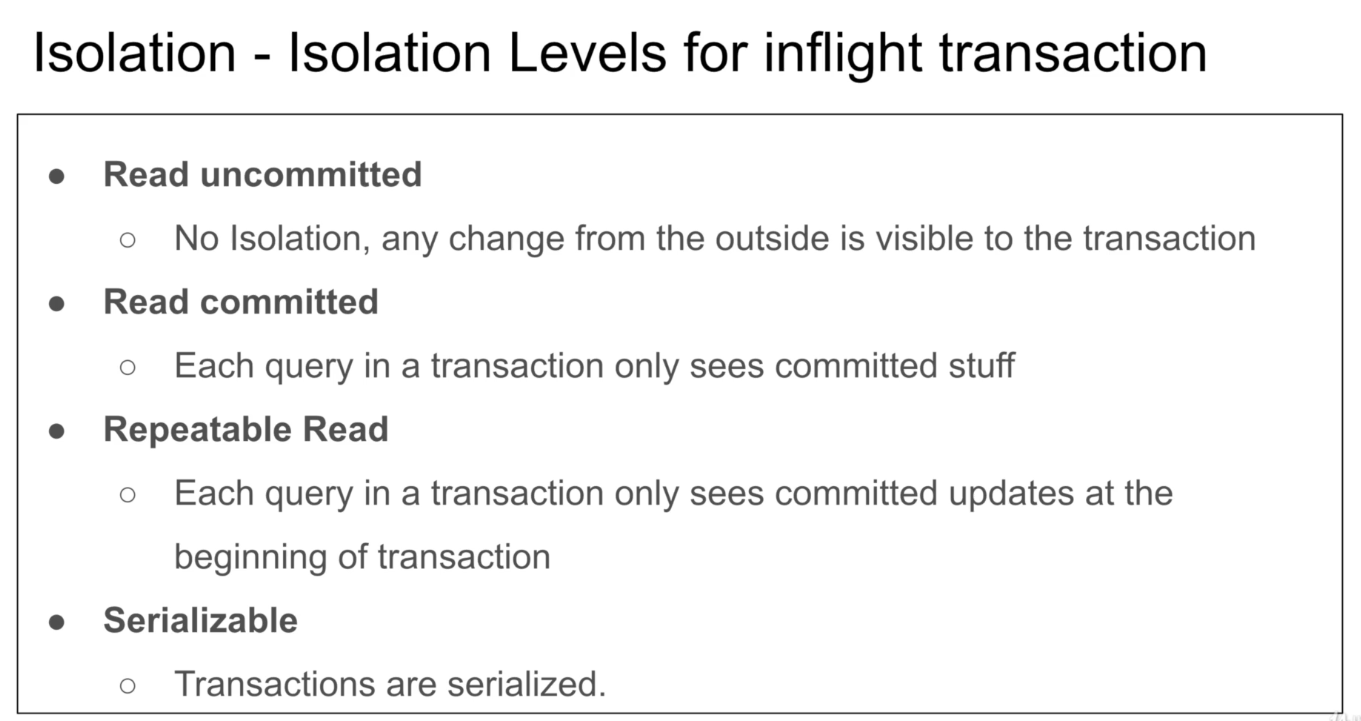
Note: Transaction 2 is actually committing the value.

PHANTOM READ



Note: Transaction 2 has actually committed the row. This is little bit harder to fix or avoid, previous one you can implement locks and version control to fix them.

Note: One of the way to fix it is using serialization.



These levels are implemented to fix those phenomena that occurs due to lack of isolation.

Note: Most of the time ‘read committed’ is the best transaction isolation level.Ex- memorymap

***Repetable Read***- it will see all the committed values(not at every single query) at the begging of the transcation.(before you start the transaction)

Some database implements versioning to implement this, eg-in Cassandra when you start a transaction I am in version zero, then anything I read belong to version zero of the database.

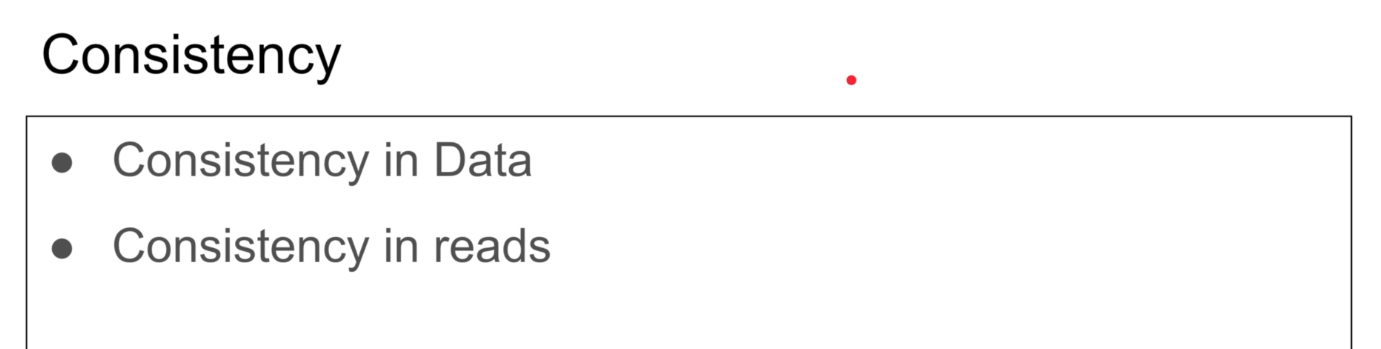
Some database implements as a lock.

**Serializable**- Performance goes down as you go down the level of isolation.





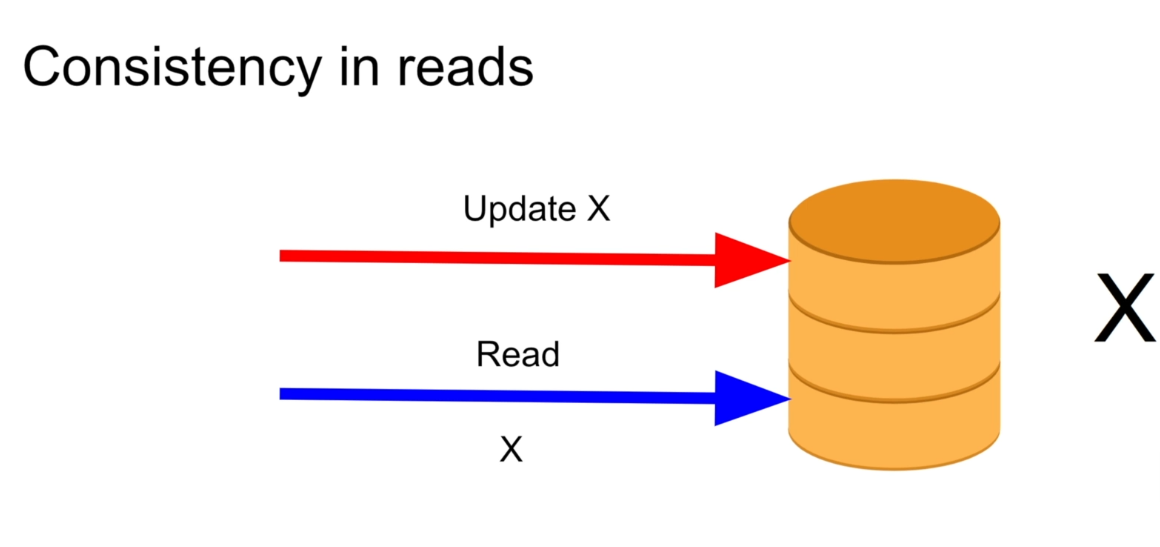
**CONSISTANCY(2 wasy we can have consistency)**



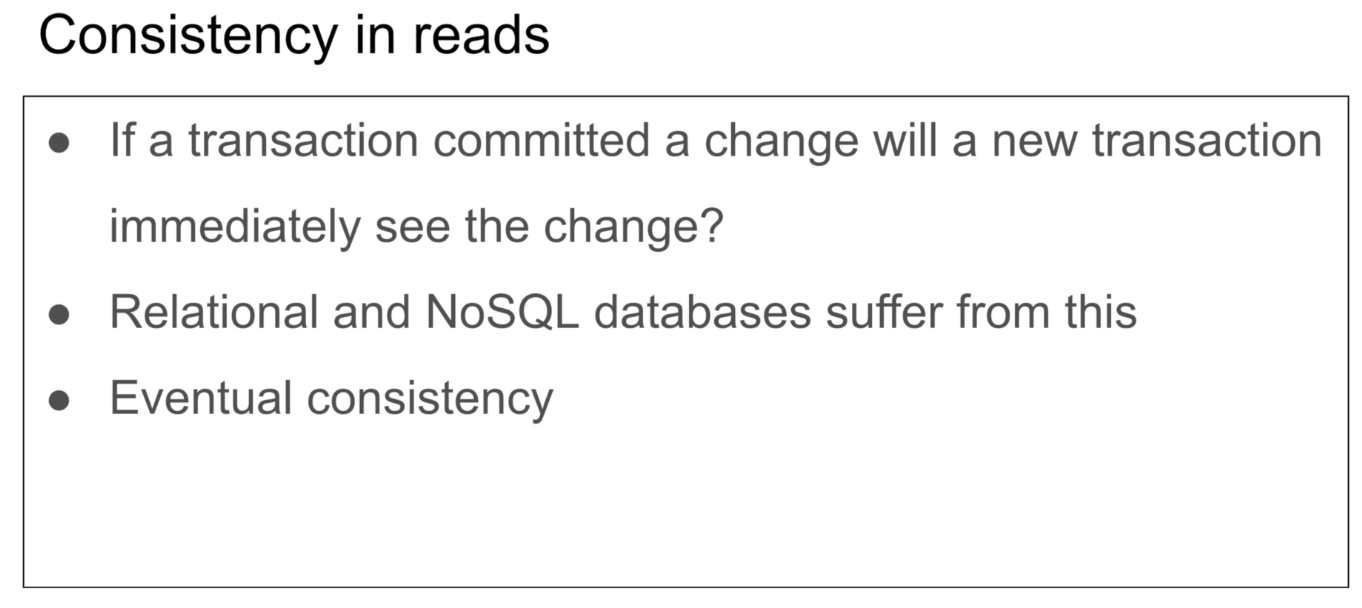
Atomicity and isolation leads to consistency.

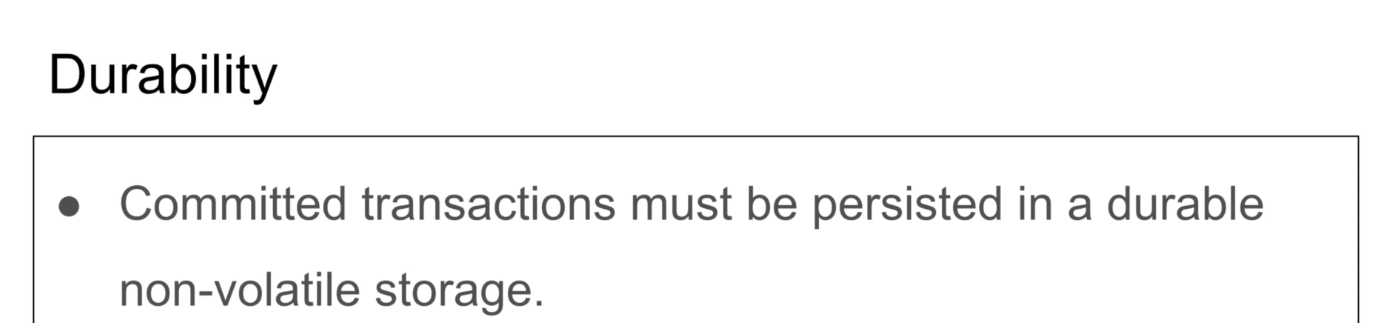
Note: Lack of atomicity leads to inconsistency.





Note: the moment you start doing leader and follower node thing, you are inconsistent. You write in primary you read in secondary and there is a delay from primary to secondary.





If it crashes it should be in disk when I start.

Reddis took the durability part and created in- memory database

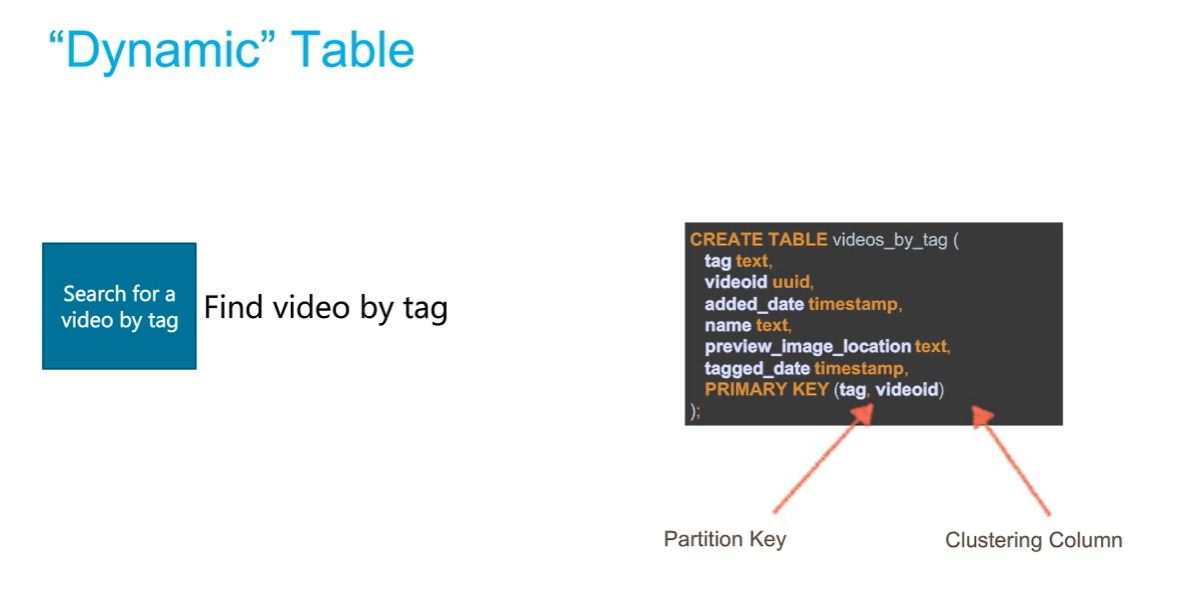
Casssanda and mongodb

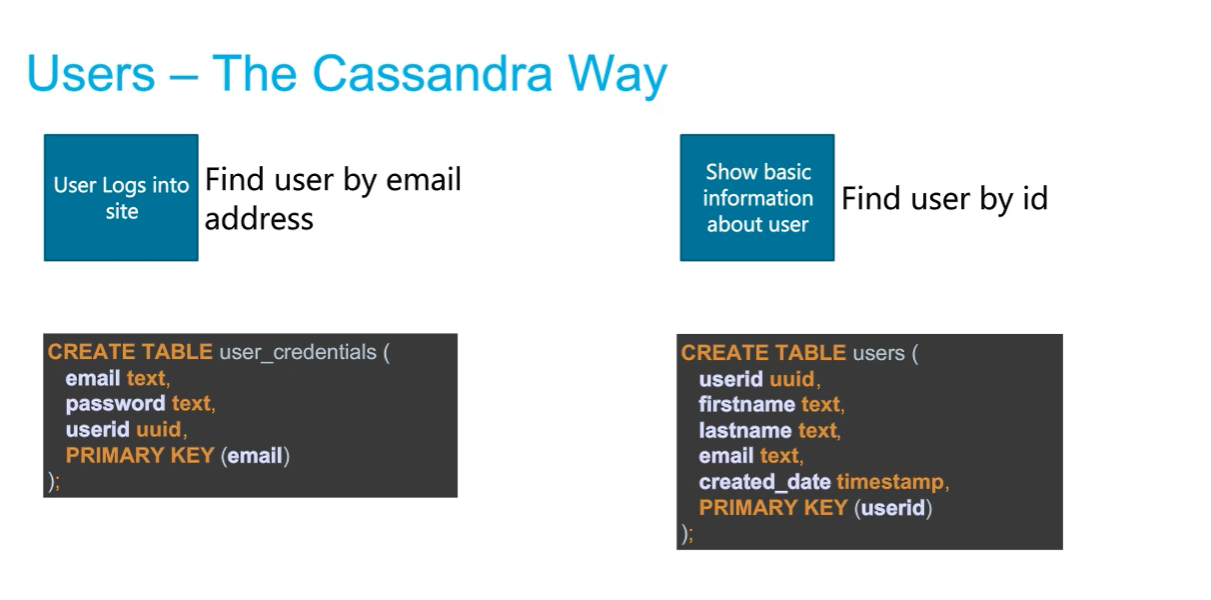




3)Make table



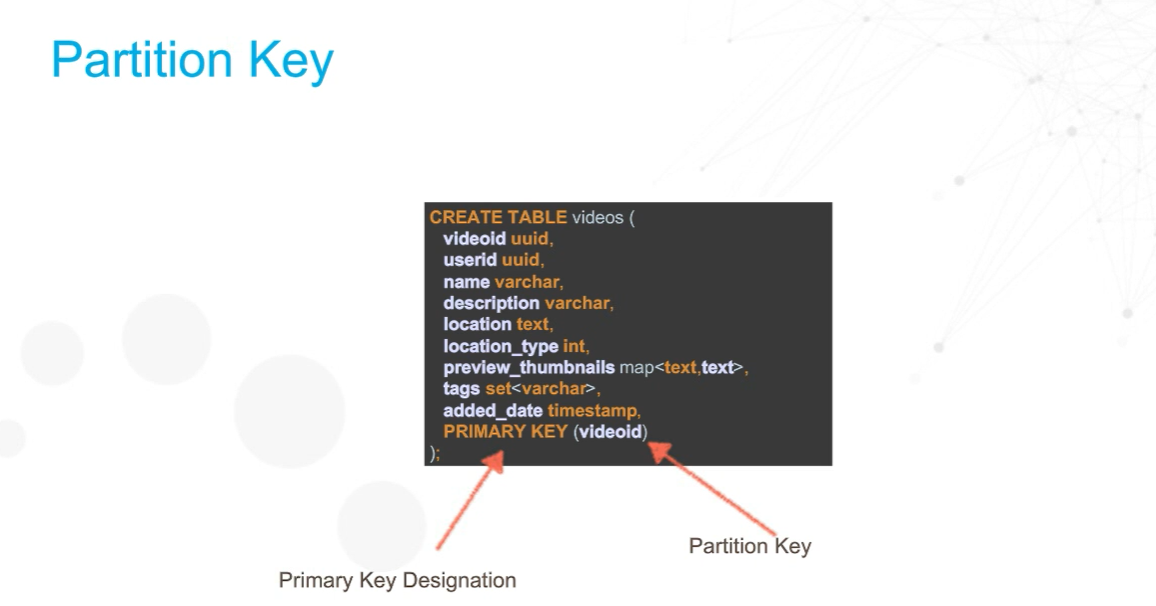




4.Getting the primary key right

Partition Key

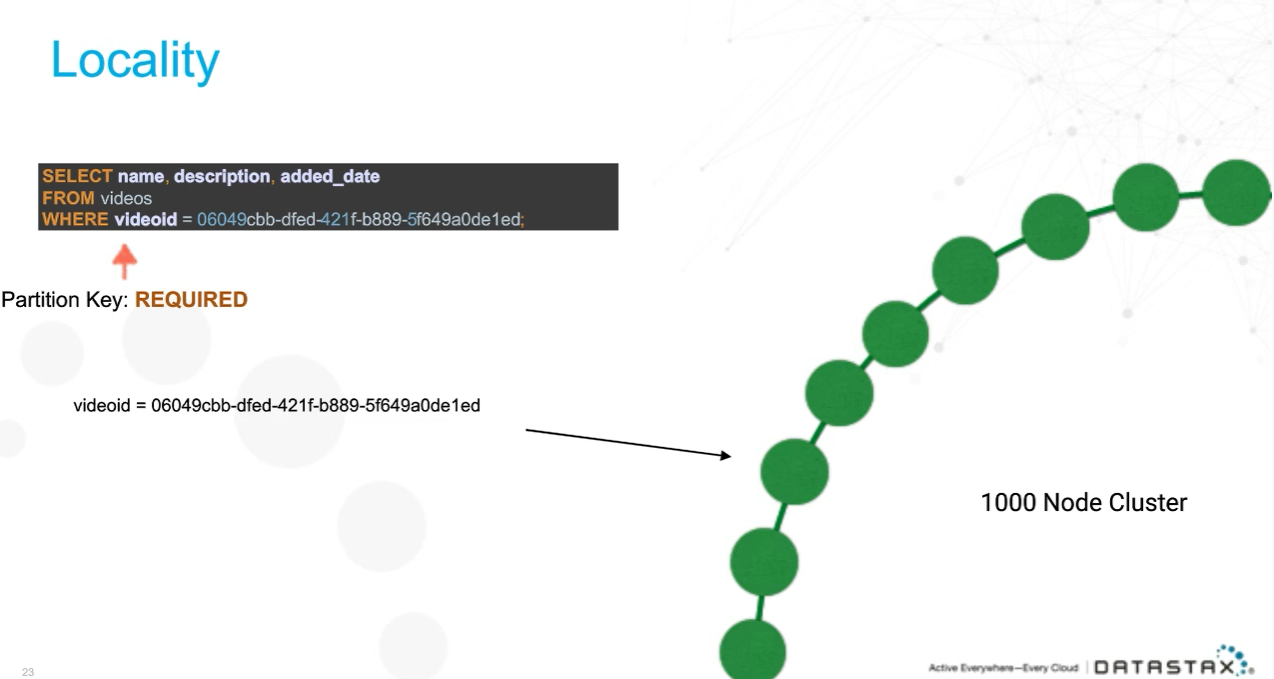
Ex PRIMARY KEY(videoid)



When we insert data into the database it takes that partition key(videoid) and hash it(uses a momentary hash and create a consistent hash).

Partition key makes it unique among other clusters.

Eg- we have 1000 clusters and we need to search something.Its going to look in every cluster. So when we select we use include partition key.



Partition key will get to the node.

Now in case of dynamic query we have partition key and clustering column.

PRIMARY KEY(tag, videoid)

Tag- Partition Key(node in the cluster)

Videoed-Clustering Column

