Transactions1:

How can we handle transactions? What could go wrong when handling transactions?

A Transaction is a single operation or series of operations and goal should be single only.

One Transaction performs all operations from start to end.

Here focus is DB transactions – series or group of operations (read / write) into a single unit

To book a flight:
- Look into flight portal
- Pick some flight
- Make a payment
- Notification my flight is booked

These 4 operations combined is a transaction
A transaction usually contains multiple steps

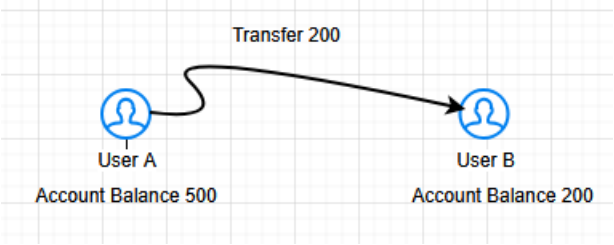
Updating some value: we are promoting Student to the next class. Update class value
Operations:
- Fetch the ID and perform a search (read some data from database)
- Update the value

Series of operations are needed to perform one Transaction

What are ACID properties?
ACID makes some promise to us how our data will be handled
ACID properties are some promises / guarantee, which are provided by transaction

1. Atomicity:
If we are performing an operation, it will be performed completely or nothing at all. If any of the operation is failed even at 99% everything will be rolled back. Say we write into two Replicas and even if one doesn't respond with an Ok message, the entire operation will be rolled back. Success if all parts of transaction is completed. We cannot break operations into smaller part.

2. Consistency:
Our data will be always in the correct form. Ensure correctness of the DB. Consistency is the only property that's not handled directly by the database, instead handled by the application itself.

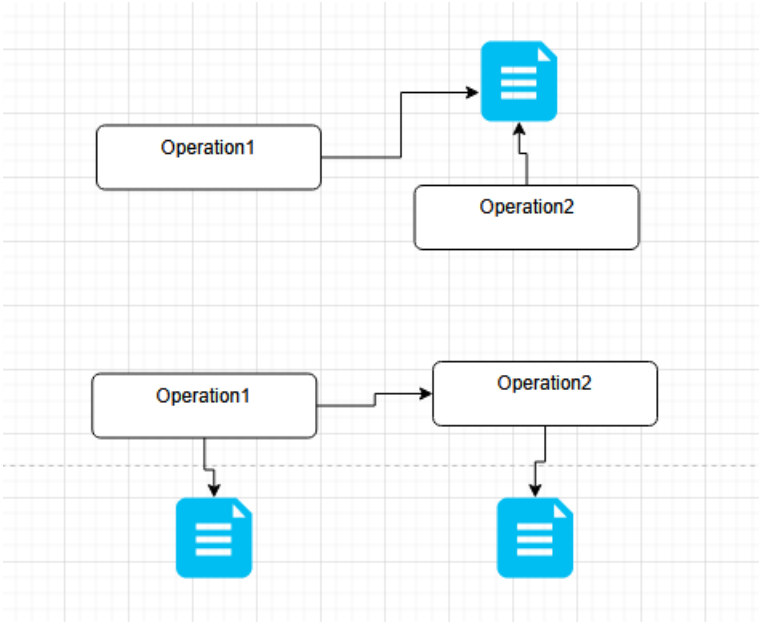


Operation: User A is transferring 200 to User B
After the operation, the new balances will be 300 and 200 respectively.
Total balance is always 700 before and after the operation. That means consistency is maintained.

Consistency tells us the state of the database before and after the operation.
Database cannot handle these things that's why we got to include an application layer. Similarly for a Delete record operation for 100-1=99, will always give 99 and it needs to be consistent.

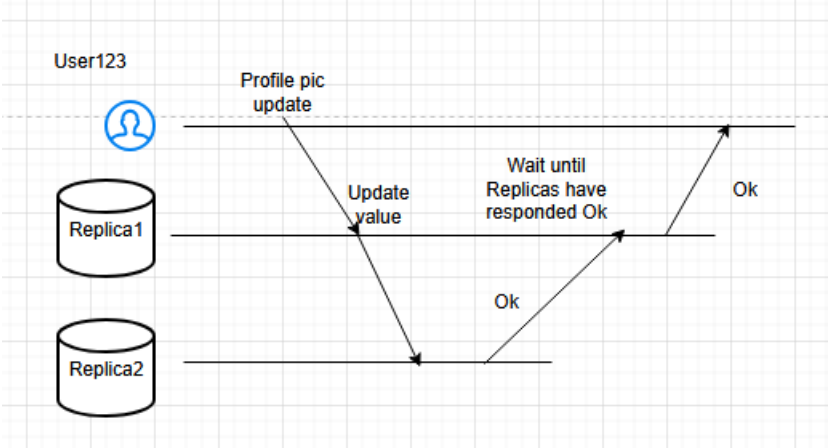
3. Isolation:
Isolation suggests even if we have concurrent operations or transactions are performed on a file, every operation should work as a single entity only. Operations should work in isolation.

Say if I perform two operations concurrently or in a sequence, the output or end result after the operations should be the same.



Operations should be performed independently and in isolation, without compromising our consistency File has the same result value no matter how the operations are performed on it.

4. Durability:
Whichever operation or transaction that are modifying things in our database those changes should not be temporary, they should be permanent. If a change is done, it should be really done and it should reflect in each and every manner moving forward. Once transaction is committed, it will be saved entirely.



Only when all Replicas are updated, it is marked as done not before that.
WAL (Write Ahead Log) is a great example of Durability. We create Write-Ahead-Log before starting to update database/Replica. Even if the db crashes, we will have the exact data that went into the db saved in WAL. So we can write into another DBs with the same WAL data. Another example of Durability. If commits are created then there is no rollback.