Day 9 Distributed transactions

## Vertices

Eg Scale

Horizontal

Merge DB → Add new tables
→ Flex schema
→ Denormalize
→ Avoid
→ NY metadata

Sharding Index

(extended use of Relational data base in vertical scaling is good but not horizontal scaling)

Eg Flex DB → Time series
→ Shard
→ Subdb
→ Flexible Data
→ Multiple Sharding

Neo4j → F B

Cassandra → Twitter
→ Customer journey

## SQL/Relational DB

→ ACID transactions

A - Atomicity → Single op fails or more
Eg: book transfer from A to B

C - Consistency → maintaining data integrity

I - Isolation → Concurrently without interfering with one another

D - Durability → Permanent, Persistent

## Amazon

Payment | Inventory

Having different DB can be challenging how to overcome → Distributed transaction

Payment | Inventory
order management
Apis

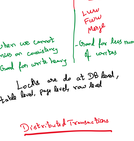
Life cycle of a transaction

→ Begin Transaction
→ Execute operations
→ Consistency check
→ Commit/Rollback

Distributed transaction

when transaction is distributed across DBs in network
So it should have better failure handling, better rollback capability

## SQL DB




R W | R W
✓ ✗ | ✗ ✗

Locks are expensive as many reads to be collected to block Read lock or writes block + it defeats the purpose of parallelism

Types of locking

Pessimistic Lock

Lock(s) before (write lock)

By default choose this if high volume

Optimistic Lock
Version number
Timestamp

Called "optimistic" because

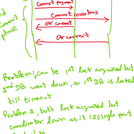
Low Fund
Many

- Good for less number of writes

Actions are cancel mid-transaction
- Good for write heavy

Locks are do at DB level, table level, page level, row level

## Distributed Transactions




(Lock is not serializable
Lock
Lock
Lock