ACID transactions

- Set of properties designed to ensure reliability and consistency of database transactions.
- Provide a high level of assurance that database transactions will be processed reliably and data will be stored accurately and consistently.
- Atomicity, Consistency, Isolation and Durability.
- Atomicity
    - each transaction is a single, indivisible unit.
    - Any part fails, entire transaction is rolled back.
- Consistency
    - database will be in valid state before and after the transaction.
    - Any transaction violating schema constraints will be rolled back.
- Isolation
    - Each transaction operates independently of other transactions.
    - Each transaction should only become visible to other transactions after it has committed.
- Durability
    - The changes made to the database during a transaction are irreversible even in a system failure.
    - Any changes made after committing a transaction persist even if the system loses power or is destroyed.

Disadvantages of ACID transactions

- High processing overhead
- Possible deadlocks with transactions waiting for each other to release resources.
- Performance and scalability challenges.

Alternatives to ACID transactions

- BASE
    - Basically Available, Soft State, Eventually consistent.
    - Emphasizes availability and partition tolerance over consistency.
- CAP
    - Consistency, Availability, Partition tolerance.
    - Emphasizes trade-off between consistency and partition tolerance.
- NoSQL databases
    - No rigid consistency standards
    - Prioritize performance and scalability over immediate consistency.
    - High throughput, low data consistency applications.

Challenges of implementing acid transactions in distributed systems

- In distributed systems, network <u>latency</u> can impact the performance of ACID transactions. Longer transaction times and higher overhead can result from network communication delays.
- Maintaining <u>consistency</u> across all nodes in a distributed system can be challenging. A distributed system's nodes might store different versions of the same data, which could result in discrepancies.
- Distributed system <u>availability</u> can be difficult. Nodes could fail, and it might be challenging to maintain the system's responsiveness.
- As the number of nodes in a distributed system increases, it becomes more difficult to maintain <u>scalability</u>.

Solutions to maintain acid properties in distributed systems

- Two phase commit - all nodes agree to commit a transaction before it is committed.
- Multi-version concurrency control (MVCC) - enabling multiple versions of same data to co-exist.
- Replication - multiple copies of same data on various nodes.
- Sharding - dividing data across multiple nodes.