Moving Beyond the Relational Model

Relational Data Model: good for structured data, can handle large amounts of data with edits from different contributors. Lots of existing tooling out there.

ACID Compliance

- Atomicity: transaction is treated as an atomic unit; it is fully executed or no parts of it are executed.
- Consistency: A transaction takes a database from one consistent state to another consistent state (meets all integrity constraints)
- Isolation: Any two transactions can happen at the same time unless they affect each other.
    - Dirty read: one transaction reading a row that has been modified by another transaction but not yet committed (may have a rollback)
    - Non-repeatable read: inconsistent reads in a single transaction because data was modified in between from another transaction.
    - Phantom Read: Adding/Deleting rows from a row that is being used by another transaction. Inconsistency in results in the same transaction.
- Durability: once a transaction is completed and committed, its changes are permanent, even in the event of a system failure.

Transaction: a sequence of one or more CRUD (Create, Read, Update, Delete) operations performed as a single, logical unit of work. Either the entire sequence succeeds (COMMIT) or the entire sequence fails (ROLLBACK or ABORT).

Distributed system: collection of independent computers that appear to it's users as one computer. Computers operate concurrently, fail independently, and don't share a global clock.

- Replication: creating multiple of the same instance. Increases redundancy, consistency, and data availability.
- Sharding: partition data into smaller, more manageable, pieces. Each shard is a replica set that is responsible for a portion of the overall data.

CAP Theorem: specifically for distributed data stores

- Consistency: Every user of the DB has an identical view of the data at any given instant
- Availability: In the event of failure, the database remains operational
- Partition Tolerance: The database can maintain operations in the event of the networks failing between two segments of the distribution system.

Consistent and Available: (RDBMS, MySQL), system always responds with the latest data and every request gets a response, but may not be able to deal with network issues

Consistent and Partition Tolerant: (MongoDB, Redis) If system responds with data from a distributed store, it is always the latest, else data request is dropped.

Available and Partition Tolerant: (DynamoDB, Cassandra) System always sends and responds based on distributed store, but may not be the absolute latest data.

ACID is considered a pessimistic concurrency model: if something can go wrong it will go wrong

Optimistic concurrency: assumes conflicts wont happen so it doesn't try and block them.