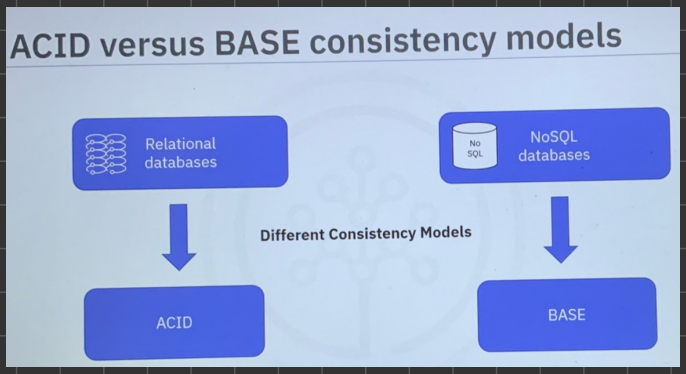
ACID vs BASE consistency models

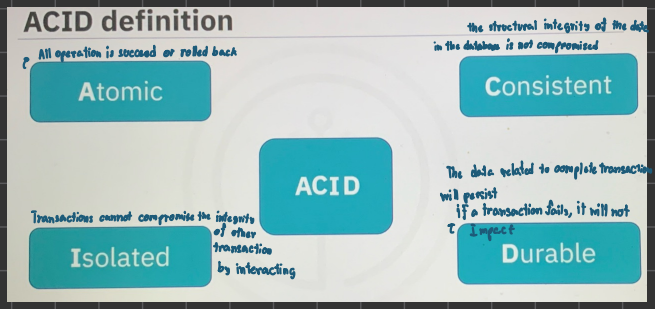
Objectives

- Define the ACID and BASE acronyms
- Describe the differences between ACID and BASE
- Identify the use cases for ACID and BASE modeled systems

ACID versus BASE consistency models



ACID definition



ACID consistency model

- Used by relational databases
- Ensures a performed transaction is always consistent
- Used by:
  - Financial institutions
  - Data warehousing
- Databases that can handle many small simultaneous transactions => relational databases
- Fully consistent system

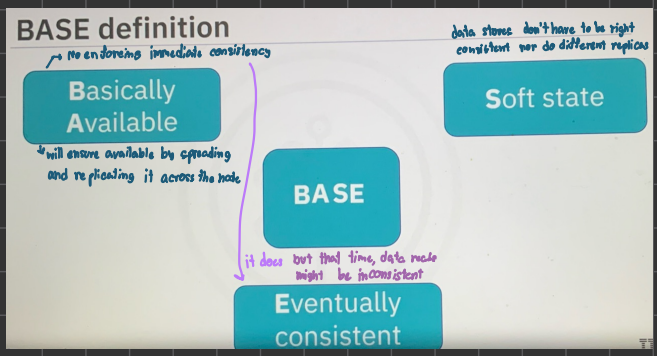
ACID database use cases

Financial institutions: Money transfers

NoSQL and BASE models

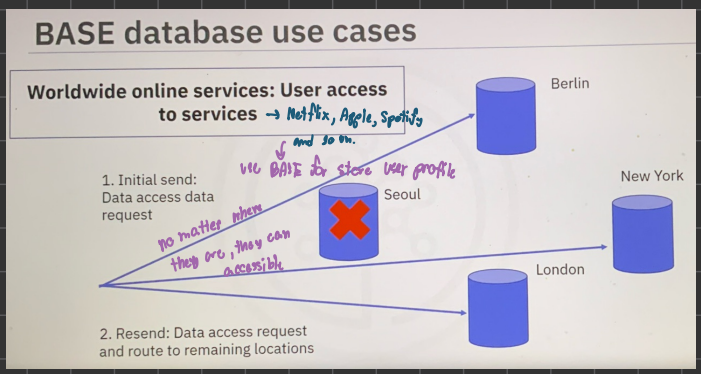
- NoSQL has few requirements for immediate consistency, data freshness, and accuracy
- NoSQL benefits: availability, scale, and resilience
- Used by:
  - Marketing and customer service companies
  - Social media apps
  - Worldwide available online services
- Favors availability over consistency of data
- NoSQL databases use BASE consistency model
- Fully available system

BASE definition



BASE database use cases

Worldwide online services: User access to services → Netflix, Apple, Spotify and so on.



Distributed Databases

What you will learn

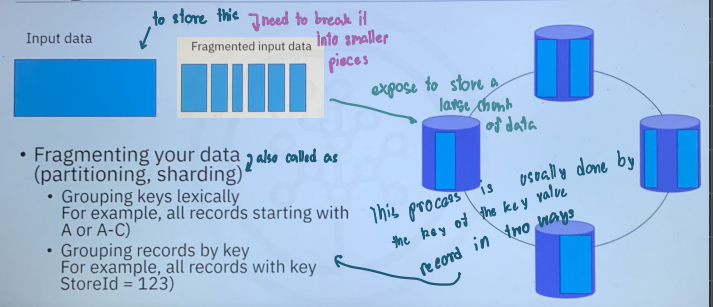
- Describe the concepts of distributed systems
- Define fragmentation and replication of data
- Describe the advantages and challenges of distributed systems

Concepts of distributed systems

A collection of multiple interconnected databases: spread physical across various locations that communicate via a computer network

- Physically distributed by fragmenting and replicating data
- Follows the BASE consistency model

Fragmentation

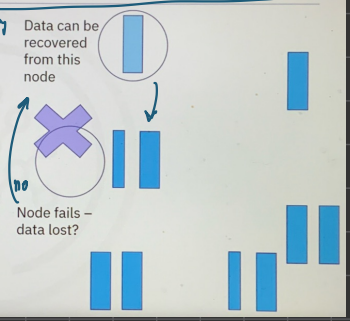


- Fragmenting your data (partitioning, sharding) also called as
  - Grouping keys lexically
    For example, all records starting with A or A-C)
  - Grouping records by key
    For example, all records with key StoreId = 123)

Replication

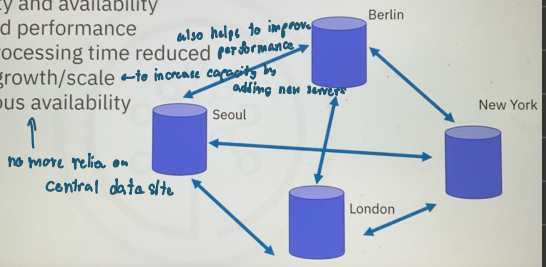
we don't lose all of the data if that node, if that node fails, by

- Protection of data for node failures
- Replication: all data fragments are stored redundantly in two or more sites
- Increases the availability of the data
- Replicated data needs to be synchronized => could lead to inconsistencies



Advantages of distributed systems

- Reliability and availability
- Improved performance
- Query processing time reduced
- Ease of growth/scale
- Continuous availability

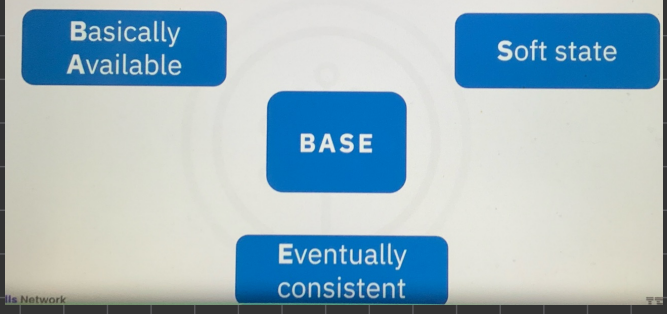


Distributed database challenges

- Distributed databases bring availability, fast scaling, and global reach capabilities
- Challenge: Concurrency control => consistency of data
  - WRITES/READS to a single node per fragment of data => data is synchronized in the background
  - WRITE operations go to all nodes holding a fragment of data, READS to a subset of nodes per Consistency
  - Developer-driven consistency of data
- No Transaction support (or very limited)

how can data synchronization be secured?

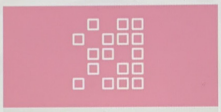
Distributed systems – BASE model



CAP theorem

Objectives

Describe the CAP theorem



Describe the characteristics of CAP theorem

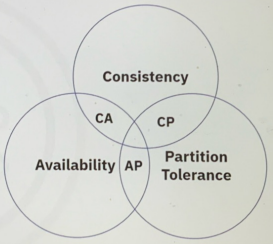
Describe the history and relevance of CAP theorem

Consistency versus availability

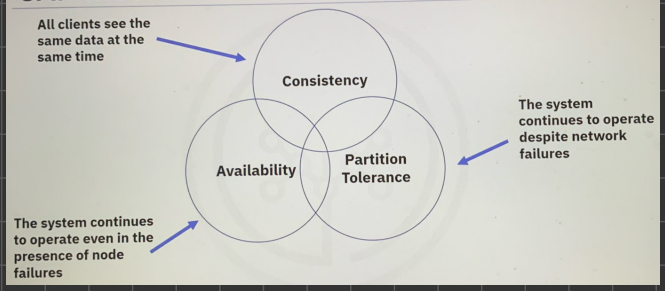
- Early 2000s: Emerging of Hadoop, the first open big data architecture that allowed distributed storage and processing of large amounts of data
- Services emerging in 2000s required distributed databases
  - Active and accessible worldwide
  - Always available
- Relational databases: ACID-based, relied on data consistency
- Availability and consistency seemed impossible

CAP theorem definition and history

- CAP Theorem (Brewer's theorem)
- Evolved by MIT professors Seth Gilbert and Nancy Lynch
- Consistency, Availability and Partition Tolerance (CAP)
- Only two of the three characteristics can be guaranteed in distributed systems

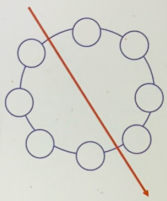


CAP theorem



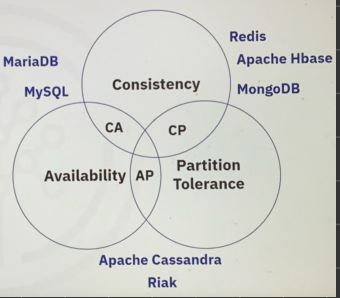
Partition tolerance

- A Partition is lost or temporarily delayed connection between nodes
- Partition tolerance means that the cluster must work despite network issues
- Distributed systems cannot avoid partitions and must be partition-tolerant
  Partition tolerance is a basic No SQL feature



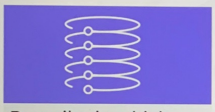
NoSQL: CP or AP

- NoSQL: A choice between consistency and availability
- MongoDB: Chooses consistency
- Apache Cassandra: Chooses availability
- These are tunable systems

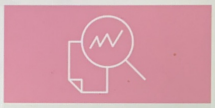


Challenges in Migrating from RDBMS to NoSQL Databases

What you will learn



Describe in which cases RDBMS and NoSQL are best used

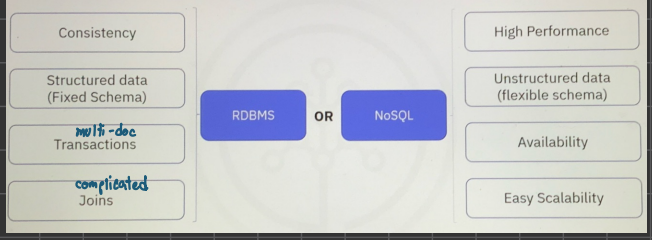


Describe the main differences between RDBMS and NoSQL

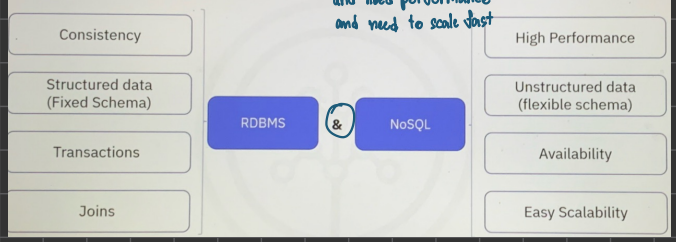
RDBMS to NoSQL: Overview

- NoSQL is not a de facto replacement for RDBMS ← they don't competing
- RDBMS and NoSQL cater to different use cases ← and requirements
- RDBMS to NoSQL should be done based on careful case-by-case analysis based on performance and flexibility requirements

RDBMS or NoSQL



RDBMS and NoSQL – consisting of a large amount of data and need performance and need to scale fast



RDBMS to NoSQL: A mindset change

- **Data-driven** model to **query-driven** data model
  - RDBMS: Starts from the data integrity, relations between entities
  - NoSQL: Starts from your queries, not from your data models, based on the way the application interacts with the data
- **Normalized** to **Denormalized** data
  - NoSQL: Think how data can be structured based on your queries
  - RDBMS: Start from your normalized data and then build the queries

RDBMS to NoSQL: A mindset change

- **From ACID to BASE model**
  - Choose availability versus consistency
  - Apply the CAP Theorem: choose between availability and consistency
  - Consider availability, performance, geographical presence, high data volumes
- **NoSQL systems, by design, do not support transactions and joins except in limited cases**

Glossary

Term	Definition
ACID	This term is an acronym for Atomicity, Consistency, Isolation, and Durability, which is a set of properties that guarantee reliable processing of database transactions in traditional relational databases.
Atomic	In the context of database transactions, atomic means that an operation is indivisible and either completed fully or is completely rolled back. It ensures that the database remains in a consistent state.
Availability	In the context of CAP, availability means that the distributed system remains operational and responsive, even in the presence of failures or network partitions. Availablity is a fundamental aspect of distributed systems..
BASE	An alternative to ACID. Stands for Basically Available, Soft state, Eventually consistent. BASE allows for greater system availability and scalability, sacrificing strict consistency in favor of performance.
Basically available	A basically available system remains operational even in the presence of failures or faults.
CAP	CAP is a theorem that highlights the trade-offs in distributed systems, including NoSQL databases. CAP theorem states that in the event of a network partition (P), a distributed system can choose to prioritize either consistency (C) or availability (A). Achieving both consistency and availability simultaneously during network partitions is challenging.
Consistency	In the context of CAP, consistency refers to the guarantee that all nodes in a distributed system have the same data at the same time.
Consistent	The action of being consistent ensures that a database transaction transforms the database from one consistent state to another.
Denormalized	Denormalization is a database design technique used in NoSQL databases (and sometimes in traditional relational databases) where redundant or duplicate data is intentionally introduced into one or more tables to improve query performance and reduce the need for complex joins.
Durable	Guarantees that once a transaction is committed, its changes are permanent and will survive any system failures.
Eventually consistent	An eventually consistent system reaches a consistent state, where all nodes have the same data given that there are no new updates.
Isolated	Isolation refers to the property that multiple transactions can run concurrently without affecting each other.
Joins	Combining data from two or more database tables based on a related column between them.
Normalized	A database design practice where data is organized to minimize redundancy and maintain data integrity by breaking it into separate tables and forming relationships between them.
Partition tolerance	In the context of CAP, partition tolerance is the ability of a distributed system to continue functioning even when network partitions or communication failures occur.
Replication	Replication involves creating and maintaining copies of data on multiple nodes to ensure data availability, reduce data loss, fault tolerance (improve system resilience), and provide read scalability.
Sharding	Refers to the practice of partitioning a database into smaller, more manageable pieces called shards to distribute data across multiple servers. Sharding helps with horizontal scaling.
Soft state	A soft state acknowledges that the system's state might be transiently inconsistent due to factors like network partitions or concurrent updates. And it's willing to accept a certain level of inconsistency or uncertainty in the data temporarily.
Transactions	Transactions are sequences of database operations (such as reading and writing data) that are treated as a single, indivisible unit.