Big Data Storage Concepts

Chapter 5
Instructor: Houssein Dhayne
houssein.dhayne@net.usj.edu.lb

Table of content

Data Wrangling and Storage

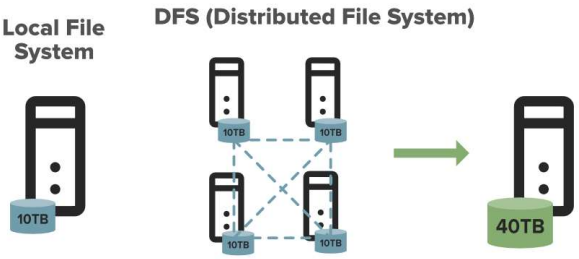
- Data wrangling is essential for preparing data for storage and processing.
- Storage is required when:
  - Acquiring external datasets or using internal data in a Big Data environment.
  - Manipulating data for analysis.
  - Processing data through ETL activities or analytical operations.

Clusters

- Definition: Tightly coupled collection of servers (nodes) with the same hardware specifications.
- Task Execution: Splits tasks into smaller pieces, distributes them across nodes.
- Dedicated Resources: Each node has its own memory, processor, and hard drive.

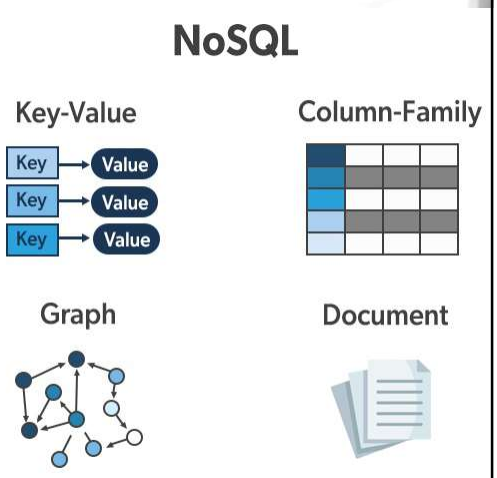
File Systems and Distributed File Systems

- File System: Method of storing and organizing data on storage devices.
- Distributed File System: Spreads large files across cluster nodes.
- Logical View: Presents a tree structure of directories and files.
- Examples: Google File System (GFS), Hadoop Distributed File System (HDFS).



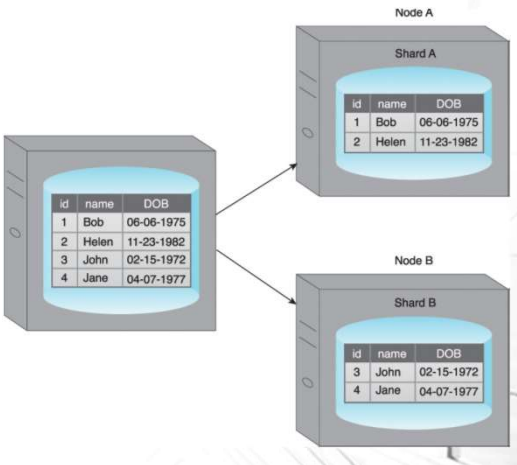
NoSQL

- **Data Model Variety:**
  - They come in a variety of types based on their data model, such as document, key-value, wide-column, and graph.
- **Flexible Schemas:**
  - Unlike traditional relational tables, NoSQL databases offer flexible schemas, accommodating varied data structures.
- **Scalability and Performance:**
  - Designed for scalability, NoSQL databases excel in handling large data volumes and high user loads, making them ideal for big data applications.
- **Use Cases:**
  - Particularly suited for distributed data stores with extensive storage requirements, NoSQL databases shine in scenarios like big data analytics and real-time web applications.



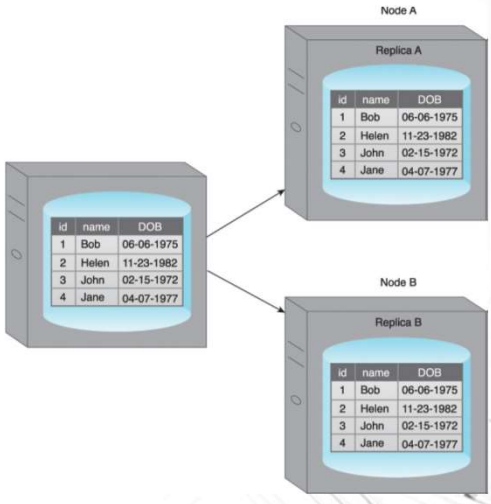
Sharding

- Definition: Horizontally partitioning a large dataset into smaller, manageable datasets (shards).
- Distribution: Shards are distributed across multiple nodes.
- Benefits: Horizontal scalability, improved read/write times.
- Transparency: Often transparent to the client.



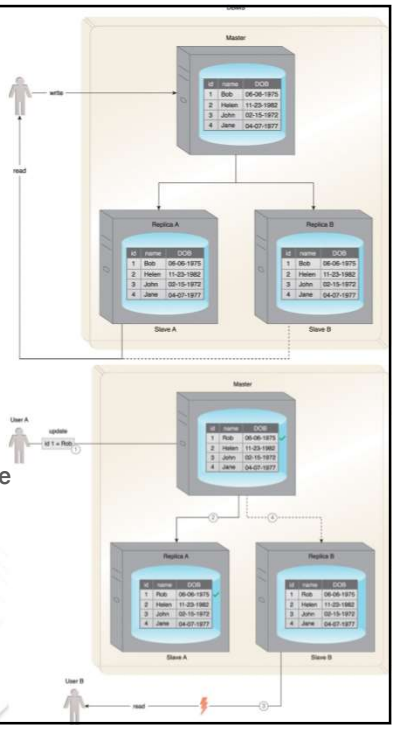
Replication

- Definition: Storing multiple copies (replicas) of a dataset on multiple nodes.
- Methods: Master-slave and peer-to-peer replication.
- Benefits: Scalability, availability, fault tolerance.



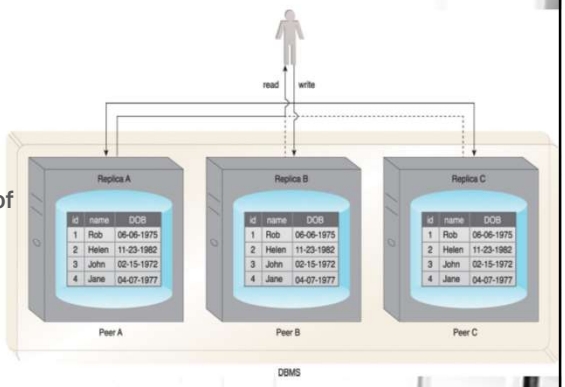
Master-Slave Replication

- Replication method where nodes are arranged in a master-slave configuration.
- All data writes (insert, update, delete) occur on the master node.
- Read requests can be fulfilled by any of the slave nodes.
- Ideal for read-intensive loads.
- Horizontal scaling by adding more slave nodes to handle growing read demands.
- Writes are consistent, coordinated by the master node.
- Potential for read inconsistency if a slave is read before an update from the master is copied.
- In case of master node failure, reads are still possible via any of the slave nodes.
- Slave node can serve as a backup for the master node.



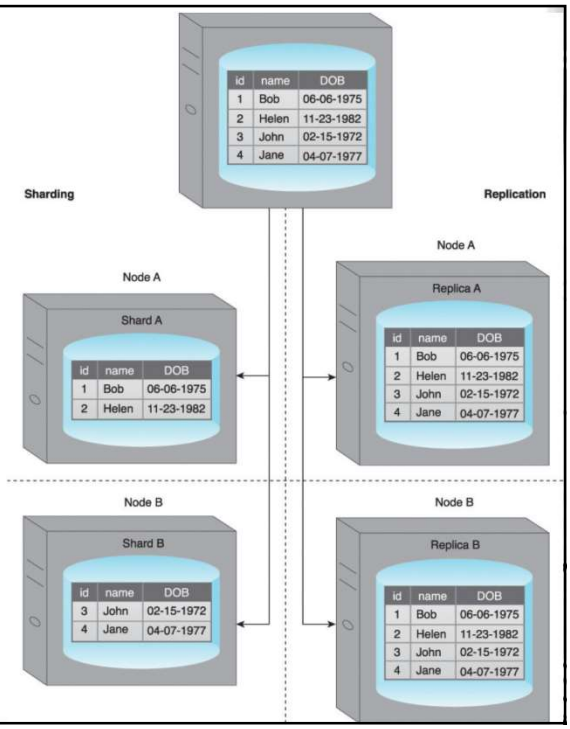
Peer-to-Peer Replication

- Replication method where all nodes operate at the same level, without a master-slave relationship.
- Each node (peer) is equally capable of handling reads and writes.
- Each write is copied to all peers simultaneously.
- Reads can be served by any peer, as each contains a copy of the data.
- Pessimistic Concurrency: Uses locking to prevent inconsistency but can impact availability.
- Optimistic Concurrency: Allows inconsistency with the expectation that consistency will be achieved after updates propagate.
- Possible due to simultaneous updates across multiple peers.
- Reads eventually become consistent when updates are executed on all peers.

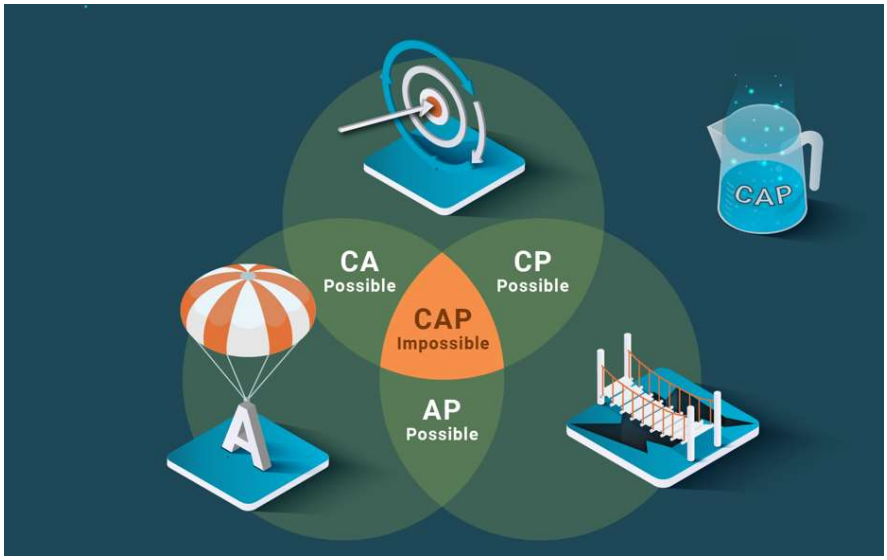


Sharding and Replication

- Combined Approach: Improves fault tolerance and scalability.
- Combinations Covered:
  - Sharding and master-slave replication
  - Sharding and peer-to-peer replication

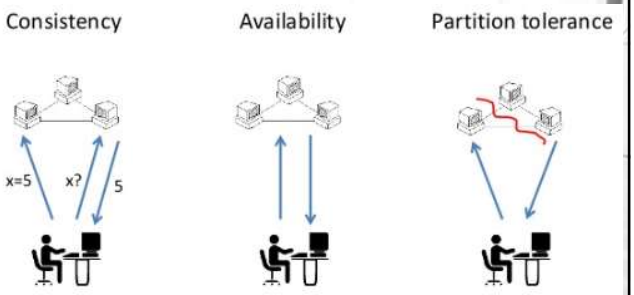


CAP Theorem



Understanding CAP Theorem

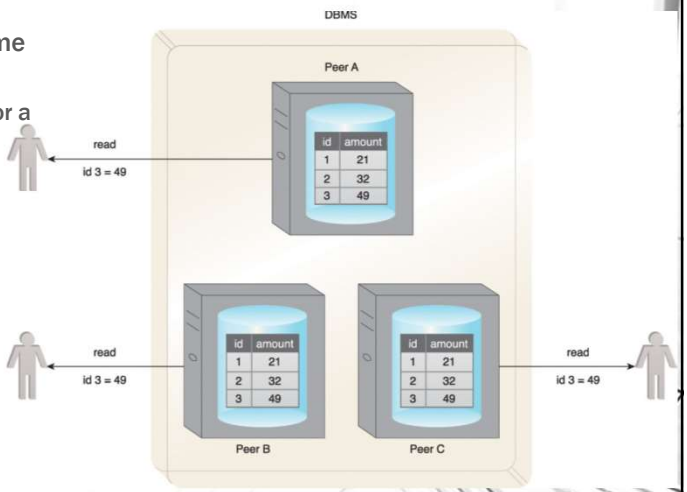
- The Consistency, Availability, and Partition tolerance (CAP) theorem, also known as Brewer's theorem, expresses a triple constraint related to distributed database systems.
- It states that a distributed database system, running on a cluster, can only provide two of the following three properties



Consistency (C)

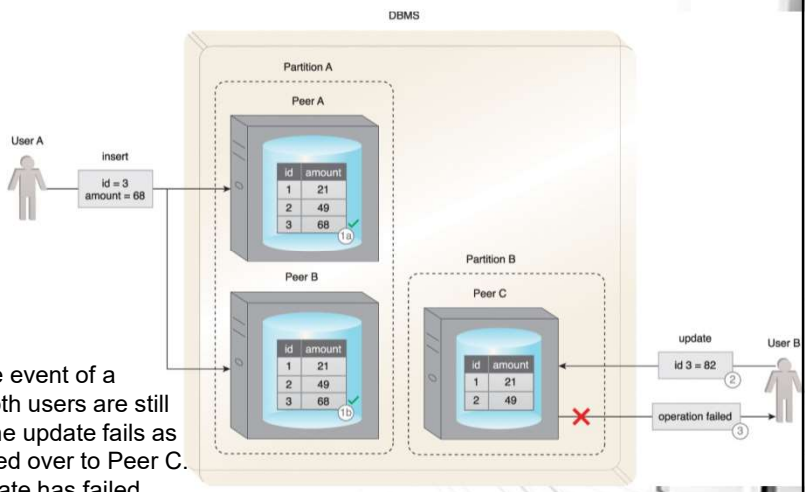
- A read from any node results in the same data across multiple nodes.
  - Example: All users get the same value for a specific data record.

All three users get the same value for the amount column even though three different nodes are serving the record.



Availability (A)

- A read/write request will always be acknowledged, either as a success or a failure.
  - Example: Even in the event of a communication failure, some users can still access the system.



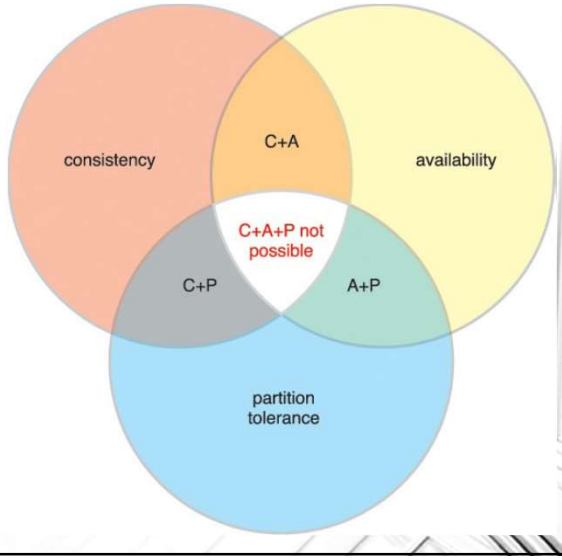
Availability and partition tolerance: in the event of a communication failure, requests from both users are still serviced (1, 2). However, with User B, the update fails as the record with id = 3 has not been copied over to Peer C. The user is duly notified (3) that the update has failed.

Partition Tolerance (P)

- The system can tolerate communication outages that split the cluster into multiple silos.
  - Example: The database can still service read/write requests during network partitions.

Exploring CAP Theorem Combinations

- **CAP Constraints:**
  - A distributed database system can only provide two out of the three properties simultaneously.



Consistency + Availability (CA)

- If consistency (C) and availability (A) are required, available nodes need to communicate to ensure consistency (C).
- Therefore, partition tolerance (P) is not possible.

Consistency and Partition Tolerance (CP)

- If consistency (C) and partition tolerance (P) are required, nodes cannot remain available (A) as the nodes will become unavailable while achieving a state of consistency (C).

Availability and Partition Tolerance (AP)

- If availability (A) and partition tolerance (P) are required, then consistency (C) is not possible because of the data communication requirement between the nodes.
- So, the database can remain available (A) but with inconsistent results.

Choosing Between CAP Options

- **Decision Factors:**
  - System requirements.
  - Trade-offs between consistency, availability, and partition tolerance.
- **System Design:**
  - Considerations for scalability, fault tolerance, and communication.

Conclusion

- Understanding storage concepts crucial for Big Data environments.
- Clusters, file systems, NoSQL, sharding, replication, and CAP theorem play key roles.
- Choose storage strategies based on specific requirements.