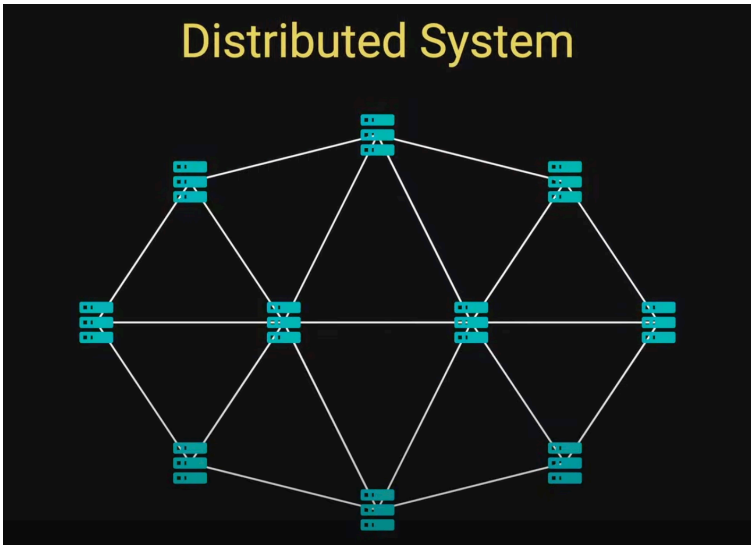
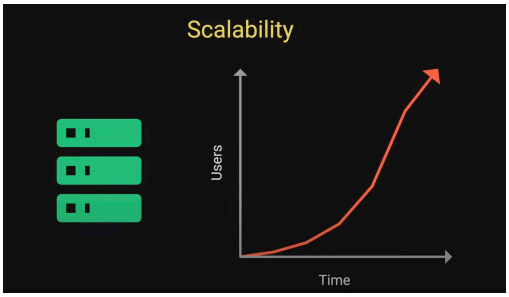
Designing a Scalable Social Media Platform

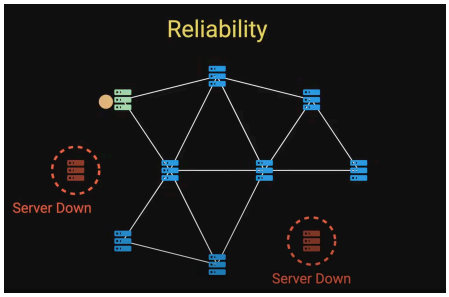
- **Starting Point :**
  - Begin with a simple web server and a single database.
  - Recognize that this setup won't scale as user demands grow.

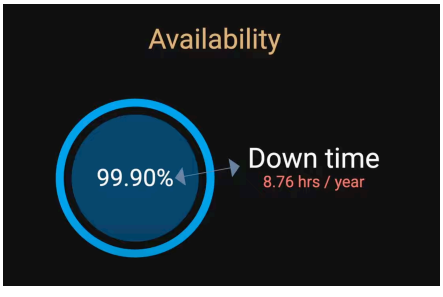
- To handle millions of user requests, a distributed system is crucial.
- When considering the design of a social media platform capable of supporting millions of users, starting with a simple web server and a single database is inadequate for scalability and performance.
- Distributed systems, which consist of networks of independent computers operating as a cohesive unit, are the recommended solution to handle the demands of a growing user base.

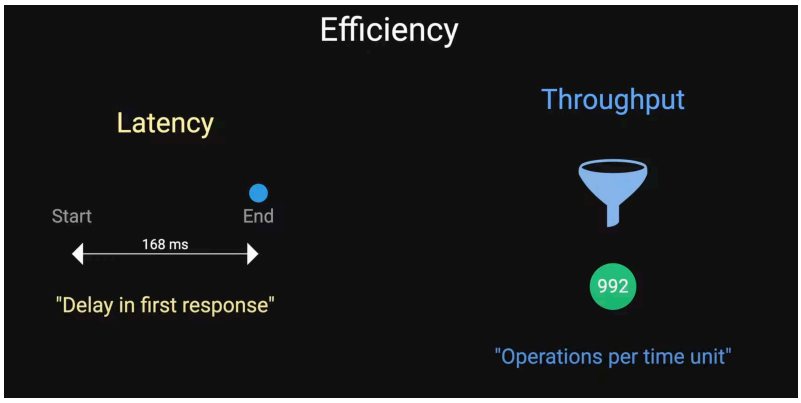


- Key characteristics of distributed systems include scalability, reliability, availability, and efficiency, each of which involves necessary trade-offs that must be balanced according to specific requirements.
  - **Scalability:** Ability to handle growth through horizontal (adding servers) or vertical (upgrading hardware) scaling.
  - **Reliability:** System continues functioning correctly despite component failures.
  - **Availability:** The operational percentage of the system, often expressed in “nines." (e.g., 99.9% availability indicates a maximum downtime of 8.76 hours per year).
  - **Efficiency:** Measured by latency (response delay) and throughput (operations handled).

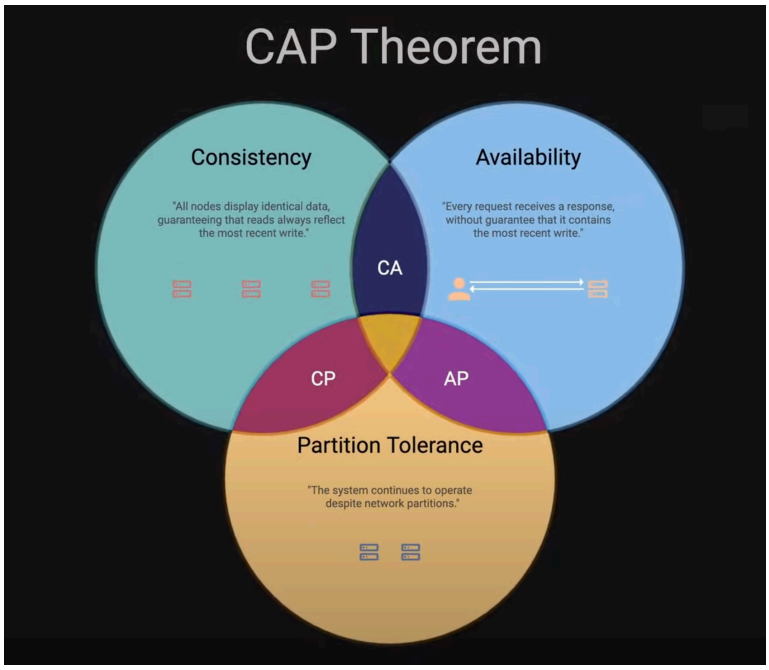




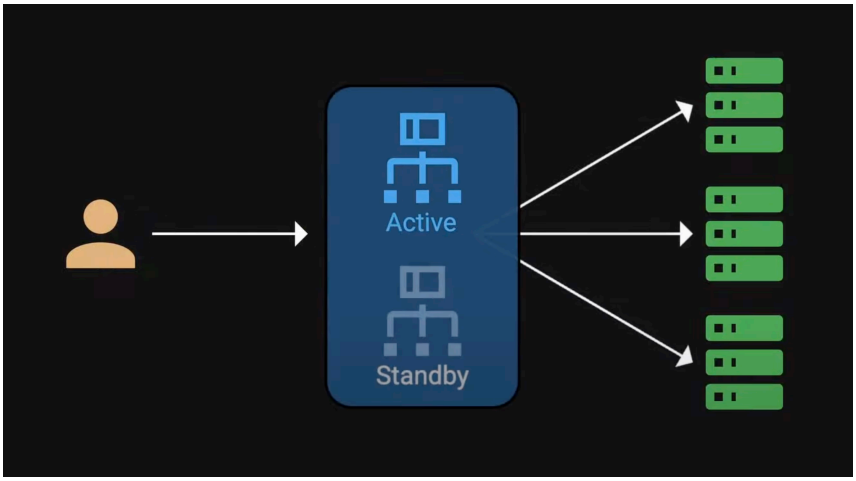




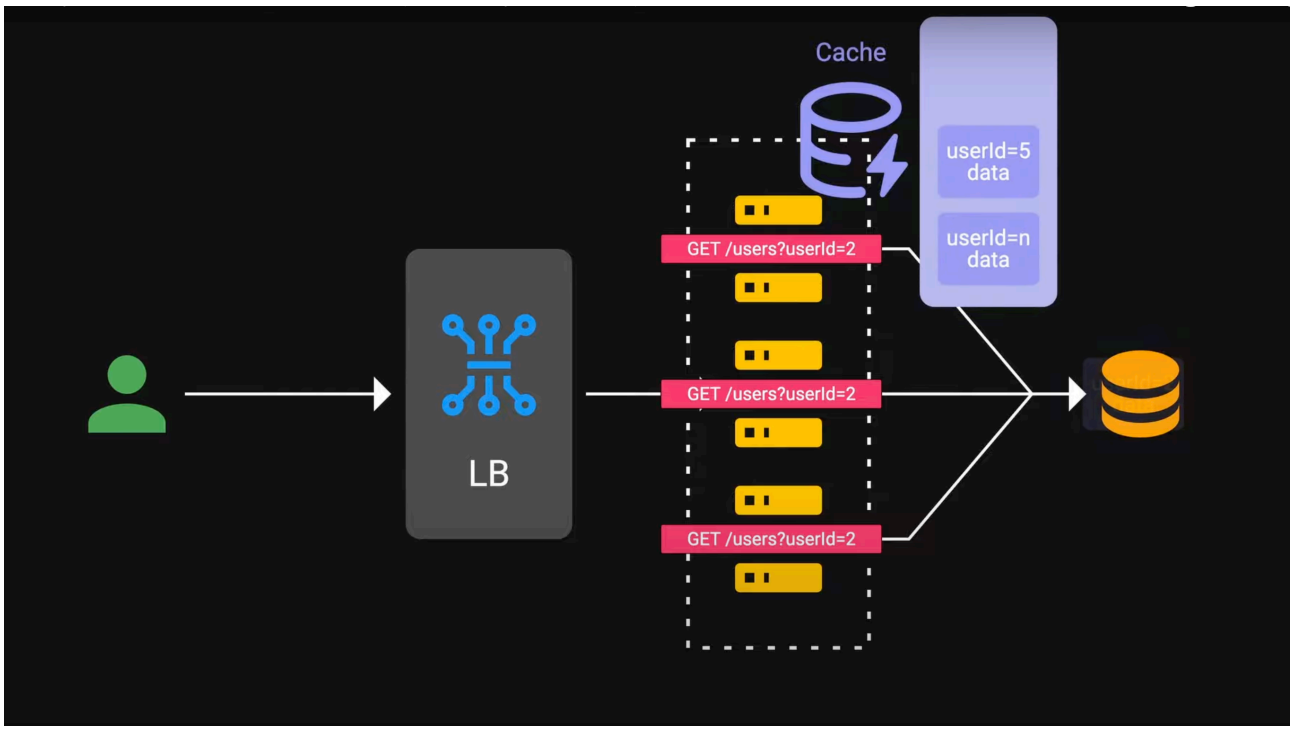
- **CAP Theorem**
  - States that a distributed system can only guarantee two out of three properties:
    - **Consistency**: All nodes display identical data.
    - **Availability**: Every request receives a response without guaranteeing the most recent data.
    - **Partition Tolerance**: System continues functioning despite network failures.



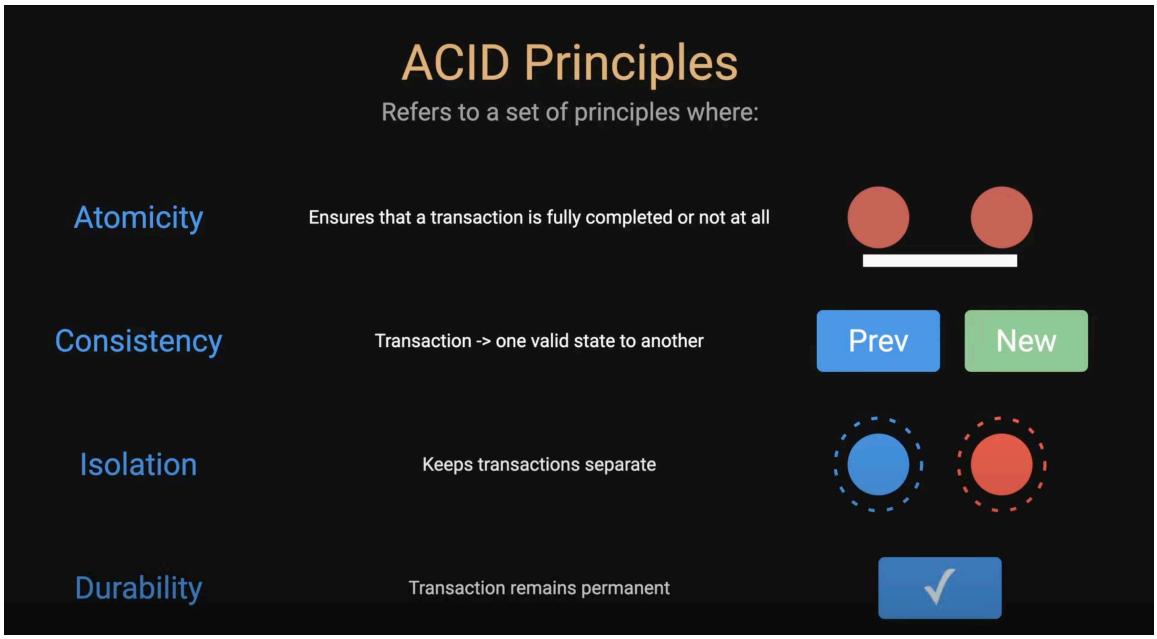
- **Load Balancing**
  - Load balancing is crucial for distributing incoming user requests across multiple servers, which prevents any single server from becoming overwhelmed with too many requests.
  - A load balancer intelligently redirects traffic to healthy servers in the event one server fails, ensuring a high level of system availability.
  - Load balancers can exist at multiple tiers: between users and web servers, between web servers and application servers, and between application servers and databases.
  - Various algorithms, such as least connection, round robin, and IP hash, are employed by load balancers to distribute requests based on specific criteria tailored to application needs.
  - It's essential to recognize that a load balancer itself can become a single point of failure, which can be mitigated by having a standby load balancer that takes over if the primary one fails.

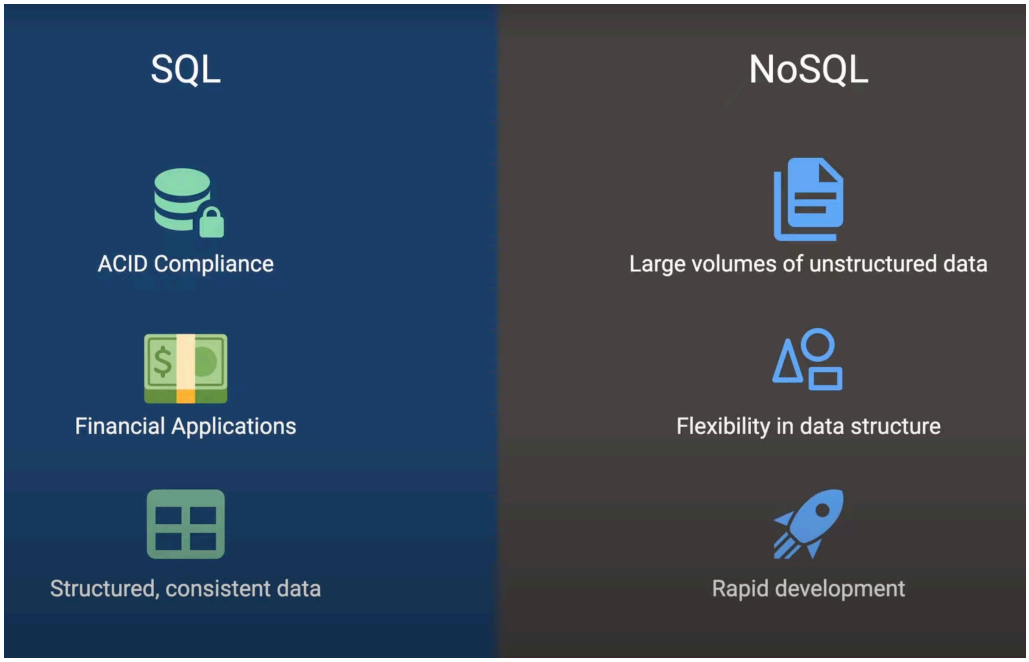


- **Caching**
  - Caching significantly enhances performance by speeding up data retrieval.
  - As distributed servers frequently request the same data from the database, caching becomes a vital strategy to improve efficiency.
  - Caching leverages the principle that data recently accessed is likely to be requested again, thus allowing for faster retrieval compared to querying the original database.
  - Static content can also be served more effectively through Content Delivery Networks (CDNs), which cache content closer to the user to reduce latency.
  - **However, challenges** arise in maintaining data consistency between the cache and the source data, necessitating effective cache in-validation strategies to prevent serving outdated information.
  - Various cache in-validation methods exist, such as write-through, write-around, and write-back strategies, each with unique benefits and risks that influence performance.
  - Cache Eviction Policy
    - LRU(Least Recently used): Removes the recently accessed data.
    - FIFO(First in First out) : Removes the oldest item first.
    - LFU(Least Frequently Used) : Removes first the least frequently used data.

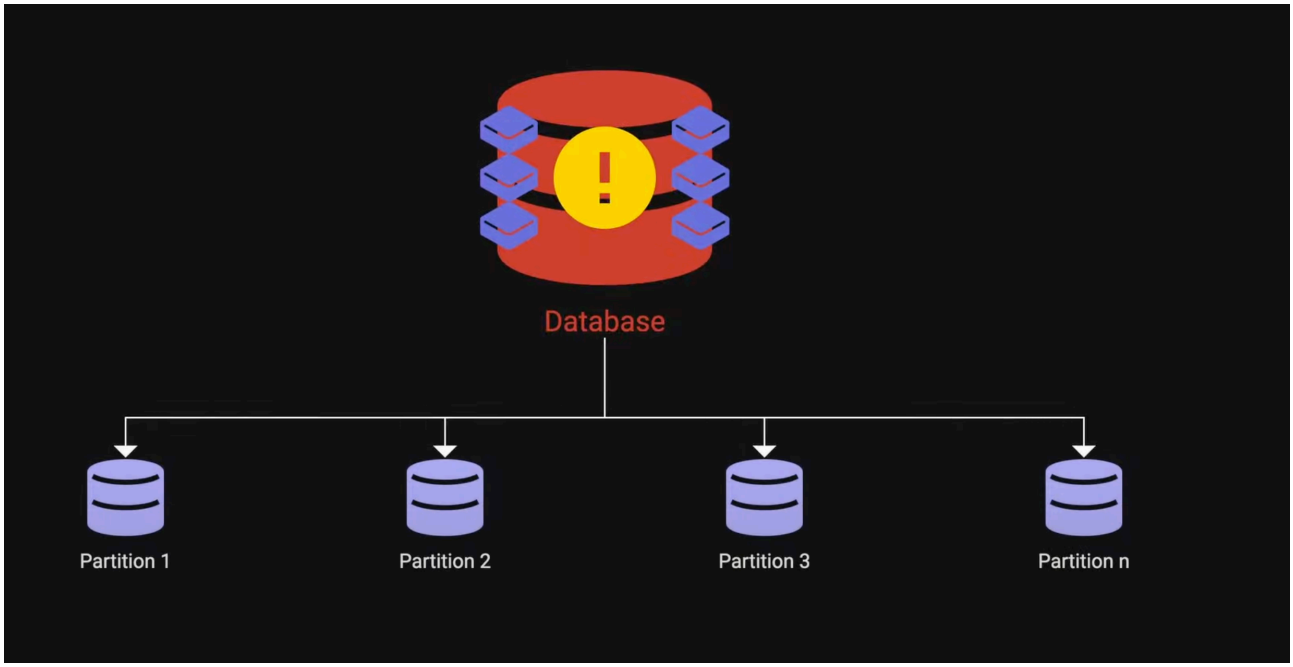


- **Database Choices**
  - When expanding the platform, a crucial decision arises regarding whether to choose a traditional SQL database or a NoSQL alternative.
  - SQL Databases: SQL databases employ a rigid schema with data stored in structured tables, requiring uniformity across records, while popular SQL databases include MySQL, Oracle, and PostgreSQL.
  - NoSQL Databases: NoSQL databases offer a more flexible structure, suitable for handling unstructured data, and encompass several types, including key-value stores, document databases, wide-column stores, and graph databases.
  - SQL databases generally scale vertically, which can be limiting, whereas NoSQL databases are designed for horizontal scalability and can be more accommodating for rapid development.
  - Reliance on ACID compliance principles is a hallmark of SQL databases, ensuring transactions are reliable, while NoSQL databases may sacrifice some of these principles for performance and scaling benefits.

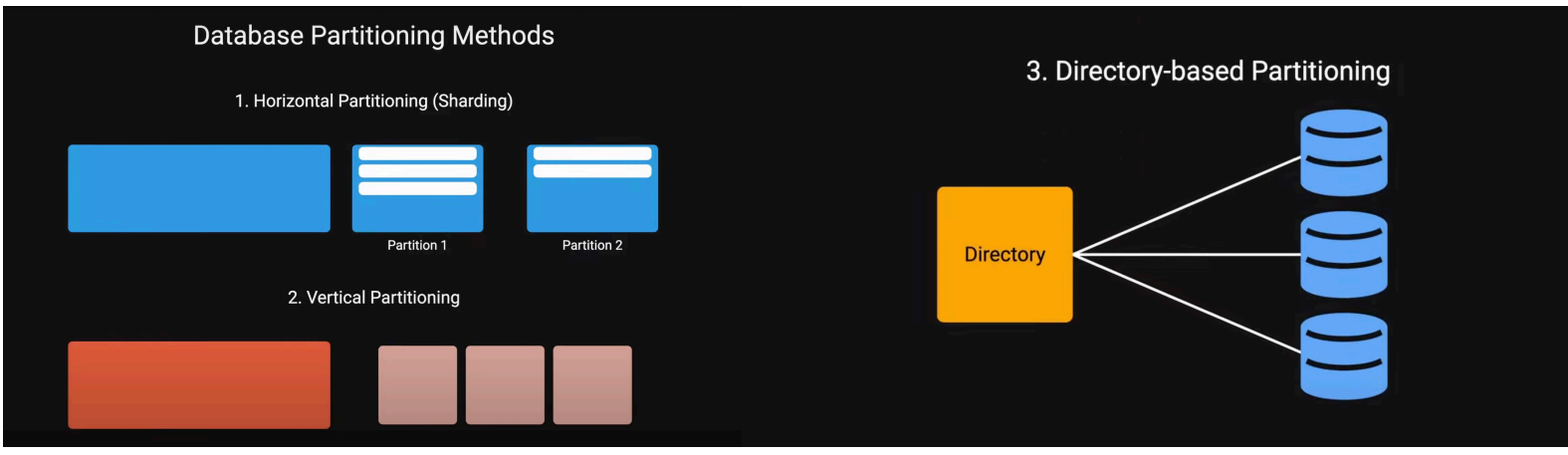




- **Indexing**
  - Improves query performance by creating separate data structures pointing to actual data.
  - Types of indexes include primary key, secondary index, composite index, and foreign key.

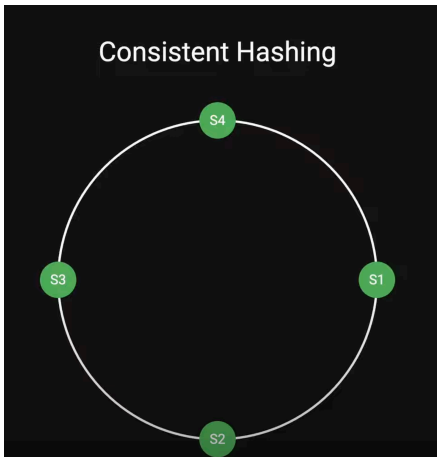


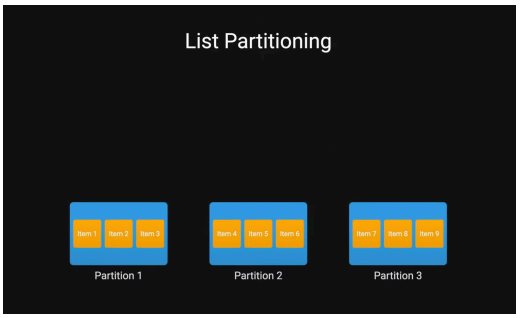
- **Data Partitioning**
  - Used when databases can no longer scale vertically.
  - Techniques include horizontal, vertical, and directory-based partitioning.
  - Challenges include complexity in joining data across partitions and data rebalancing.
  - Data partitioning is a critical technique used to distribute data evenly across different partitions, often in a circular order. This approach helps in managing large datasets effectively.
  - While partitioning can enhance performance and scalability, it can also introduce complexities, particularly in joining and querying data across multiple partitions. This difficulty may necessitate additional strategies to ensure smooth data access and manipulation.
  - The transition from a simple single server setup to a more robust scalable architecture, such as what has been implemented in social media platforms, showcases the evolution of system design. However, this process comes with its own set of challenges, including the need for effective data rebalancing to maintain efficiency.

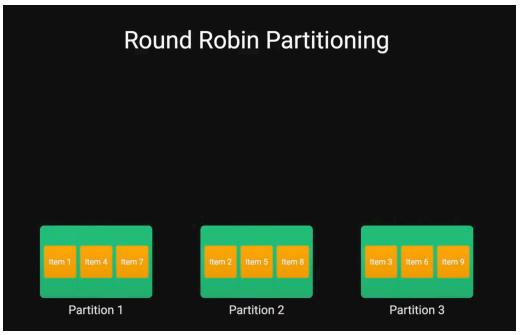


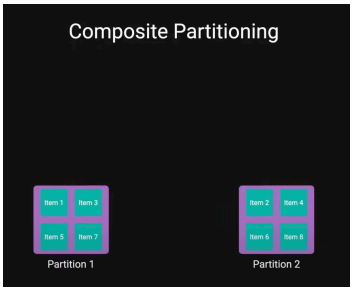
Partitioning Techniques :

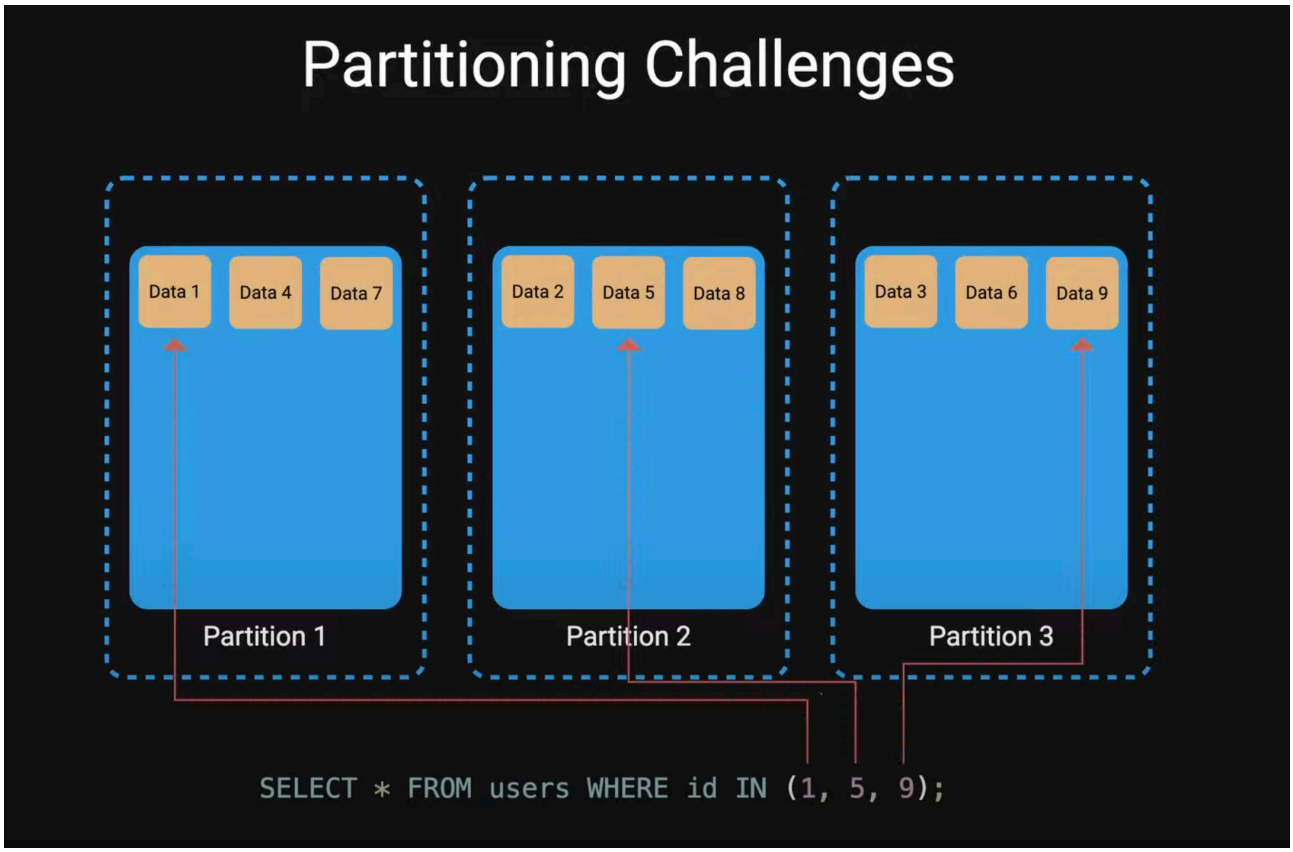
- Hash Based
- Key Based



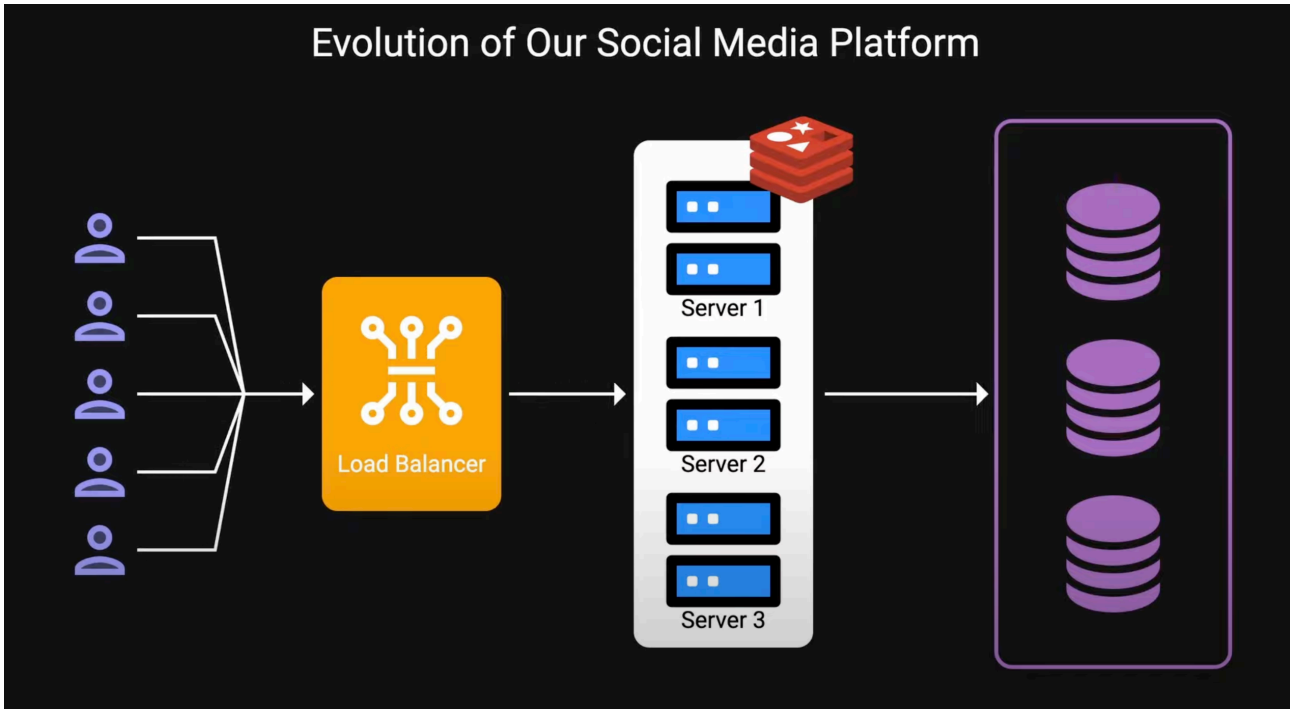








In Partitioning data, the challenges can be of data joining and re-arranging data


Evolution of Our Social Media Platform
Load Balancer
Server 1
Server 2
Server 3