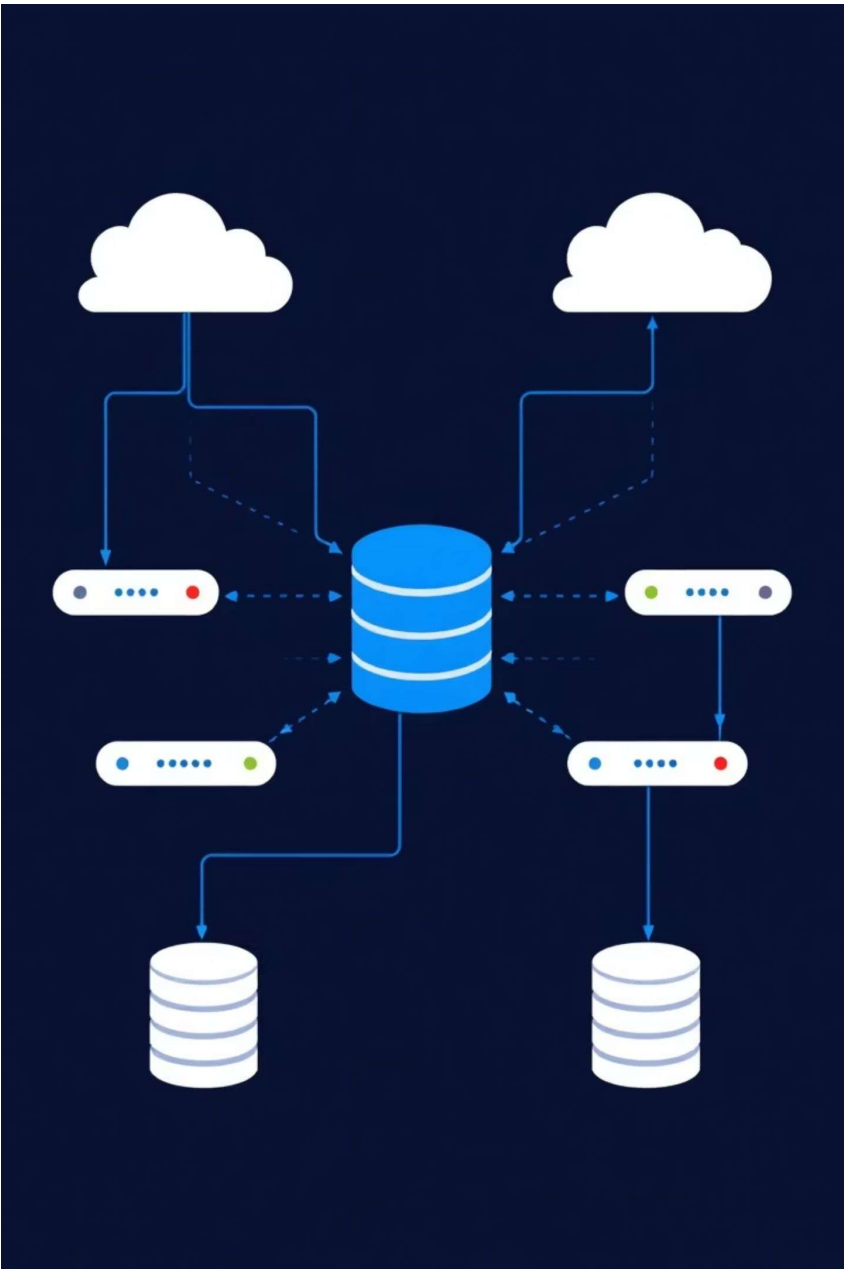


MERiT

Introduction to Distributed Databases

In modern data-driven organizations, a single monolithic database can become a bottleneck. Distributed databases spread data across multiple nodes to improve scalability, fault tolerance, and performance.

This module explores essential distributed database concepts including replication, sharding, failover, and load balancing - technologies that address the challenges of large-scale data management in today's demanding applications.

By the end of this presentation, you'll understand how these techniques improve reliability and scalability for high-demand applications.

Replication Fundamentals

Replication involves creating and maintaining copies of data on multiple database nodes. This critical strategy improves three key aspects of database systems:

Availability

Data remains accessible even when individual nodes fail

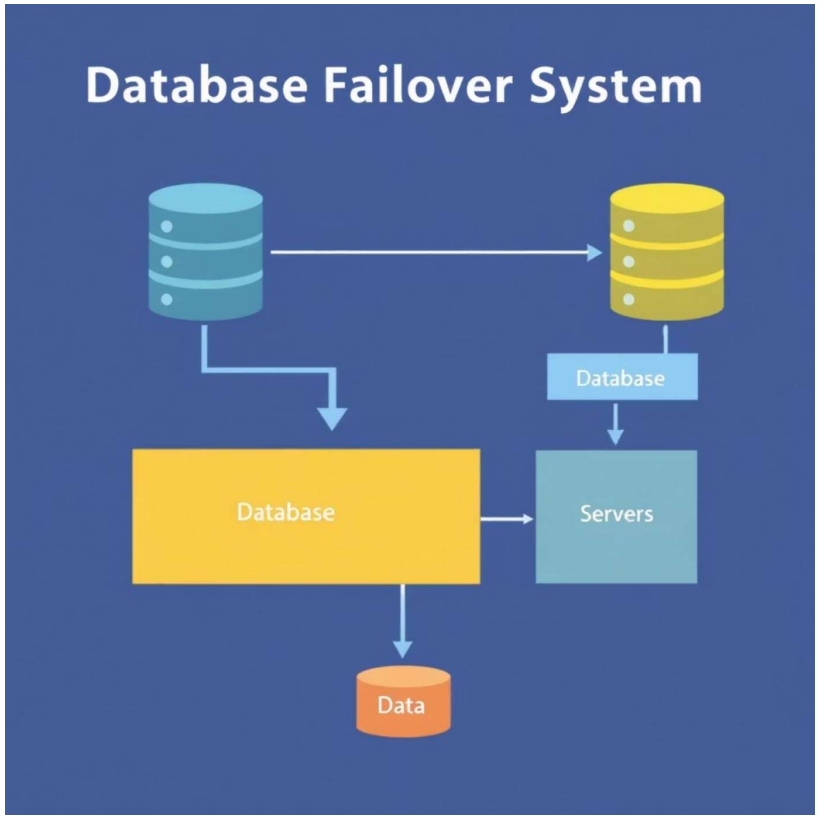
Load Distribution

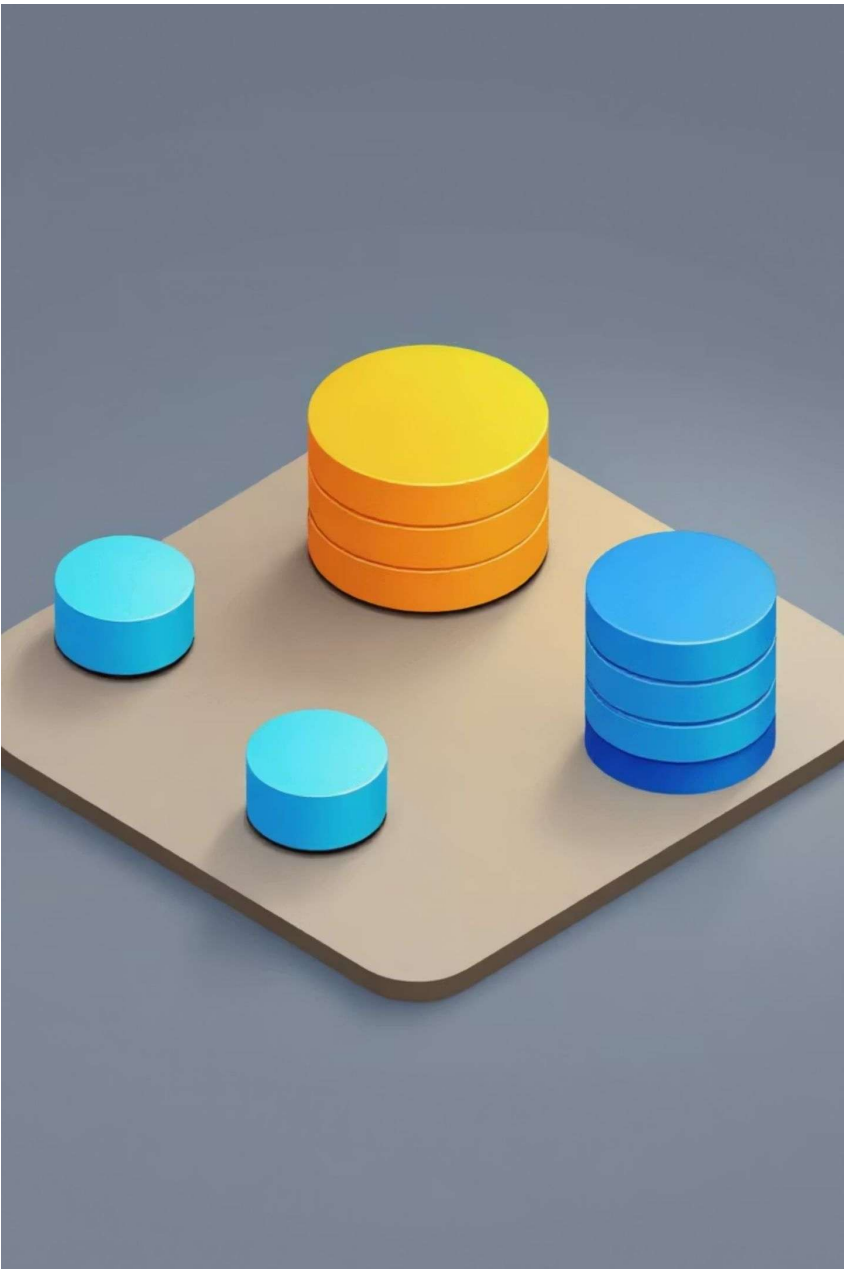
Read queries can be directed to replicas, reducing master load

Fault Tolerance

Multiple data copies provide redundancy against hardware failures

Common replication types include master-slave and multi-master configurations, each addressing different needs in a distributed system architecture.





Master-Slave Replication

MERiT

Master Node

Handles all write operations (INSERT, UPDATE, DELETE)

Changes are recorded in transaction logs

Replication Process

Transaction logs are shipped to slave nodes

Changes are applied in the same order as on master

Slave Nodes

Read-only copies of the master data

Can serve read requests to distribute load

While this approach simplifies data consistency and reduces write burden on slaves, the single master remains a critical point of failure, requiring failover mechanisms for high availability.

Multi-Master Replication

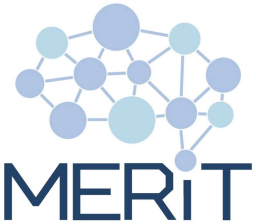
In a multi-master setup, two or more nodes can handle write operations simultaneously. Each node then synchronizes with others to maintain data consistency across the system.

This approach offers significant advantages in high-availability environments:

- Improved fault tolerance through redundant write capabilities
- Better write availability and performance through load distribution
- Geographic distribution to reduce latency for global applications

However, this flexibility comes with challenges - particularly conflict resolution when the same data is updated simultaneously on different masters.

Clustering and Sharding



Clustering

Clustering groups several database servers to act as a single, cohesive system. It typically involves:

- Coordinated node management
- Shared storage or replicated data
- Unified access interface

Clustering focuses on node coordination and redundancy while maintaining a logical single-system view.

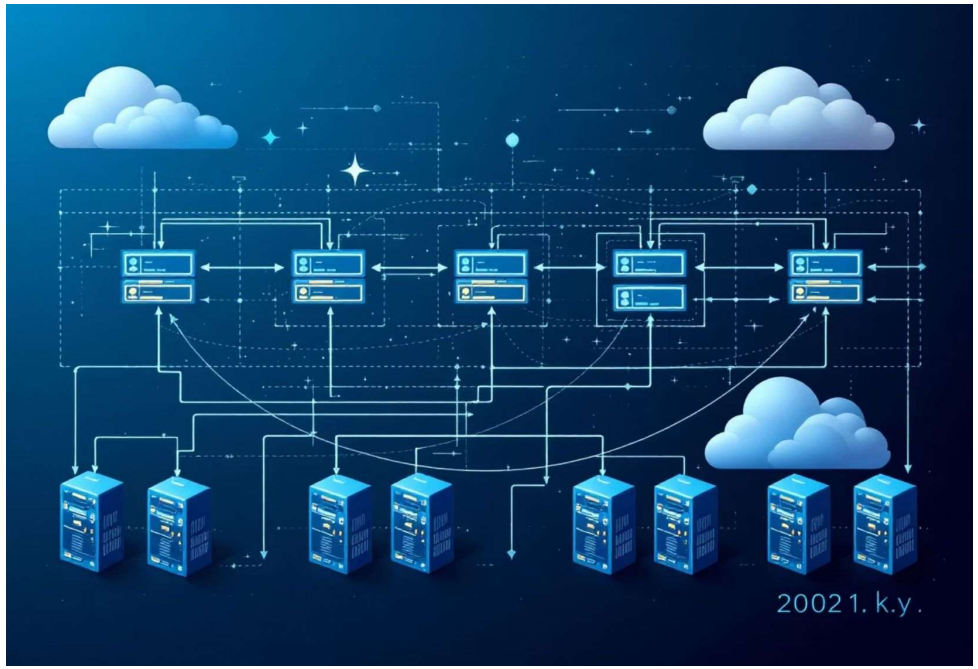
Sharding

Sharding distributes datasets across multiple nodes, with each shard storing a subset of data. Common sharding methods:

- Range-based (by key ranges)
- Hash-based (using hash functions)
- Directory-based (lookup tables)

This approach physically separates data to address the storage and performance limitations of single servers.

Techniques for Horizontal Scaling



Horizontal scaling adds more machines to handle increased traffic rather than upgrading a single server (vertical scaling). This approach offers several advantages:

Cost Efficiency

Using commodity hardware instead of expensive high-end servers

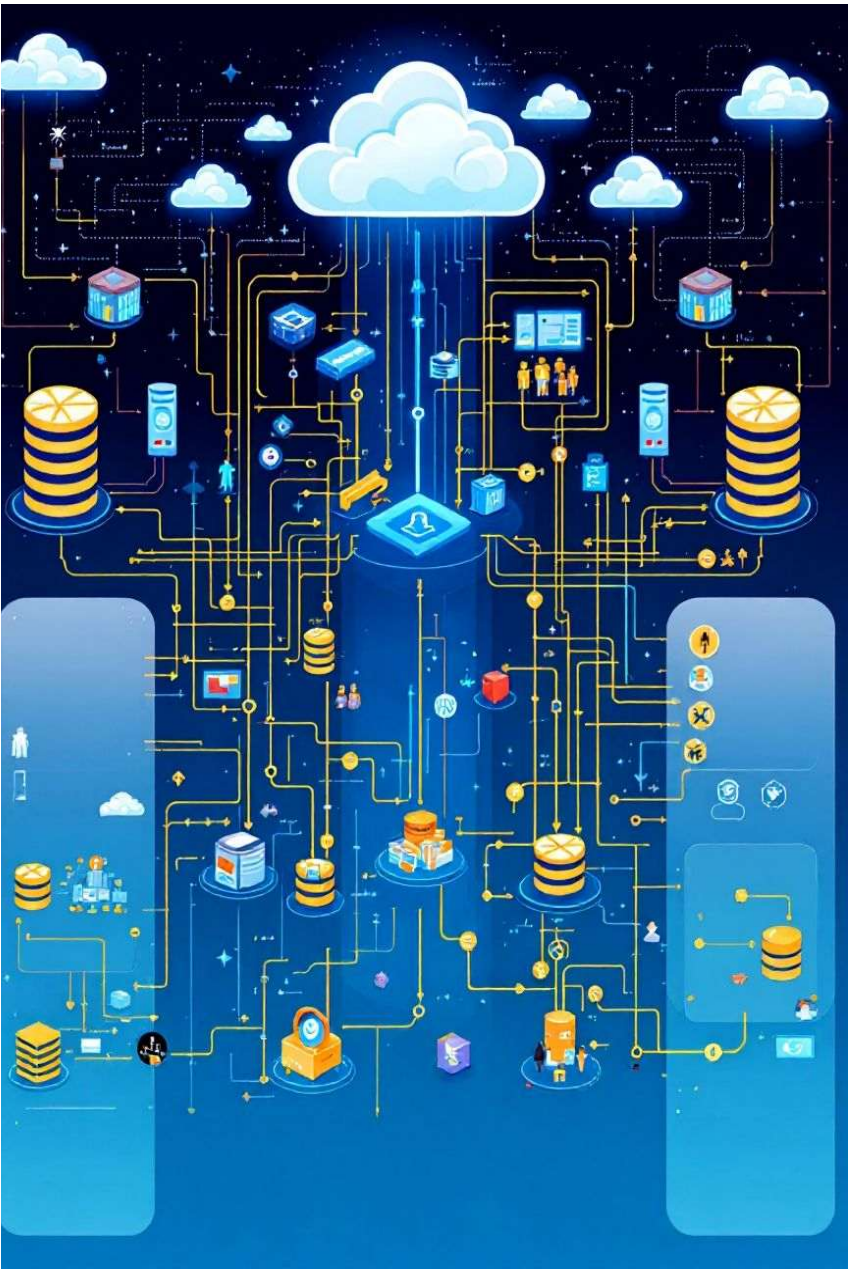
Flexibility

Scaling resources precisely as needed, adding or removing nodes

Reliability

No single point of failure when properly implemented

Combined with replication or sharding, horizontal scaling ensures growing demand is met without placing all load on a single server.



Failover and Load Balancing

MERiT

Failover

Automatically transfers control to a standby system when the primary fails

- Continuous monitoring of primary health
- Automatic promotion of replicas
- Redirection of client connections

Load Balancing

Distributes workload across multiple database nodes

- Read/write splitting
- Query routing based on node capacity
- Geographic distribution for global applications

Together, these mechanisms maintain high availability and stable performance, ensuring applications continue running smoothly even during partial system failures.

Automatic Failover and Recovery

Automatic failover solutions detect primary node failure, promote a replica, and re-route traffic without manual intervention. This process involves several critical steps:

1. **Failure Detection**
   Health checks identify when primary node becomes unavailable
2. **Consensus Building**
   Remaining nodes agree on failure confirmation
3. **Replica Promotion**
   Most suitable replica is promoted to new primary
4. **Traffic Redirection**
   Client connections redirected to new primary

Proper configuration is essential to avoid "split-brain" scenarios where multiple nodes believe they are primary, potentially leading to data inconsistency. Tools like Patroni for PostgreSQL help coordinate cluster-wide consensus.

Practical Considerations

Choosing the optimal distributed database architecture requires careful consideration of multiple factors:

Data Volume

Terabytes of data may require sharding while smaller datasets might work with simple replication

Query Patterns

Read-heavy workloads benefit from many replicas while write-intensive applications may need multi-master

Network Topology

Geographic distribution affects latency and synchronization strategies

Redundancy Requirements

Critical applications may need geographic redundancy and automatic failover

Remember that combining strategies adds complexity - ongoing monitoring and testing are crucial to avoid unexpected failures or data inconsistencies.

Summary and Next Steps

Key Concepts Covered

- Replication strategies (master-slave and multi-master)
- Clustering and sharding for data distribution
- Horizontal scaling techniques
- Failover and load balancing mechanisms

Practical Application

In the upcoming hands-on lab, you will:

- Set up a PostgreSQL replication environment using Docker
- Configure master and replica nodes
- Simulate failover scenarios
- Test database performance under various conditions

This practical experience will demonstrate the real-world benefits of distributed databases: enhanced reliability, fault tolerance, and the flexibility to scale with growing demands.