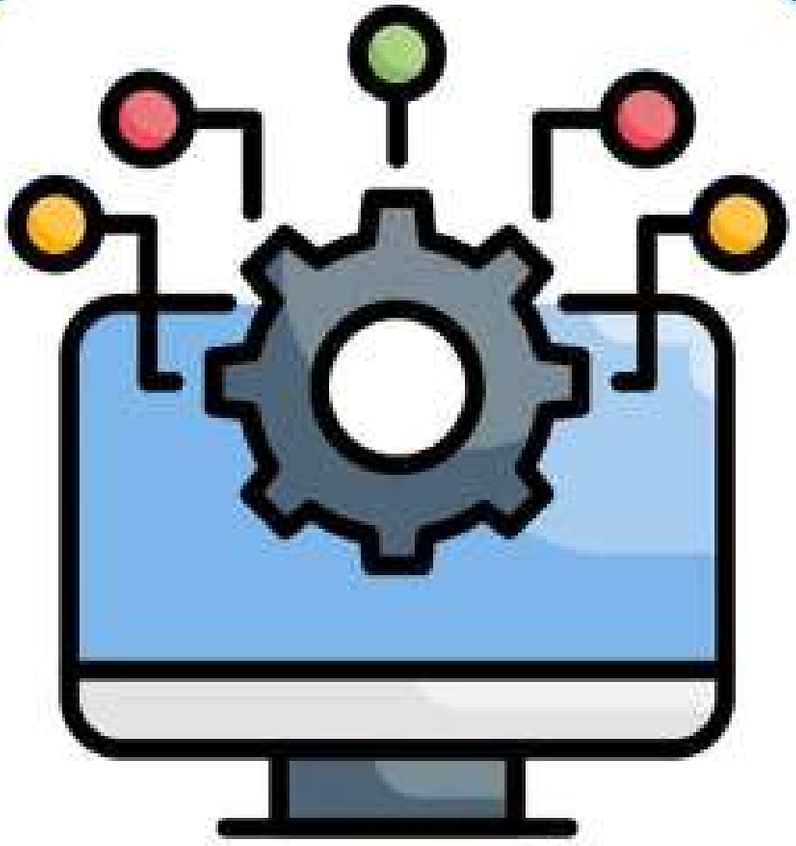
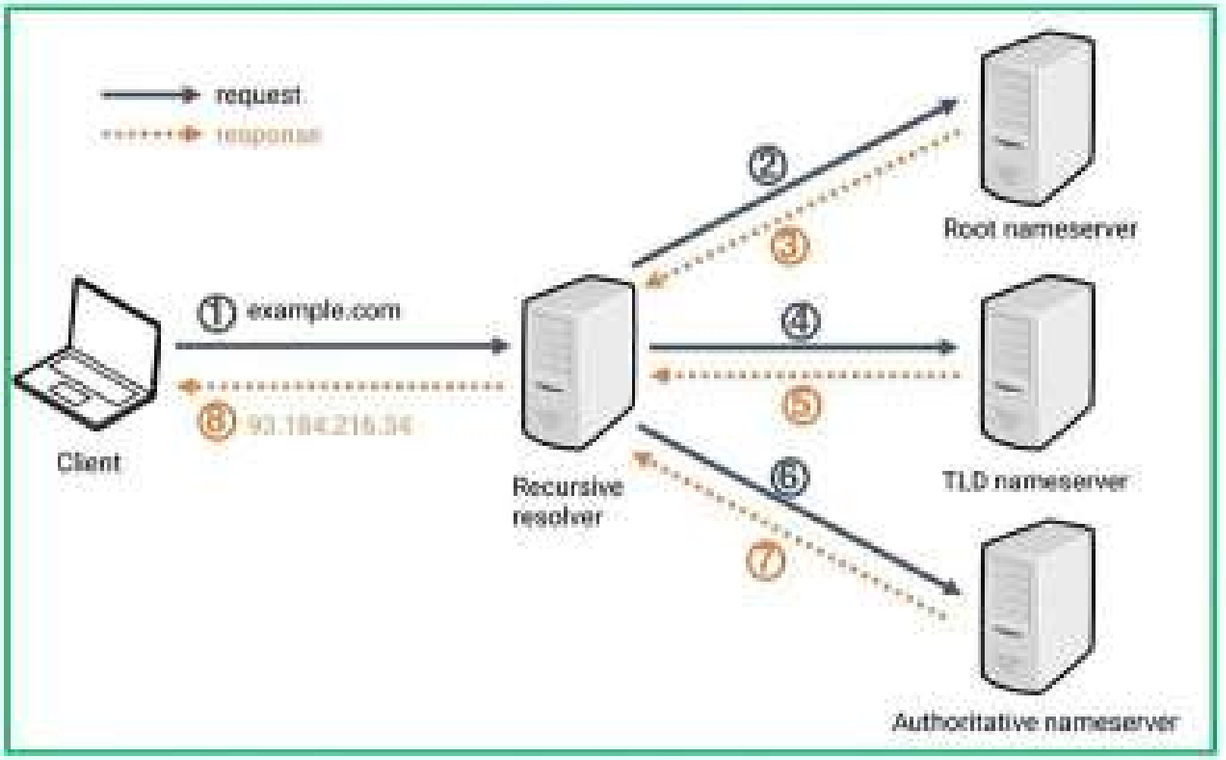
System Design Concepts

EVERY DEVELOPER SHOULD KNOW BEFORE THE INTERVIEW



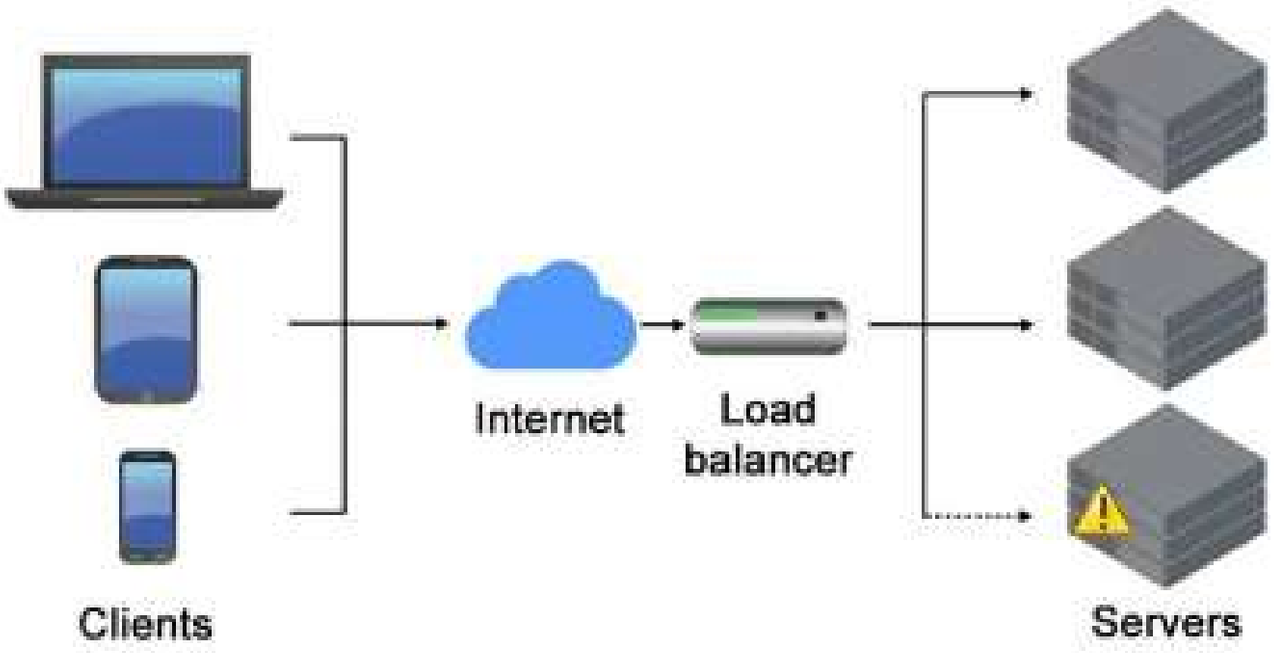
Domain Name System

(DNS)



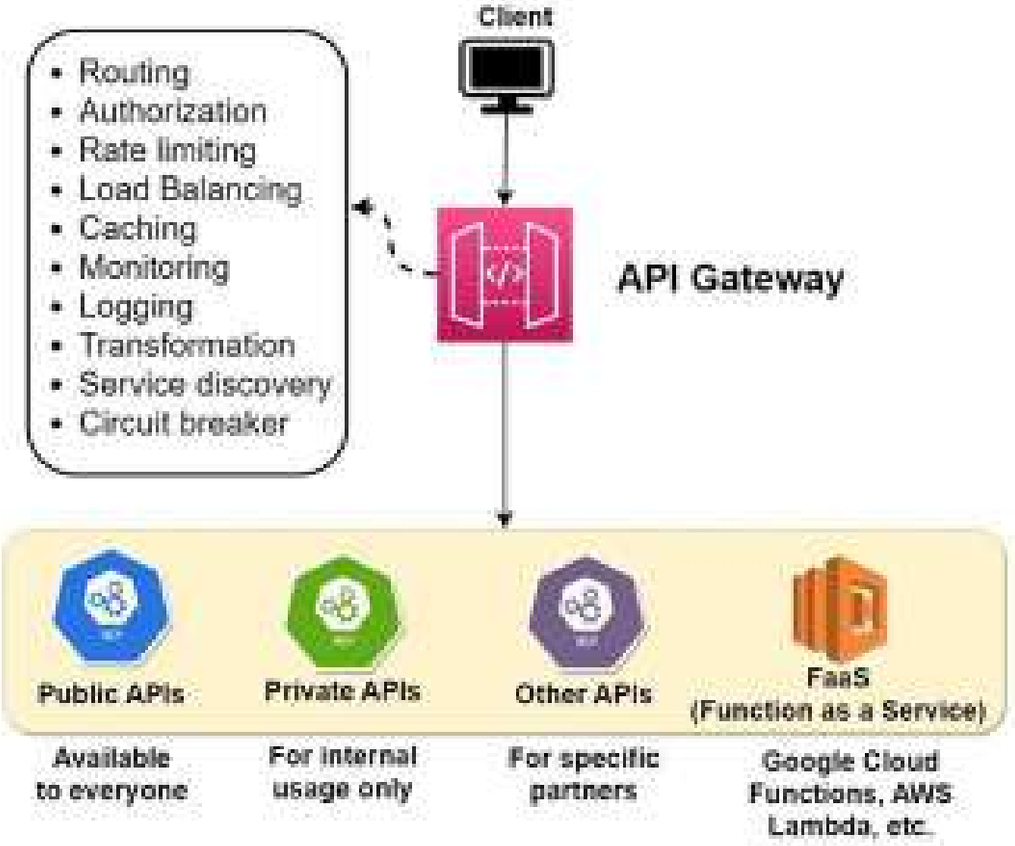
The **Domain Name System (DNS)** translates human-friendly domain names into IP addresses, acting as the internet's phonebook. It enables users to access websites by entering domain names instead of numerical IP addresses. DNS queries traverse recursive resolvers, root servers, TLD servers, and authoritative name servers.

Load Balancer



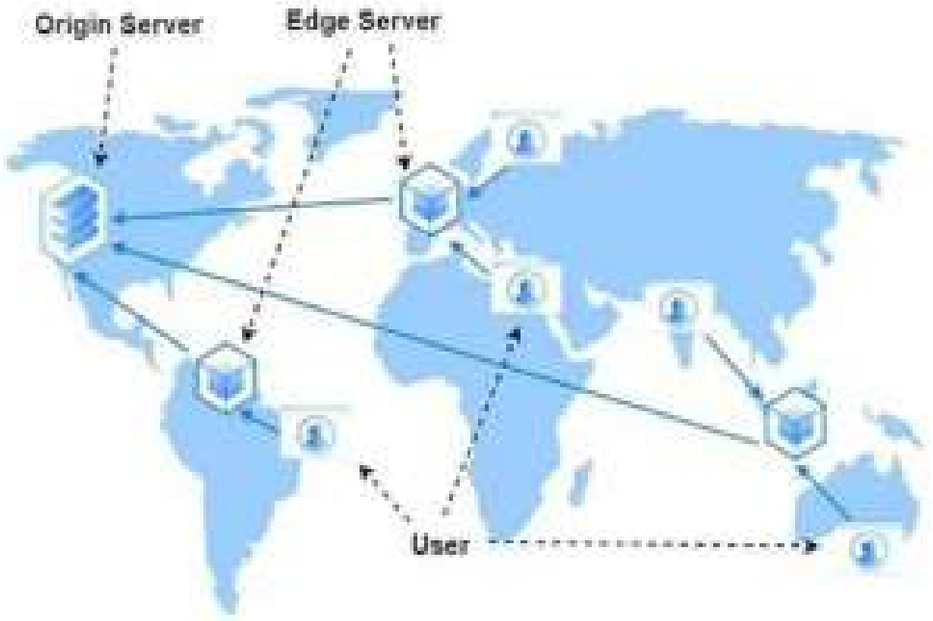
Load balancers distribute incoming network traffic across multiple servers to ensure optimal resource utilization and high availability. Common algorithms include Round Robin, Least Connections, and IP Hash. Load balancing helps manage server workloads and ensures scalability, particularly during traffic surges.

API Gateway



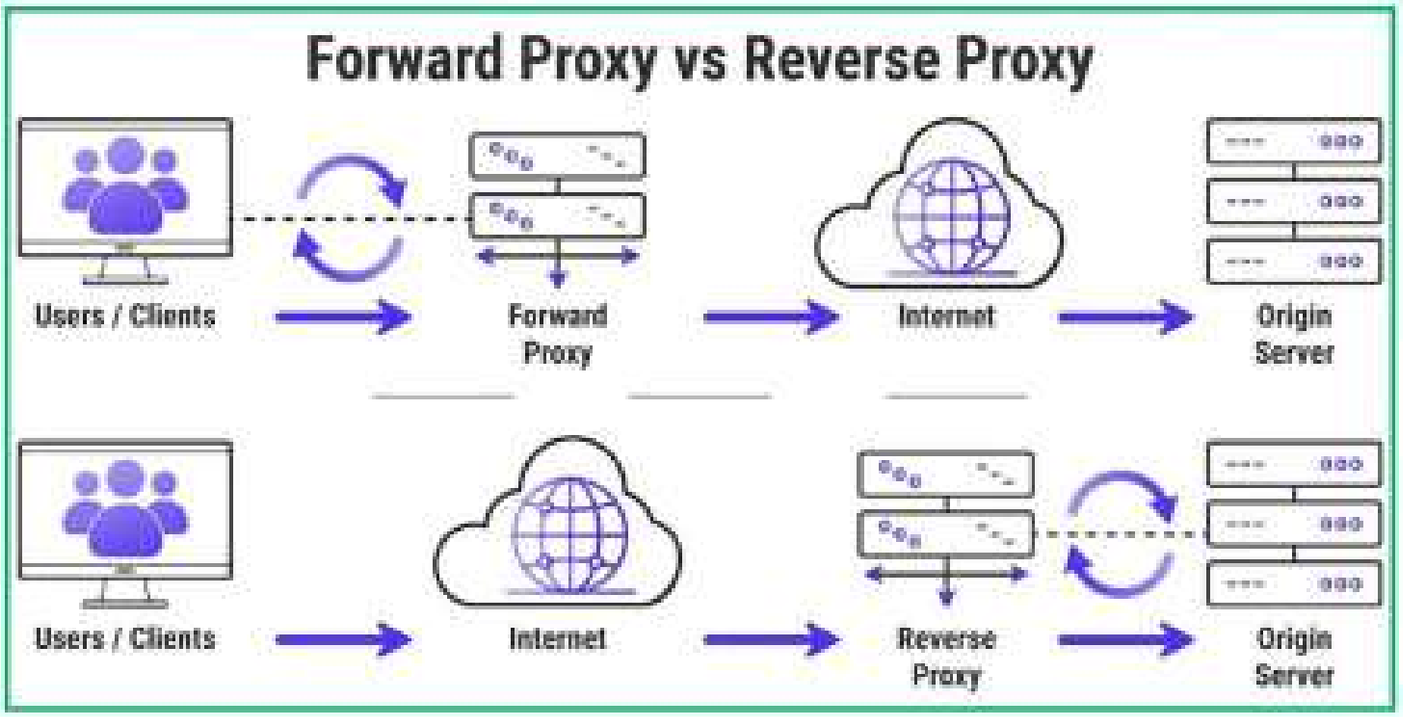
An **API Gateway** acts as an intermediary between clients and backend services, streamlining communication in microservices architectures. It manages request routing, authentication, rate limiting, caching, and request/response transformations, enhancing security and efficiency while offering a single entry point for clients.

Content Delivery Network (CDN)



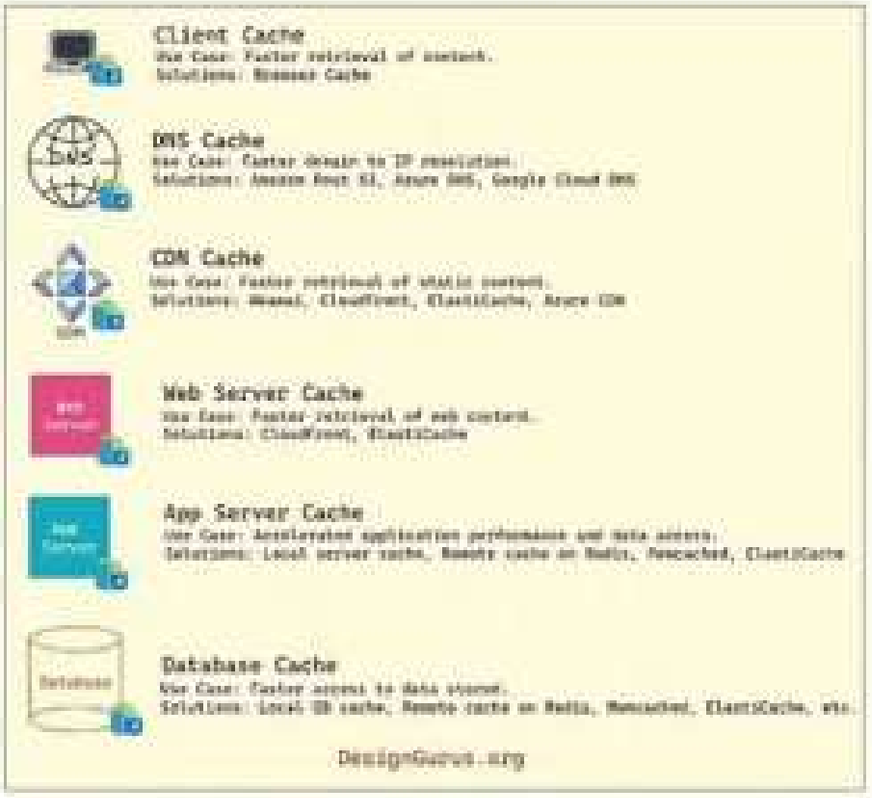
A **CDN** is a distributed network of servers that deliver content from geographically closer locations to users. It improves performance by reducing latency and ensures content availability. CDNs cache content on edge servers and periodically update it from the origin server to maintain freshness.

Forward Proxy VS Reverse Proxy



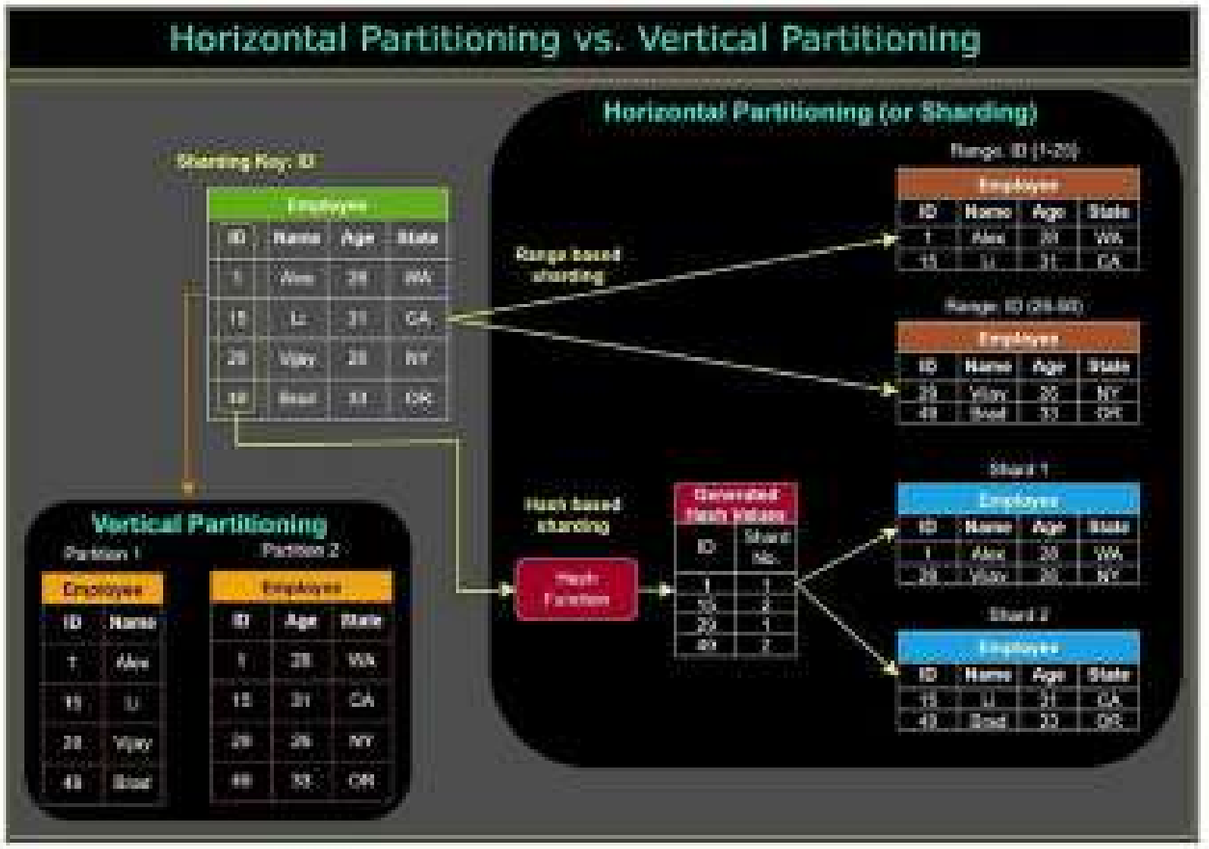
A **forward proxy** sits between client machines and the internet, forwarding client requests. A **reverse proxy** sits between web servers and the internet, forwarding client requests to servers. Both enhance security, load balancing, and caching but serve different purposes in network architecture.

Caching



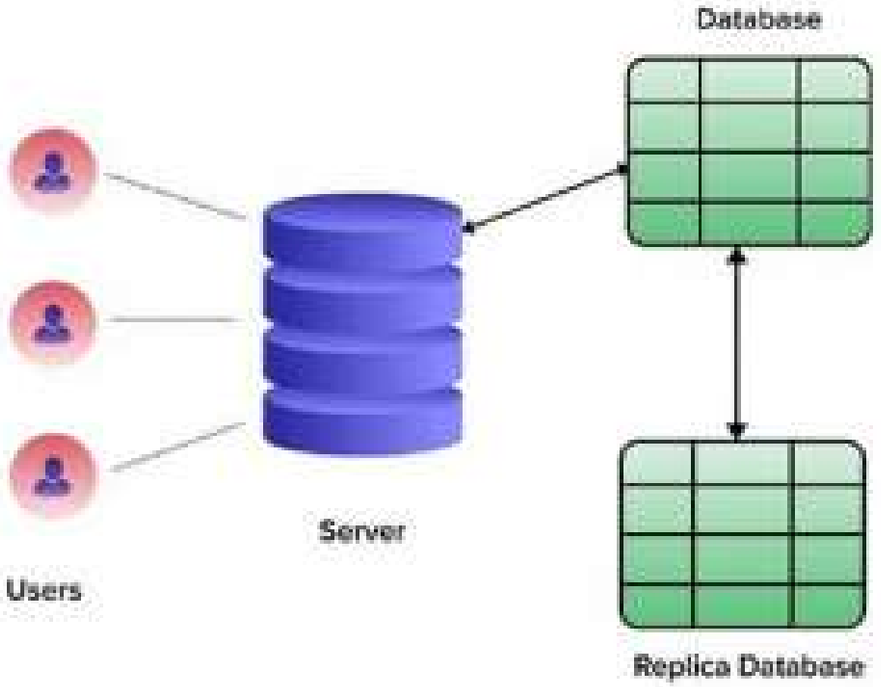
Caching stores frequently accessed data in a high-speed storage layer, reducing latency and load on the original data source. Caches can be placed at various points in a system, including clients, DNS, CDNs, load balancers, and servers, to enhance performance and efficiency.

Data Partitioning



Horizontal partitioning, or sharding, divides table rows across multiple servers to distribute database load. **Vertical partitioning** splits table columns into separate tables to improve query performance. Both techniques enhance scalability and manageability in large databases by distributing data effectively.

Database Replication



Database replication maintains multiple copies of a database across different servers for improved availability, redundancy, and performance. Primary and replica configurations ensure data consistency and fault tolerance. Replication enhances read performance and provides data protection against hardware failures.