Khaula Molapo

Database week 6

1.

In an e-commerce database, replication and partitioning play key roles in improving performance and reliability. Imagine an online store with millions of daily transactions across multiple countries. Replication ensures copies of the database are available in different regions, so if one server fails, another replica can take over, ensuring no downtime for customers. This also reduces latency because users connect to the nearest replica. Partitioning, on the other hand, helps manage massive amounts of data more efficiently. For example, the sales transactions table can be partitioned by region or by date, so queries only scan the relevant partition instead of the entire dataset. This makes operations like "find all orders in Africa last month" much faster. Together, replication provides fault tolerance and global availability, while partitioning speeds up queries and balances load. In such a system, customers experience faster responses, and the company handles large-scale growth smoothly.

2.

The CAP theorem states that in a distributed database, it is impossible to guarantee all three properties—Consistency, Availability, and Partition tolerance—at the same time. Systems must choose two out of three, depending on their priorities. For example, a banking system may prioritize Consistency and Partition tolerance, sacrificing some Availability, while social media platforms may prioritize Availability and Partition tolerance, accepting some temporary inconsistency. CAP helps developers design systems with realistic trade-offs. It is a guiding principle for distributed databases such as Cassandra, MongoDB, or DynamoDB, where decisions about performance and reliability depend on which guarantees are most important.

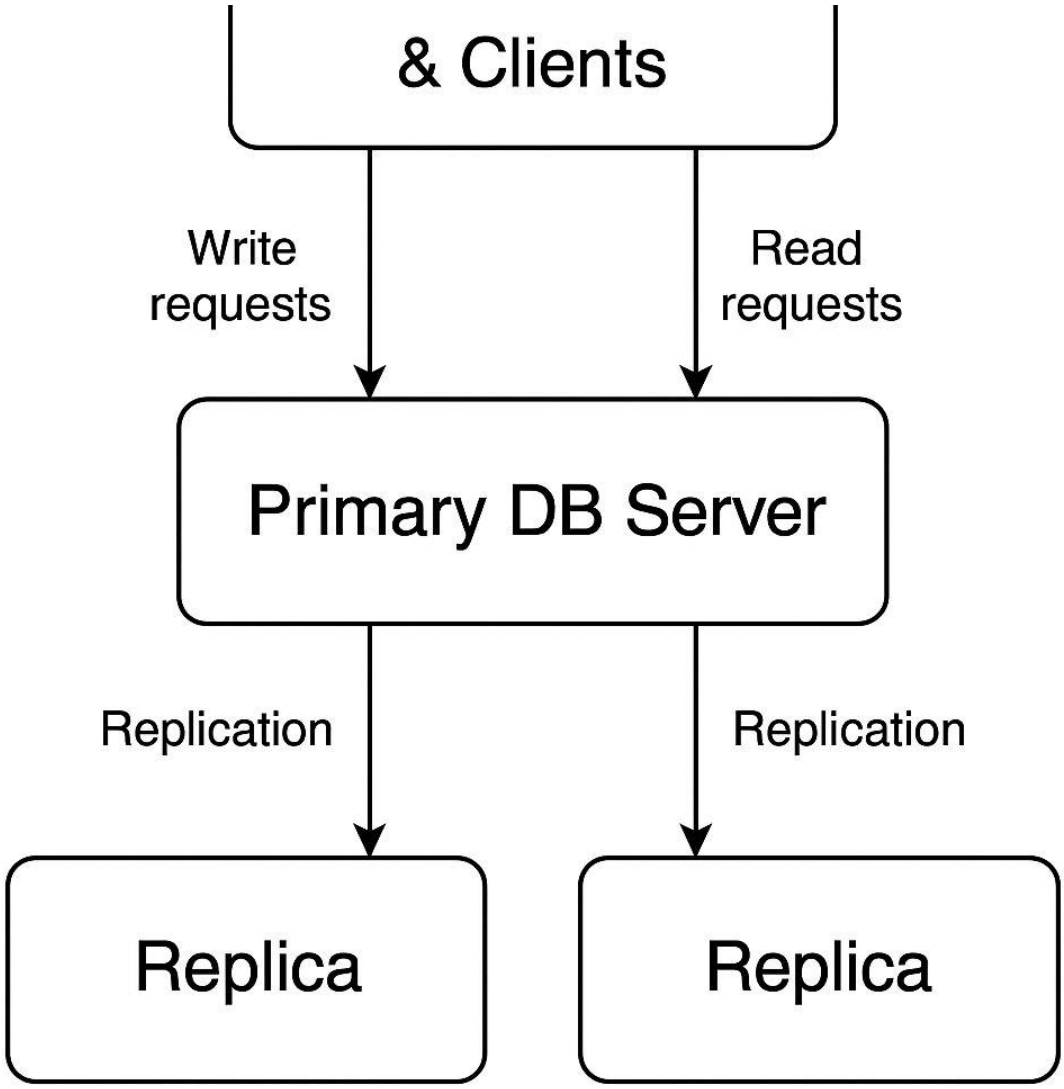
3.

In PostgreSQL, I practiced partitioning a table that stores order records. I created a parent table called "orders" and then defined child partitions based on order dates (monthly). The

PARTITION BY RANGE feature allowed me to divide data into smaller chunks, such as orders for January, February, and so on. When inserting data, PostgreSQL automatically placed rows in the correct partition. Queries that filtered by date only scanned the relevant partition, reducing query time. This exercise helped me see how partitioning improves efficiency for large datasets. It also simplified data management because partitions could be archived individually.

4.

In Draw.io, I created a replication flowchart showing a primary database server and two secondary replicas. The diagram has arrows from the primary server pointing to the replicas, representing continuous replication of transactions. Clients send write requests only to the primary server, while read requests can be directed to replicas to balance the workload. If the primary fails, one replica can be promoted to act as the new primary. The flowchart illustrates how replication improves fault tolerance and read scalability. This visual tool makes it easier to explain the process of database replication to both technical and non-technical audiences.


& Clients
Write
requests
Read
requests
Primary DB Server
Replication
Replication
Replica
Replica