Advanced Database – Week 5

Khaula Molapo

20240001

Advanced Database

Week: 5

1.

A distributed database in a banking system allows different branches of a bank to share and manage data across multiple locations. For example, a bank may have branches in Johannesburg, Cape Town, and Durban. Each branch has its own server that stores customer account information locally. However, all servers are connected through a network so they can share data when needed.

If a customer from Johannesburg withdraws money in Cape Town, the system must access the same account information. The distributed database ensures that the balance is updated correctly in all locations. This improves speed because users connect to the nearest server, and it also improves reliability because if one server fails, others can still operate.

Security is very important in this system. Data must be encrypted and access must be controlled. Distributed databases help banks provide fast services, reduce downtime, and maintain accurate financial records across all branches.

2.

Two-Phase Commit (2PC) is a protocol used in distributed databases to make sure transactions are completed safely across multiple servers. It ensures that either all systems commit (save) the transaction, or none of them do. This prevents data inconsistency.

2PC works in two stages. In Phase One, called the prepare phase, the coordinator server asks all participating servers if they are ready to commit the transaction. Each server replies with either yes or no. In Phase Two, if all servers agree, the coordinator tells them to commit the transaction. If any server says no, the coordinator tells all servers to roll back (cancel) the transaction.

This process ensures data integrity and consistency in distributed systems, especially in banking and financial systems.

3.

For this distributed banking system, I would use horizontal partitioning based on branch location. This means that customer data is divided according to the branch where the account was opened. For example, Johannesburg customers' data is stored on the Johannesburg server, Cape Town customers' data on the Cape Town server, and so on.

This method reduces network traffic because most transactions happen at the customer's local branch. It also improves performance because each server handles fewer records.

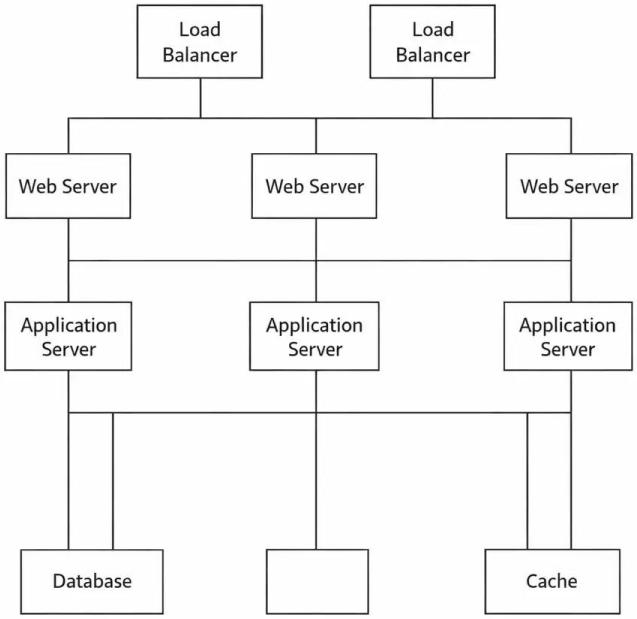
However, if a customer moves or frequently uses different branches, the system must still access remote data, which may cause slight delays. Overall, horizontal partitioning by location is simple, efficient, and suitable for large distributed banking systems.

4.

In Draw.io, I created a simple distributed architecture diagram using basic rectangles. At the top, I placed a Main Coordinator Server. Below it, I added three boxes labeled Johannesburg Branch Server, Cape Town Branch Server, and Durban Branch Server. Each branch server connects to its own local database.

All branch servers are connected to the main coordinator using straight lines to show communication. The coordinator manages transactions and controls the Two-Phase Commit process.

This design shows how data is stored at different locations but still controlled centrally for consistency. It also shows fault tolerance, because if one branch server fails, the others can continue operating. The diagram is simple but clearly explains how a distributed database system works in a banking environment


Load Balancer
Load Balancer
Web Server
Web Server
Web Server
Application Server
Application Server
Application Server
Database
Cache

.