Khaula Molapo

20240001

Advanced Database

Week 2

1

A global retailer uses a distributed database to manage data from stores in different countries. The database is spread across many servers located in various regions to improve speed, reliability, and availability. Partitioning is used to divide data into smaller parts based on regions or store locations, so each server handles only a portion of the data. This helps reduce delays and improves performance. Replication is used to copy data across multiple servers, so if one server fails, another can still provide the data. This also helps users access information faster from nearby servers. By using both partitioning and replication, the retailer ensures that its systems are scalable, reliable, and able to handle large amounts of data from customers, products, and transactions across the world.

2

The CAP theorem states that a distributed system cannot guarantee Consistency, Availability, and Partition Tolerance at the same time. Consistency means all users see the same data at the same time. Availability means the system always responds to requests. Partition tolerance means the system continues to work even when network failures occur. In real systems, designers must choose which two properties to prioritize. For example, in a banking system, consistency and partition tolerance are often prioritized over availability to ensure accurate account balances, even during network issues.

3

Apache Cassandra is a distributed NoSQL database designed for high availability and scalability. After installing Cassandra using Docker or a local setup, a sample table can be created to store customer or product data. Partitioning is done by choosing a partition key, such as customer ID or region, which determines how data is distributed across nodes. Cassandra automatically spreads data across multiple servers to improve performance and

reliability. This makes it suitable for large-scale applications that need fast access to data and can handle massive workloads without downtime.

4

Strong consistency ensures that all users always see the latest version of data immediately after an update. This is important for systems like financial or medical databases where accuracy is critical. Eventual consistency allows temporary differences in data between servers, but all copies become consistent over time. This model is often used in IoT databases because devices generate large amounts of data and need fast responses. Strong consistency provides accuracy but may be slower, while eventual consistency offers better performance and scalability but may show slightly outdated data for a short time.