10/24

PAGE NO. 1
DATE / /

Assignment - 1

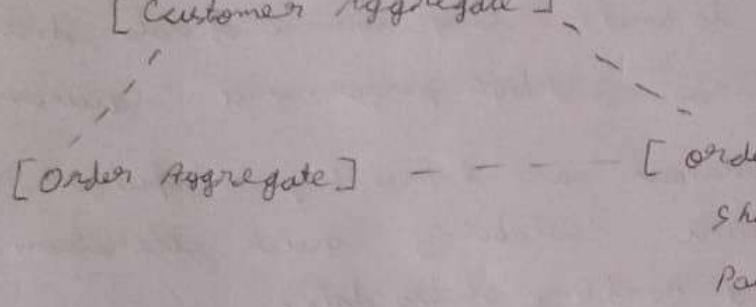
1) What is NO SQL ? Explain briefly about aggregate data models with a neat diagram ?

No SQL databases are non relational databases designed to handle large volume of data, distributed data storage and high performance requirements. NO SQL databases are especially useful in applications that require scalability, quick iteration and efficient handling of big data.

Aggregate data models organize data into "aggregates" which are collections of data treated as single units. Aggregates simplify data distribution across clusters in NO SQL databases as the data often needs to be moved or processed together. For example, an "Order" in an e Commerce system could include customer details, ordered items and payment information within one aggregate.

PAGE NO.: 2
DATE: / /

This model is especially helpful for distributed data systems because each aggregate can manage as a single unit, which supports easier data partitioning and replication.



each aggregate can be stored and managed independently, making it suitable for large distributed databases where data consistency is crucial.

(2) Explain briefly about impedance mismatch with a neat diagram?

Impedance mismatch refers to the conflict that arises due to the difference in how data is represented in object-oriented programming (OOP) and relational databases.

Data is structured as complex objects, often containing nested attributes and hierarchical relationships, while relational databases store data in tables with rows and columns with each table holding flat, tabular data.

The impedance mismatch occurs because:-

1. Objects in code have attributes that can reference other objects, creating complex nested data structure, but relational databases require data to be normalised and stored across multiple tables

2. Converting these objects to tables

requires an Object - Relational mapping layer, which can lead to performance and consistency issues

[Object - Oriented Structure] <---> [Relational Structure]

ORM Layer

(e.g., Hibernate)

In this diagram:

- The Object - Oriented Structure represents complex nested relationships, which are more natural in applications.
- The Relational Structure represents normalized data in tables, which are better suited for storage in relational databases.
- The ORM layer serves as a bridge, translating between Objects in code and rows/columns in the database, although it may introduce overhead and complexity

PAGE NO: 5
DATE: / /

3) write a short note on a) Consequences of Aggregate Orientation b) Key value data model c) Document data model d) Column family stores e) Graph database

a) Consequences of Aggregate Orientation

- Simplified Distribution :- Aggregates are well-suited for distribution because they are self contained units of data, making it easier to partition data across nodes in a cluster.
- Data Duplication :- Storing related data together can lead to a data duplication if the same data is a part of multiple aggregates.
- Consistency Challenges :- Since aggregates are designed to operate as a single units.

b) Key Value Data model

- Description :- In the key-value model data is stored as a collection of key-value pairs where each key is unique and maps directly to a specific value.

Advantages :- Fast lookups by key and flexible data storage as values can be any data type.

c) Document Data model

- Description :- Document databases store data in documents, typically using formats like Json, Bson or XML

- Advantages :- Supports semi structured data and nested data, making it adaptable for changing data requirements

d) Column Family Stores

- Description :- Column family stores organize data in columns and rows similar to relational tables but more flexible schema definition. Data is stored in column families

- Advantages :- Optimized for queries on specific columns and allows for efficient storage of sparse data.

e) Graph Database

- Description :- Graph databases use nodes, edges and properties to represent and store data, making them ideal for applications with complex relationships

PAGE NO.: 7
DATE: / /

Advantages :- Efficiently handles highly connected data and complex queries involving relati

4 What are Schemaless databases ? Explain in details?

Schemaless Database also known as schema-free are databases that do not requires a predefined schema to define the structure of data.

Unlike traditional relational databases, which require a rigid schema with specified tables, Columns and datatypes.

Characteristics of Schemaless Databases

1. Flexible Data Structure :- Schemaless databases allow each record to have a unique structure meaning new fields can be added to documents without altering existing data or requiring database migrations

2. Rapid development and Iteration :- Developers can modify and evolve the database schema directly in the application code, which accelerates development, particularly in Agile environments where requirements change frequently.

Advantages of Schema less Databases

1.) Adaptability :- As application requirements change, developers can add, remove, or modify fields without restructuring the database allowing quick adaptation.

2) Efficiency for modern Applications
many modern applications handle large volume of data that do not fit well into rigid schemas.

3.) Speed of Deployment :- without the need for database migrations, updates to data models can be deployed faster, reducing downtime and accelerating feature delivery

5 what are distribution models? Explain two paths of data distribution!

Refers to strategies used to distribute data across multiple nodes or servers within a network. Distribution improves performance scalability, availability and fault tolerance in large scale applications. In distributed database systems, data can be managed across several servers, ensuring that if one node fails, others can continue to provide access to data.

Two path of Data Distribution

1. Republication :-

- Description - Replication involves copying data across multiple nodes, ensuring that data is available on more than one server. This approach improves data availability and fault tolerance because if one server fails the data still be accessed from another server

PAGE NO.: 10
DATE: / /

[2] Sharding:

- Description: Sharding is a form of horizontal partitioning where data is divided into smaller, distinct parts (called shards), each shard is stored on a separate server.

Each server (node) holds only a subset of the data rather than a copy of the entire dataset

Benefits: Sharding allows for distributed write operations as each shard can handle write request independently. This helps with scalability because as data grows, new nodes can added to store additional shards

PAGE NO. 11
DATE : / /

Q) Write a short notes on a) Single Server
b) Combining Sharding and Replication
distribution data model

a) Single server

- Description :- In a single server setup, all data resides on a single machine. This is the simplest database configuration where the server handles all read and write operations

- Advantages :- A single server setup is easy to set up, manage and reason about, making it ideal for applications with minim data storage and access needs. It is also eliminates the complexity of distributed systems, such as network latency and data consistency.

b) Combining Sharding and replication Distribution Data model.

Description :- Combining and replication involves partitioning the data into multiple shards and replicating each shard across several nodes

ensuring that each shard has multiple copies for redundancy.

- Advantages:
  - Scalability: Sharding allows the system to handle large datasets and distribute write operations across nodes, reducing the load on any single node.
  - Fault Tolerance: Replication ensures data redundancy, so if one node fails, the data can still be accessed from another replica of the shard
  - High Availability: By replicating shards, read and write operations can continue even if some nodes are down, increasing the overall availability of the database.

7 Explain about Update and Read Consistency with an example?

Update and Read Consistency in Distributed Databases

In distributed databases, consistency ensures that data is synchronized across multiple nodes so that users see the same data regardless of where or when they access it.

1. Update consistency:

Direction: Update Consistency refers to the guarantee that when data is updated in one location, this update will be propagated consistently across all nodes where the data is replicated. This helps avoid conflicts, &

Example:

Suppose two users, Alice and Bob, try to reserve the last available room in a hotel at the same time. Without update consistency both users might see the room as available and attempt to book it simultaneously, resulting in a conflict.

2. Read Consistency :-

• Description :- Read Consistency, also known as read-your-writes consistency ensures that when a user writes or updates data, they can immediately see the change upon reading it.

• Example :- If Alice posts a comment on a blog and then refreshes the page, read consistency ensures she immediately sees her comment, even if it's stored on multiple nodes. Without read consistency, Alice might not see her own comment immediately, which can be confusing.

PAGE NO.: 15
DATE: / /

(8) Explain about CAP Theorem

The CAP Theorem also known as Brewer's Theorem was formulated by computer scientist Eric Brewer and states that is impossible for a distributed data system to achieve all three of the following properties simultaneously :-

1. Consistency :
   - Consistency in CAP terms means that every read operations returns the most recent write or an error. Essentially, all nodes in the system see the same data at the same time.
   - For example, if Alice updates her profile photo in a social network application, a consistent system would ensure that anyone viewing Alice's profile profile immediately sees the updated photo.

2. Availability :-
   - Availability means that every request receives a response, even if some nodes are down. In an available system the system always responds to read and write requests.

- For example, if a node in a distributed system fails, an available system would route the request through it may not reflect the most recent data

③ Partition Technique :

- Partition tolerance implies that the system continues to operate even if there is a loss of communication between nodes. A partition-tolerant system handles scenarios where nodes can't communicate with each other by ensuring that the system continues functioning despite network failure

- For instance, if there is a temporary network partition between two data centers, the systems should continue operating independently on each side of the partition.