Times asked: 5 times
4 times
3 times
2 times
1 time

# indicates 5-mark question

Papers considered 2019 May & Dec. 2022 Dec 2023 May & Dec

Modules 4,5,6 were not the same in 2019 so questions asked 3 times in those 3 modules mean they have been asked in all the papers.

ADBMS Question bank

Module 1: Distributed DBMS

1. Explain Distributed DBMS Architecture in detail
2. Explain Data Fragmentation in Distributed Databases
3. Explain Allocation and Replication Techniques for Distributed Database Design
4. Write a short note on: Failures in Distributed Database

Module 2: Query Processing and Optimization

5. Explain phases in Distributed Query Processing with neat diagram.
6. Draw the neat diagram of Query Processing
7. Explain 2PC in detail with neat diagram
8. Explain 3PC in detail with neat diagram
9. Explain ACID properties
10. Write a short note on: Query Evaluation Plan
11. Write a short note on: Query Processing Issues in Heterogeneous Database
12. Differentiate between 2PC and 3PC Protocol

Module 3: JSON, XML and Semi-Structured Data

13. Explain basic JSON syntax with data types
14. What is XML? Explain XML Schema Document with example.
15. Create a XML document. PYQ:
    XML document of 'Restaurant Menu Card' has food items categorized into Starters, Drinks, Chinese, South and Punjabi. Each food item element contains name, cost, calories, and veg/non-veg flag. Write XML Schema for the above XML document.
16. Differentiate between XML and JSON

Module 4: NoSQL

17. Explain CAP theorem in NoSQL Database
18. Differentiate between SQL and NoSQL
19. Write a short note on: NoSQL Data Modelling
20. Explain Replication and Sharding in NoSQL
21. Explain Benefits of NoSQL

Module 5: NoSQL using MongoDB

22. Explain MongoDB Sharding
23. Explain MongoDB CRUD Operation
24. Explain basic data types in MongoDB

Module 6: Trends in advance databases

25. What is a Graph Database? What are the features of Graph Database?
26. Explain Spatial Database in detail
27. Explain Temporal Database in detail

	1	2	3	4	5	6
2023 Dec	20	25	20	30	15	15
2023 May	0	35	20	25	20	25
2022 Dec	20	30	30	30	25	5
Last 3 Avg	20	30	20	25	20	15

ADBMS Answer bank

Module 1: Distributed DBMS

1. Explain Distributed DBMS Architecture in detail

DDBMS architectures are generally developed depending on three parameters

Distribution – It states the physical distribution of data across the different sites.

Autonomy – It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.

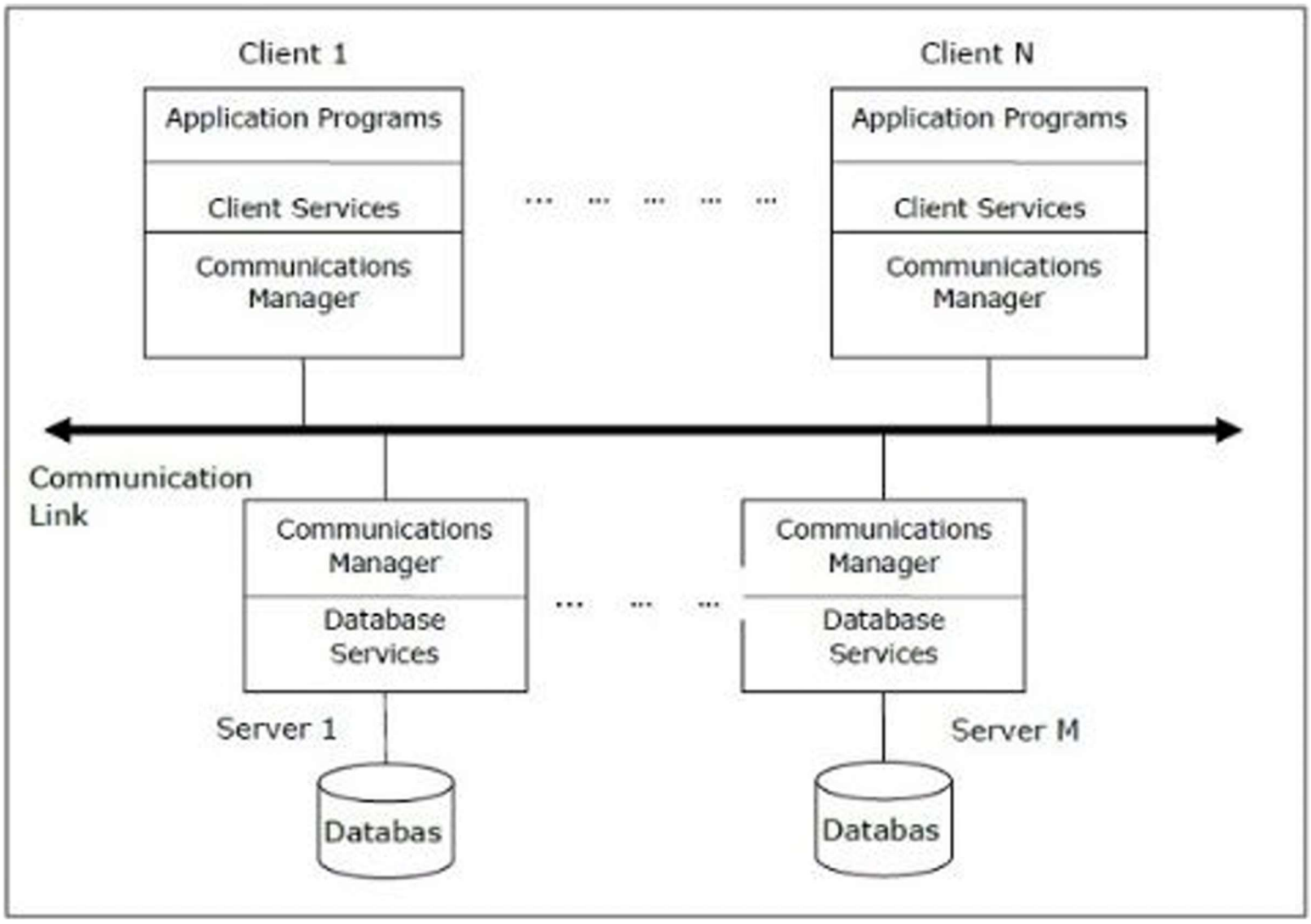
Heterogeneity – It refers to the uniformity or dissimilarity of the data models, system components and databases.

Common architectural models are:-

Client-server architecture: In this architecture, clients connect to a central server, which manages the distributed database system. The server is responsible for coordinating transactions, managing data storage, and providing access control.

Types:

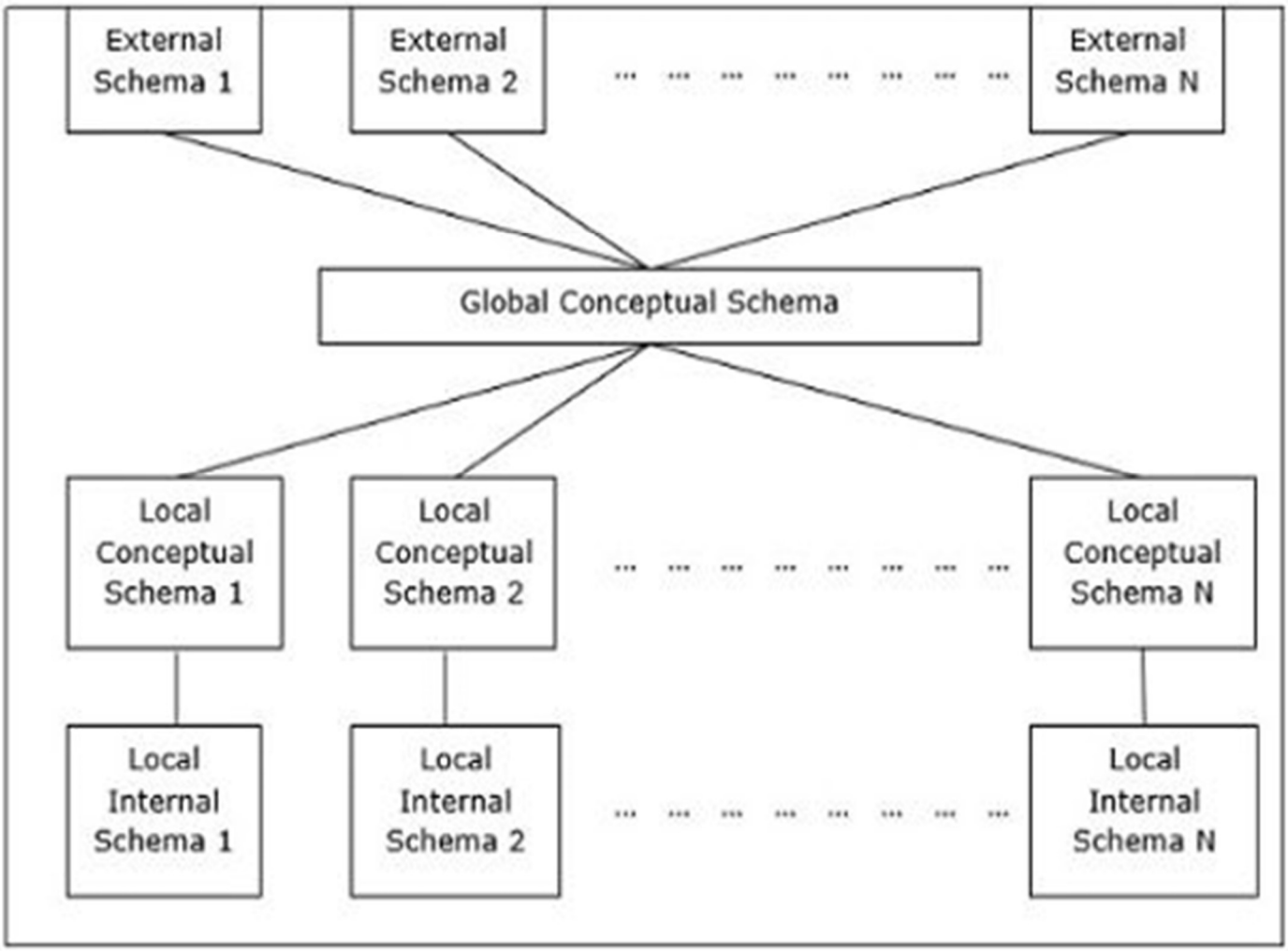
- Single Server Multiple Client
- Multiple Server Multiple Client



Peer-to-peer architecture: In this architecture, each site manages its own data and coordinates transactions with others. Every peer acts as both client and server, sharing resources and coordinating activities across the system.

This architecture generally has **four** levels of schemas –

- **Global Conceptual Schema** – Depicts the global logical view of data.
- **Local Conceptual Schema –** Depicts logical data organization at each site.
- **Local Internal Schema –** Depicts physical data organization at each site.
- **External Schema –** Depicts user view of data.



Multi-Database System Architecture:-

A Multi-Database System (MDBS) is a type of Distributed DBMS where each local database system is autonomous and independently managed, but they are integrated using a middleware or global schema for unified access.

2. Explain Data Fragmentation in Distributed Databases.

In fragmentation, the relations are broken (i.e., separated into smaller portions) in this manner, and each of the fragments is kept in multiple locations as needed. It must be ensured that the fragments can be utilized to recreate the original relationship (i.e., that no data is lost).

Fragmentation is useful since it avoids the creation of duplicate data, and consistency is not an issue.

Horizontal fragmentation — Splitting by rows — Each tuple is allocated to at least one fragment once the relation is broken into groups of tuples.

Vertical fragmentation — Splitting by columns — The relation's schema is broken into smaller schemas. To achieve a lossless join, each fragment must have a shared candidate key.

3. Explain Replication and Allocation Techniques for Distributed Database Design

Replication Techniques of distributed database:

1)Full Replication:

- Entire database copies are stored on multiple nodes.
- High availability and fault tolerance.
- Increased storage requirements, synchronization overhead.

2)Partial Replication:

- Only selected portions of the database are replicated.
- Reduces storage overhead, allows for tailored replication.
- Complexity in managing consistency across replicas.

3)Multilevel Replication:

- Different levels of replication based on importance or access frequency.
- Balances performance and resource utilization.
- Requires careful planning to determine replication levels.

Allocation Techniques of distributed database

1)Centralized Allocation:

- A central server manages data distribution to nodes.
- Simplifies administration, centralized control.
- Single point of failure, potential performance bottleneck.

2)Decentralized Allocation:

- Nodes autonomously manage their data allocation.
- Improved scalability, reduced dependency on a central server.
- Coordination challenges, potential for uneven data distribution.

3)Hash-Based Allocation:

- Data is distributed using a hash function on a key attribute.
- Uniform data distribution, efficient for range queries.
- Limited adaptability to changing access patterns.

4)Round Robin Allocation: Data is assigned to nodes in a cyclical fashion.

5)Random Allocation: Data is placed on nodes randomly.

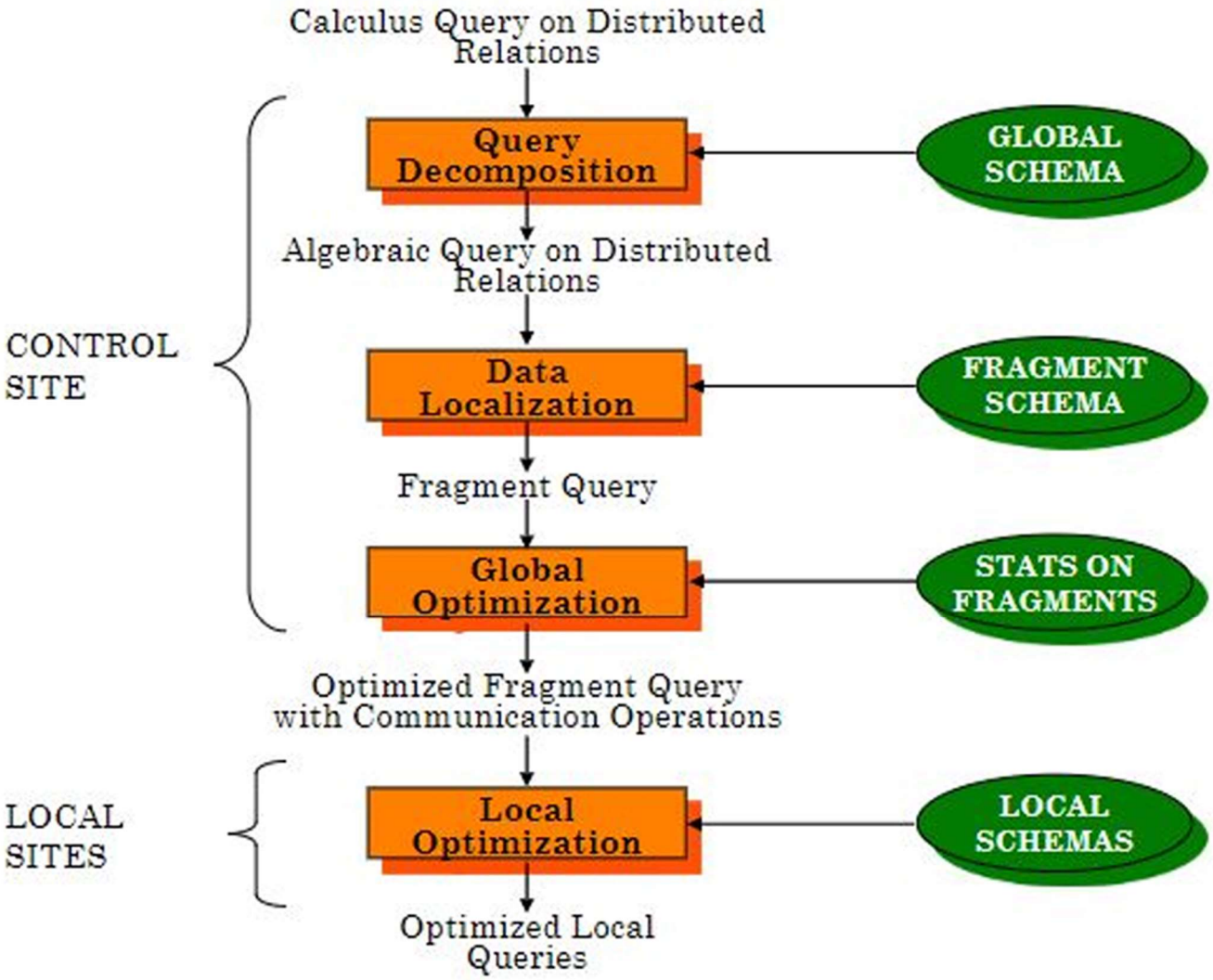
4. Write a short note on: Failures in Distributed Database

Types of Failures:

1. **Transaction Failures**
    - Occur when a transaction cannot complete successfully due to logical or system errors (e.g., deadlock, integrity constraint violation).
2. **Site Failures**
    - Happen when an entire site (node) goes down due to power failure, hardware crash, or OS failure.
3. **Communication Failures**
    - Arise when messages between sites are lost or delayed due to network issues, leading to timeouts or inconsistent states.
4. **Media Failures**
    - Involve loss or corruption of data stored on disk due to hardware malfunction or file corruption.

Module 2: Query Processing and Optimization

5. Explain phases in Distributed Query Processing with neat diagram.



Distributed Query Processing means running a query on data that is stored at different locations (sites) in a network. The goal is to get correct results quickly and efficiently by coordinating across all those sites.

Main Phases:

1. Query Decomposition

- The system checks if the user query is correct.
- It breaks the query into smaller steps using a simple format (like relational algebra).
- It also simplifies the query by removing unnecessary parts.

2. Data Localization

- Finds out where the required data is stored (which site or fragment).
- Changes the query so it can collect the right data from the right places.

3. Global Optimization

- Creates different ways to run the query.

- Chooses the best one based on speed, cost, and how much data needs to move.
- Uses special tricks like semi-joins to reduce data transfer.

4. Local Optimization

- Improves performance at each site where the query runs.
- Uses the local database's own tools to make things faster.

5. Query Execution

- Runs the final plan.
- Collects results from all sites and sends the final answer to the user.

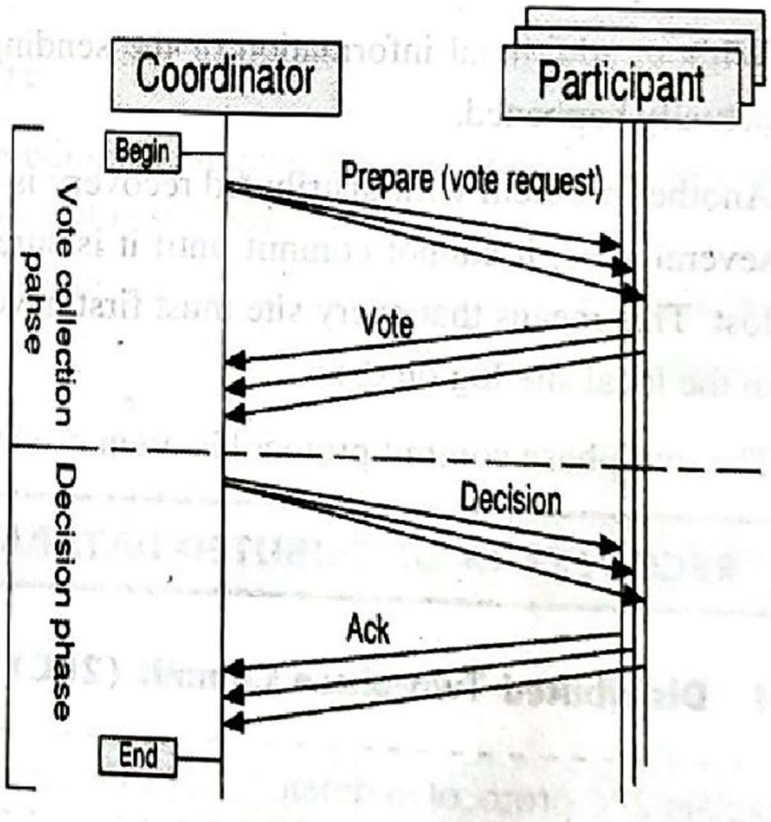
Draw the neat diagram of Query Processing

Just draw the diagram from previous page, asked for 5 marks

6. Explain 2PC in detail with neat diagram

2PC is a **distributed algorithm** used in **DBMS** to ensure **atomicity** in a distributed transaction across multiple databases or systems.

It ensures that **either all participants commit** the transaction or **none do**, even in case of failures.



Phases of 2PC

Phase 1: Prepare Phase (Voting Phase)

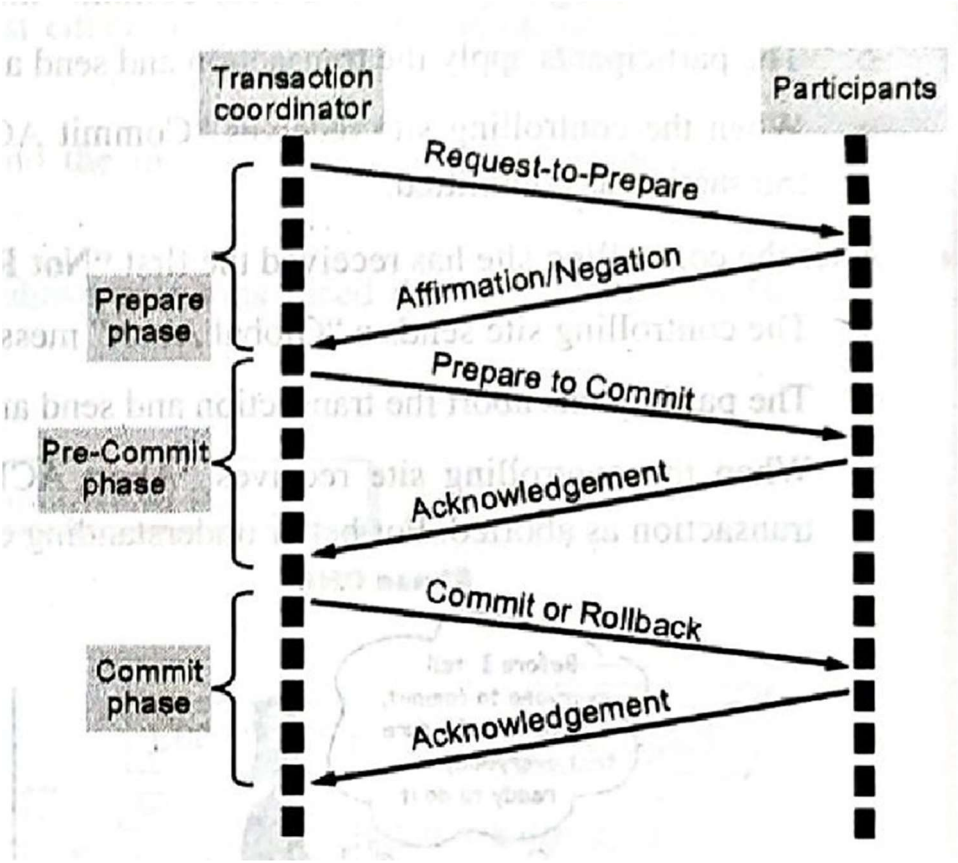
- The **coordinator** sends a **"Prepare to Commit?"** request to all participants.
- Each **participant** does:
  - Logs the transaction locally.
  - Replies with **“Yes (Vote-Commit)”** if ready.
  - Replies with **“No (Vote-Abort)”** if there's a problem (e.g., constraint violation).

Phase 2: Commit/Abort Phase

- If **all participants voted Yes**:
  - Coordinator sends **"Commit"** message.
  - All participants **commit the transaction** and log it.
- If **any participant voted No**:
  - Coordinator sends **"Abort"** message.
  - All participants **abort** and **roll back** the transaction.

7. Explain 3PC in detail with neat diagram

3PC is an improvement over the 2PC (Two-Phase Commit) protocol. It eliminates the blocking problem by introducing a third phase and ensuring non-blocking behaviour in most cases.



Phases of 3PC

3PC divides the commit process into **three phases** to reduce the chance of participants being stuck in uncertainty.

Phase 1: Can Commit? (Voting Phase)

- **Coordinator** sends **"CanCommit?"** request to all participants.
- Each **participant** replies with:
  - **"Yes"** (ready to commit), or
  - **"No"** (cannot commit).

Phase 2: Pre-Commit Phase

- If all participants say **"Yes"**:
  - Coordinator sends **"PreCommit"** message.
  - Participants **log** the message and **prepare to commit**, but **do not commit yet**.
  - Participants acknowledge receipt.

Phase 3: Do Commit (Commit Phase)

- After receiving **all acknowledgments**:
  - Coordinator sends **"DoCommit"**.
  - Participants **finally commit** the transaction.

8. Explain ACID properties

1. Atomicity – All or Nothing

- A transaction should either finish completely or not happen at all.
- If there's any error, all changes are cancelled.
- **Example**: When ₹1000 is transferred from Account A to B, both debit and credit must happen. If one fails, both are cancelled.

2. Consistency – Stay Valid

- The database must remain correct before and after a transaction.
- All rules and conditions (like no negative balance) must be followed.
- **Example**: If a rule says balance can't go below ₹0, the transaction must not break that.

3. Isolation – No Interference

- Transactions running at the same time should not affect each other.
- Each transaction acts like it's the only one running.
- **Example**: Two people booking the same train seat won't get the same seat.

4. Durability – Changes Stay Safe

- Once a transaction is done, its changes are saved permanently.
- Even if there's a crash or power cut, the changes remain.
- **Example**: After paying online, the payment stays recorded even if the system shuts down.

9. Write a short note on: Query Evaluation Plan

A **Query Evaluation Plan** is a method the database uses to understand and run your query in the best way.

When you ask the database a question (like using SQL), the system figures out different ways to get the answer, then chooses the fastest and cheapest way.

- The query is broken into small steps like **filtering, joining, or sorting**.
- There are **many ways** to do these steps.
- The system **checks which way is faster or uses less memory**.
- It picks the **best plan** and runs it.

10. Write a short note on: Query Processing Issues in Heterogeneous Database

1. **Different Query Languages**:
   Each database may use a different language (e.g., SQL, XQuery), so it's hard to write one query that works for all.
2. **Different Data Models**:
   One database may use tables; another may use documents or key-value pairs. Combining them in one query is tricky.
3. **Different Schemas**:
   The way data is organized may be different in each database, so joining data is complex.
4. **Data Location and Access Speed**:
   Data is stored in different places and some databases may be slower to respond, affecting overall speed.
5. **Security and Access Control**:
   Each database might have its own rules for who can access what, making it hard to run queries across all of them.

11. Differentiate between 2PC and 3PC Protocol

Point	2PC (Two-Phase Commit)	3PC (Three-Phase Commit)
1. Phases	Has 2 phases: Prepare and Commit/Abort	Has 3 phases: Prepare, Pre-Commit, and Commit/Abort
2. Blocking Nature	Can block if coordinator crashes	Non-blocking due to extra pre-commit phase
3. Coordinator Failure	Participants may wait forever	Participants can decide if coordinator fails
4. Message Overhead	Fewer messages (less overhead)	More messages (extra phase adds overhead)
5. Simplicity	Simpler and easier to implement	More complex to implement
6. Crash Handling	Poor recovery after crash	Better crash recovery and decision-making
7. Example	Bank transfer waits if server crashes	In 3PC, transaction can still continue using timeout rules

Module 3: JSON, XML and Semi-Structured Data

12. Explain basic JSON syntax with data types

JSON (JavaScript Object Notation) is a lightweight data format used for storing and exchanging data between systems, especially in web applications. It is easy for humans to read and write and for machines to parse and generate.

Basic JSON Syntax Rules:

1. **Data is in key-value pairs**
   Example: "name": "John"
2. **Data is separated by commas**
   Example: "name": "John", "age": 25
3. **Curly braces { } hold objects**
   Example:

```
{
  "name": "John",
  "age": 25
}
```

4. **Square brackets [ ] hold arrays**
   Example:

```
{
  "fruits": ["apple", "banana", "mango"]
}
```

JSON Data Types:

Type	Example	Description
String	"name": "Alice"	Text value inside double quotes
Number	"age": 30	Integer or float (no quotes)
Boolean	"married": true	true or false (lowercase, no quotes)
Object	"address": {"city": "Delhi"}	Nested data in key-value format
Array	"colors": ["red", "blue"]	Ordered list of values
Null	"middleName": null	Represents no value

13. What is XML? Explain XML Schema Document with example.

XML (eXtensible Markup Language) is a markup language used to store and transport data. It is designed to be both human-readable and machine-readable. XML defines a set of rules for encoding documents in a format that is both flexible and structured.

Features of XML:

- Self-descriptive (tags define the data)
- Supports nested data (hierarchical structure)
- Platform-independent and language-independent
- Used in data exchange between systems (e.g., APIs, configurations)

XML Schema: An XML Schema (often written in XSD - XML Schema Definition) defines the structure and data types of an XML document. It acts like a blueprint or validation rulebook that checks:

- What elements are allowed
- Order of elements
- Data types (string, integer, date, etc.)

XML Document:

```
<student>
 <name>John Doe</name>
 <age>20</age>
 <email>john@example.com</email>
</student>
```

XML Schema (XSD):

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="student">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="age" type="xs:integer"/>
    <xs:element name="email" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

14. Create a XML document. PYQ:

XML document of ‘Restaurant Menu Card’ has food items categorized into Starters, Drinks, Chinese, South and Punjabi. Each food item element contains name, cost, calories, and veg/non-veg flag. Write XML Schema for the above XML document.

XML Document (menu.xml)

```
<menu>
 <starters>
  <item>
   <name>Paneer Tikka</name>
   <cost>180</cost>
   <calories>250</calories>
   <veg>true</veg>
  </item>
  <item>
   <name>Chicken Kebab</name>
   <cost>220</cost>
   <calories>300</calories>
   <veg>false</veg>
  </item>
 </starters>

 <drinks>
  <item>
   <name>Sweet Lime Soda</name>
   <cost>50</cost>
   <calories>120</calories>
   <veg>true</veg>
  </item>
 </drinks>

 <chinese>
  <item>
```

```
    <name>Veg Manchurian</name>
    <cost>150</cost>
    <calories>270</calories>
    <veg>true</veg>
   </item>
  </chinese>

  <south>
   <item>
    <name>Masala Dosa</name>
    <cost>90</cost>
    <calories>320</calories>
    <veg>true</veg>
   </item>
  </south>

  <punjabi>
   <item>
    <name>Butter Chicken</name>
    <cost>240</cost>
    <calories>500</calories>
    <veg>false</veg>
   </item>
  </punjabi>
</menu>
```

XML Schema (menu.xsd)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="menu">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="starters" type="CategoryType"/>
    <xs:element name="drinks" type="CategoryType"/>
    <xs:element name="chinese" type="CategoryType"/>
    <xs:element name="south" type="CategoryType"/>
    <xs:element name="punjabi" type="CategoryType"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:complexType name="CategoryType">
  <xs:sequence>
   <xs:element name="item" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="cost" type="xs:decimal"/>
      <xs:element name="calories" type="xs:integer"/>
      <xs:element name="veg" type="xs:boolean"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>

</xs:schema>
```

15. Differentiate between XML and JSON

Point	XML	JSON
1. Format	Markup language	Data format
2. Syntax	Uses tags like <name>John</name>	Uses key-value pairs like "name": "John"
3. Readability	More verbose and harder to read	Short, simple, and easy to read
4. Data Size	Larger due to opening and closing tags	Smaller in size
5. Parsing Speed	Slower parsing	Faster parsing
6. Support for Data Types	All values are strings by default	Supports strings, numbers, Boolean, null etc.
7. Use in Web	Mostly used in legacy systems and SOAP APIs	Commonly used in modern REST APIs
8. Example	<name>John</name>	"name": "John"

Module 4: NoSQL

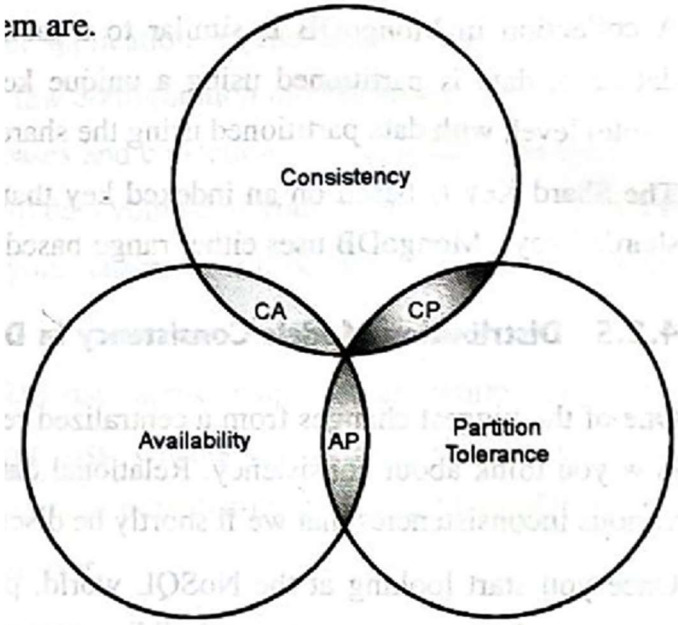
16. Explain CAP theorem in NoSQL Database

The CAP Theorem, proposed by Eric Brewer, describes the three essential guarantees that a distributed database system can provide:

1. **Consistency (C)**
   All nodes in the system return the same data at the same time after an update.
2. **Availability (A)**
   Every request receives a response, even if some nodes are down — no guarantee that it contains the latest data.
3. **Partition Tolerance (P)**
   The system continues to operate despite network failures that prevent communication between nodes.

CAP Theorem Rule:

In any distributed system, it is impossible to simultaneously guarantee all three: Consistency, Availability, and Partition Tolerance. A system can satisfy at most two of these three properties.



g. 4.2.2: Three main features Distributed system

Implications in NoSQL:

NoSQL databases sacrifice one property to achieve the other two, depending on the use case:

Database Type	Properties Focused	Example
CP (Consistency + Partition Tolerance)	Strong consistency even during partition	MongoDB (default), HBase
AP (Availability + Partition Tolerance)	High availability with eventual consistency	CouchDB, Cassandra
CA (Consistency + Availability)	Not practical in a distributed system as partition tolerance is essential	

17. Differentiate between SQL and NoSQL

Feature	SQL (Relational DB)	NoSQL (Non-relational DB)
Data Model	Table-based with rows and columns	Document, key-value, graph, or column-based
Schema	Fixed schema; predefined structure	Dynamic schema; flexible structure
Scalability	Vertically scalable	Horizontally scalable
Query Language	Uses SQL (Structured Query Language)	Uses various query methods (e.g., JSON-like)
ACID Compliance	Fully ACID compliant	BASE model (eventual consistency)
Best Suited For	Complex queries, transactional systems	Big data, real-time web apps, unstructured data
Example	MySQL, PostgreSQL	MongoDB, Cassandra

18. Write a short note on: NoSQL Data Modelling

NoSQL data modelling is the process of designing the structure of data for non-relational databases, based on the application's access patterns. Unlike relational databases which follow a normalized structure, NoSQL focuses on denormalization to improve performance and scalability.

In NoSQL, the data model depends on the type of database (document, key-value, column-family, or graph). It is typically designed around how the data will be read or queried, rather than how it is stored. For instance, in a document store like MongoDB, related data is often embedded within the same document to reduce the need for joins.

Key aspects of NoSQL data modelling include:

- **Design for queries**, not for storage.
- **Embed or reference** data depending on consistency and performance needs.
- **Balance between duplication and normalization**.
- Consider **partitioning** and **sharding** from the start.

19. Explain Replication and Sharding in NoSQL

Replication:

Replication involves **copying data across multiple servers** (nodes). One node acts as the **primary** (or master), and others are **secondary** (or replicas). The primary handles write operations, while secondaries replicate data and can serve read requests.

Benefits:

- Ensures **high availability** in case the primary node fails.
- Improves **read scalability** by distributing read operations across replicas.
- Provides **data redundancy** for disaster recovery.

Example: In MongoDB, a **replica set** maintains multiple copies of data across different nodes.

Sharding:

Sharding is the process of **partitioning data horizontally** across multiple servers, called **shards**, where each shard holds a subset of the total dataset.

Benefits:

- Enables handling of **large-scale datasets** by splitting them across servers.
- Improves **write and read performance** by parallelizing data access.
- Supports **scalability** by allowing more shards to be added as needed.

Example: In MongoDB, sharding uses a **shard key** to determine which shard stores a particular document.

20. Explain Benefits of NoSQL

High Scalability: NoSQL databases offer horizontal scaling through sharding, making them capable of handling large amounts of data and growing with demand.

Flexibility: NoSQL databases accommodate unstructured or semi-structured data, making them suitable for applications with evolving data models.

High Availability: Auto-replication features enhance data availability by replicating data in case of failures.

Performance: NoSQL databases are optimized for handling large data volumes and traffic, resulting in improved performance.

Cost-effectiveness: NoSQL databases are often cost-effective due to their simplicity and reduced hardware/software requirements.

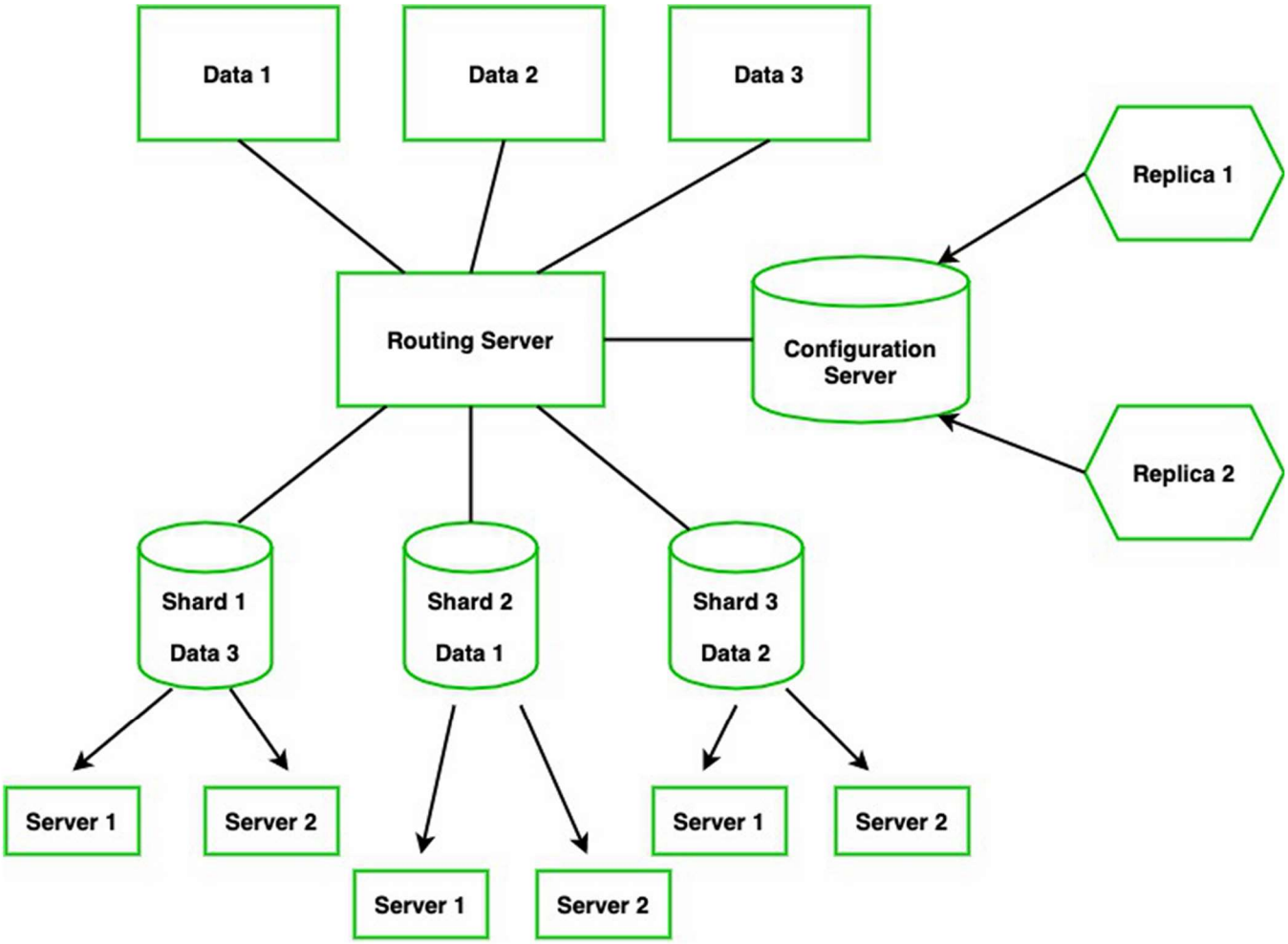
Agility: Well-suited for agile development methodologies.

Module 5: NoSQL using MongoDB

21. Explain MongoDB Sharding

Sharding in MongoDB enables horizontal scaling by distributing data across multiple servers (shards). It is used to handle large datasets that exceed the capacity of a single server. Sharding is a method for allocating data across multiple machines.

MongoDB used sharding to help deployment with very big data sets and large throughput the operation. By sharding, you combine more devices to carry data extension and the needs of read and write operations.



Sharding determines the problem with horizontal scaling, breaking the system dataset and store over multiple servers, adding new servers to increase the volume as needed. Now, instead of one signal as primary, we have multiple servers called Shards. We have different routing servers that will route data to the shard servers.

Advantages of Sharding:

1. **Scalability:** Easily adds more servers and distributes data automatically.
2. **Load Balancing:** Reduces the number of operations each shard handles.
3. **Increased Capacity:** Total storage and performance scale with more shards.

22. Explain MongoDB CRUD Operations.

In MongoDB, CRUD operations allow users to interact with the database by Creating, Reading, Updating, and Deleting documents in collections. Each operation plays a vital role in managing NoSQL data effectively.

1. Create (Insert)

This operation is used to add new documents to a collection.

- **insertOne()**: Adds a single document.
- **insertMany()**: Adds multiple documents.

Example:

```
db.students.insertOne({ name: "Aryan", age: 21, course: "DBMS" });
```

2. Read (Find)

This retrieves documents based on specified criteria.

- **find()**: Returns a cursor to all matching documents.
- **findOne()**: Returns the first matching document.

Example:

```
db.students.find({ course: "DBMS" });
```

3. Update

Used to **modify existing documents**.

- **updateOne()**: Updates the first matched document.
- **updateMany()**: Updates all matched documents.

Operators used: $set, $inc, $unset, etc.

Example:

```
db.students.updateOne(
 { name: "Aryan" },
 { $set: { age: 22 } }
);
```

4. Delete

Used to **remove documents** from a collection.

- **deleteOne()**: Deletes the first matched document.
- **deleteMany()**: Deletes all matched documents.

Example:

```
db.students.deleteOne({ name: "Aryan" });
```

23. Explain basic data types in MongoDB

String: Most commonly used data type to store text (e.g., "name": "Ayaan").

Integer: Stores numerical values (e.g., "age": 21).

Boolean: Stores true or false values (e.g., "isStudent": true).

Double: Stores floating-point numbers (e.g., "score": 88.5).

Array: Stores multiple values in a single key (e.g., "skills": ["Python", "C++"]).

Object: Stores embedded documents (e.g., "address": {"city": "Mumbai", "pin": 400001}).

Module 6: Trends in advance databases

24. What is a Graph Database? What are the features of Graph Database?

A graph database is a type of NoSQL database that is designed to store and process data in the form of graphs. Graph databases are particularly well-suited for managing highly interconnected data and are used to represent and query relationships between entities.

The fundamental components of a graph database are nodes, edges, and properties.

Nodes: Represent entities in the graph.

Edges: Represent relationships between nodes.

Properties: Key-value pairs associated with nodes and edges to store additional information.

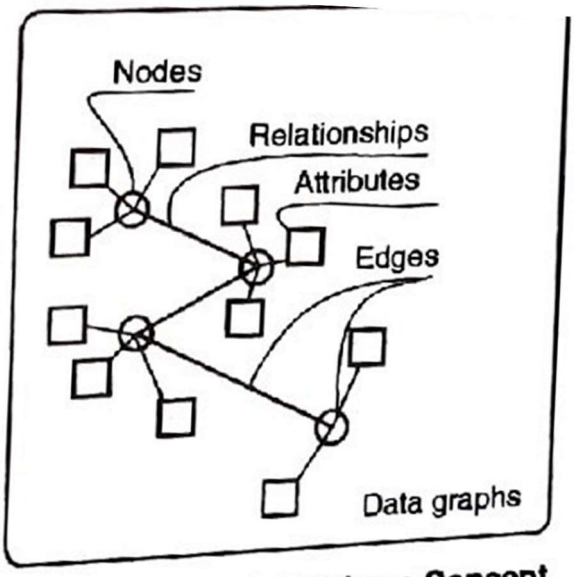


Fig. 9.5.1 : Graph Database Concept

Features of Graph Databases:

1. **Graph-based Design:**
   They store data as nodes and connections (edges), so you don't need to convert it into tables. It's a natural way to show how things are related.
2. **Fast at Finding Links:**
   Graph databases are really good at quickly checking how things are connected. This is helpful for things like finding friends of friends.
3. **Flexible Structure:**
   You don't need to stick to one fixed layout. You can add new types of data or connections whenever you want, without changing everything.
4. **Great for Connected Data:**
   Perfect for data that has lots of links—like social media, recommendation systems, or spotting fraud.
5. **Special Query Language:**
   They use their own easy-to-use language made just for searching through graphs and relationships.

25. Explain Spatial Database in detail

A Spatial Database is a type of database that is optimized to store and query data related to objects in space, including points, lines, and polygons. These objects represent real-world geographical entities like cities, rivers, roads, and boundaries.

Spatial Data Types:

Point: Represents a single location in space, defined by its coordinates (latitude, longitude).

Line String: Represents a sequence of connected line segments, forming a path.

Polygon: Represents a closed, planar area defined by a sequence of connected points.

Geometry Collection: A collection of spatial objects of different types, such as points, lines, or polygons.

Spatial Data Models:

Vector Model: Represents spatial data as discrete objects with precise geometry (points, lines, polygons). Commonly used in GIS.

Raster Model: Represents spatial data as a grid of cells, where each cell has a value.

Spatial Indexing:

R-Tree: A tree structure that is used to organize spatial data in a way that facilitates efficient spatial queries.

Quadtree: A tree structure that recursively subdivides a space into quadrants.

Examples of Spatial Databases:

- **PostGIS** (extension of PostgreSQL)
- **Oracle Spatial**
- **MySQL Spatial Extensions**

26. Explain Temporal Database in detail

A **Temporal Database** is designed to store and manage data that changes over time. Unlike traditional databases that only reflect the current state of data, temporal databases maintain historical records, allowing users to track changes and query data as it existed at specific points in time.

These databases typically use two-time dimensions: **valid time**, which indicates when a fact is true in the real world, and **transaction time**, which records when the data was stored in the database. Some systems support **bi-temporal data**, combining both dimensions to provide a comprehensive timeline of data changes.

Time Representation:

Point-in-Time Representation: Represents data as it exists at a specific point in time. Querying retrieves data valid at a particular timestamp.

Interval Representation: Represents data as valid within a specific time interval.

Incorporating Time in Relational Databases:

Temporal Tables:

Tables are extended to include temporal columns for valid time and transaction time. Each row in the table has associated time intervals.

Temporal Queries:

Temporal databases support temporal queries that involve querying data based on valid time, transaction time, or both. Common operations include "time travel" queries to retrieve data at a specific point in time.

Temporal Constraints:

Temporal databases can enforce temporal constraints, ensuring that data in the database adheres to valid time and transaction time semantics.

Challenges in Temporal Databases:

Data Volume: Maintaining historical versions of data can lead to increased data volume, impacting storage requirements.

Query Complexity: Temporal queries can be more complex than traditional queries.

2019 May

Q1. Answer the following (any four):

a) What are the operations performed on files?
b) Explain join operations in DBMS.
c) What are the various allocation techniques in distributed databases?
d) Why is object-oriented database needed?
e) Explain security issues in database design.

Q2.

a) What is heuristic query optimization? Explain steps with example.
b) Explain concurrency control in distributed databases.

Q3.

a) Explain the following types of fragmentation with suitable examples:
   i. Horizontal fragmentation
   ii. Vertical fragmentation
   iii. Derived fragmentation
b) Compare document-oriented databases and traditional databases.

Q4.

a) What are the steps in query processing and optimization in distributed databases?
b) Explain mobile computing architecture for database systems.

Q5.

a) Explain hashing techniques used in databases.
b) Explain discretionary access control with example.

Q6. Write short notes on (any two):

a) Measures of query cost
b) Two-phase commit protocol
c) Document-oriented database
d) Multimedia databases

2019 Dec

Q1. Answer the following (any four):

a) Explain query processing in distributed databases.
b) Why do we need document-oriented databases?
c) What is cost-based query optimization?
d) What is SQL injection? Explain with example.

Q2.

a) Explain external sorting with example.
b) Consider a relation R(A, B, C, D, E) horizontally fragmented into R1, R2, and R3. Write the process to reconstruct the original relation from these fragments.

Q3.

a) What is the three-phase commit protocol? Explain with example.
b) Explain XML and XML Schema with example.

Q4.

a) Explain architecture of distributed database system.
b) What are the types of spatial data models?

Q5.

a) What is temporal data model? Explain with example.
b) Explain discretionary access control with example.

Q6. Write short notes on (any two):

a) Single-level ordered index
b) Replication and allocation techniques
c) Mandatory access control

2022 Dec

Q1. Solve any four:

a) Draw the neat diagram of query processing.
b) Describe query evaluation plan.
c) Compare between SQL and NoSQL.
d) Write the different parameters for measuring the cost of query.
e) What is Graph database?

Q2

a) Discuss the phases of distributed query processing with neat diagram.
b) Explain Temporal database.

Q3

a) Explain CAP Theorem in NoSQL Databases.
b) XML document of 'Restaurant Menu Card' has food items categorized into Starters, Drinks, Chinese, South and Punjabi. Each food item element contains name, cost, calories, and veg/non-veg flag. Write XML Schema for the above XML document.

Q4

a) Explain MongoDB Sharding.
b) Explain ACID properties.

Q5

a) Explain 2PC in detail with neat diagram.
b) Explain JSON data types. Give example of JSON document.

Q6

a) Explain Spatial Database.
b) Explain MongoDB CRUD Operations.

2023 May

Q1. Attempt any four

a) What are the features of Graph Database?
b) Explain Basic data types with MongoDB.
c) List responsibilities of Transaction Manager in Distributed Transaction Model.
d) Explain Query Processing issues in Heterogeneous Database.

Q2

a) Explain how horizontal scaling is done through sharding in MongoDB.
b) Explain Three Phase Commit (3PC) Protocol.

Q3

a) Explain Allocation techniques for Distributed Database Design.
b) Explain Basic JSON syntax with data types.

Q4

a) Create an XML Document of 'Restaurant Menu Card' which has food items categorized into Drinks, Starters, and Main Course. Each food item element contains name, cost and veg/non-veg flag.

- Write DTD rules for above XML Document. (8 marks)
- Write the following query in XQuery (2 marks):
  "List the entire list of food name with non-veg flag."

b) Explain Replication and sharding in NoSQL.

Q5

a) Explain Phases in Distributed Query Processing in Distributed Database.
b) Explain Basic JSON syntax with data types.

Q6. Write a detailed note on the following (Any Two):

a) NoSQL data modelling
b) Basic Data types in MongoDB
c) Failures in Distributed Database
d) CAP theorem in NoSQL

2023 Dec

Q1. Solve any four

a) Compare between SQL and NoSQL.
b) Explain BASE Theorem.
c) Explain sharding in MongoDB.
d) Draw the neat diagram of query processing.
e) What is Graph database?

Q2

a) Explain Spatial Database. (10 marks)
b) Write an XML to accept student details (Name, ID, Branch and CGPA).
Write an XSL to display the list of students in descending order of their CGPA. (10 marks)

Q3

a) Explain with a suitable example XML DTD? (10 marks)
b) Explain Data Fragmentation in Distributed Databases. (10 marks)

Q4

a) Compare 2PC and 3PC protocol. (10 marks)
b) Explain ACID properties. (10 marks)

Q5

a) Explain Distributed DBMS Architecture. (10 marks)
b) Explain MongoDB CRUD Operations. (10 marks)

Q6

a) Discuss the phases of distributed query processing with neat diagram. (10 marks)
b) Explain Benefits of NoSQL. (10 marks)