CBCS SCHEME

21CS745

USN 1 C R 2 1 C S 1 1 6

Seventh Semester B.E./B.Tech. Degree Examination, Dec.2024/Jan.2025
NOSQL Database

Time: 3 hrs. Max. Marks: 100

Note: *Answer any FIVE full questions, choosing ONE full question from each module.*

Module-1

1 a. Discuss the key difference between NOSQL and Relational databases. (10 Marks)

b. Provide strategies for optimizing data models to improve application performance. (10 Marks)

OR

2 a. Explain impendence mismatch problem in the context of data storage, and how does it affect application development. (10 Marks)

b. Describe the concept of schemaless databases and their benefits. (10 Marks)

Module-2

3 a. Explain the concepts of Master – Slave and Peer- to – peer replication in distributed databases. (10 Marks)

b. Discuss the importance of version stamp in esuring consistency across multiple nodes in a distributed database. (10 Marks)

OR

4 a. Discuss the challenges associated with achieving update consistency in a distributed database system. (10 Marks)

b. Discuss read consistency in distributed databases, considering factors such as staleness and isolation levels. (10 Marks)

Module-3

5 a. Provide an example of composing Map-Reduce calculations to process and Analyze data. (10 Marks)

b. Discuss the scalability characteristics of key-value databases. (10 Marks)

OR

6 a. Discuss key features of key value stores and their advantages. (10 Marks)

b. Describe the basic structure of data in a key value databases. (10 Marks)

Module-4

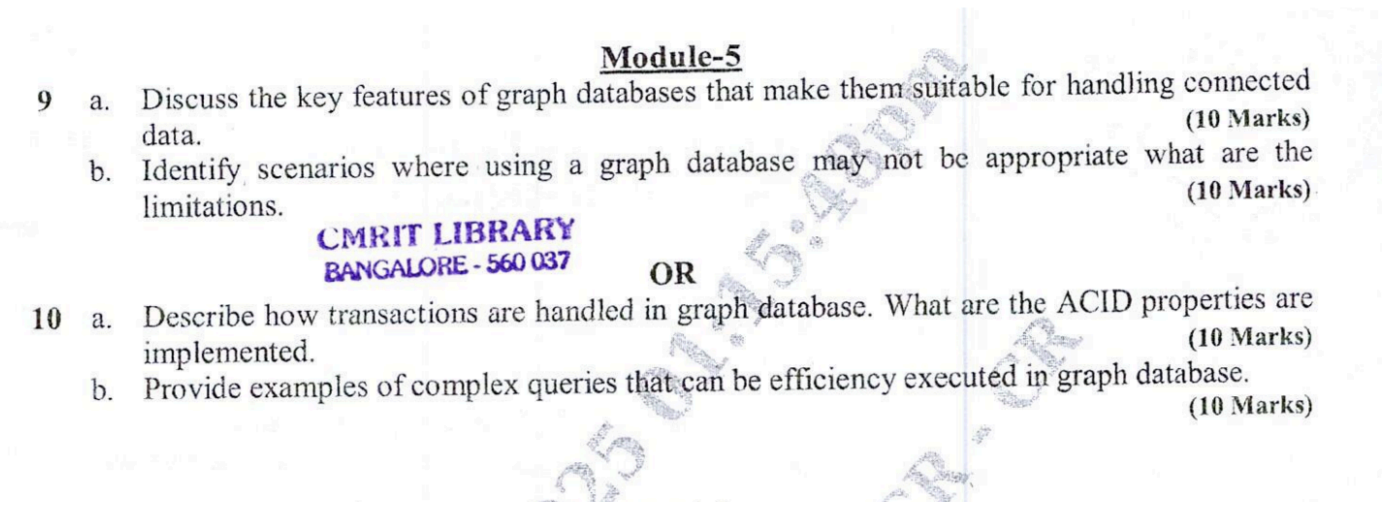
7 a. Explain fundamental principles of document database. (10 Marks)

b. Discuss the importance of SEO (Search Engine Optimization) in the context of blogging platforms. (10 Marks)

OR

8 a. Provide examples of common query operations performed on document databases and explain their significance. (10 Marks)

b. Explain the importance of event logging in web applications with examples. (10 Marks)


Module-5

9 a. Discuss the key features of graph databases that make them suitable for handling connected data. (10 Marks)

b. Identify scenarios where using a graph database may not be appropriate what are the limitations. (10 Marks)

OR

10 a. Describe how transactions are handled in graph database. What are the ACID properties are implemented. (10 Marks)

b. Provide examples of complex queries that can be efficiency executed in graph database. (10 Marks)

Module 1

1. a. Discuss the key differences between NoSQL and Relational Databases. (10 Marks)

Feature	NoSQL Databases	Relational Databases (RDBMS)
Data Model	Flexible schema (document, key-value, graph, column-family)	Fixed, pre-defined schema (tables with rows and columns)
Scalability	Horizontally scalable (scale-out by adding more servers)	Vertically scalable (scale-up by increasing server resources)
Consistency	Focus on eventual consistency (CAP theorem: sacrificing consistency for availability and partition tolerance)	ACID properties (Atomicity, Consistency, Isolation, Durability) ensure strong consistency
Data Relationships	Less emphasis on complex relationships, often uses denormalization	Designed for managing complex relationships using joins and foreign keys
Use Cases	Handling large volumes of unstructured or semi-structured data, real-time data processing, high traffic web applications	Structured data, applications requiring complex transactions and reporting, data warehousing
Examples	MongoDB, Cassandra, Redis, Neo4j	MySQL, PostgreSQL, Oracle, SQL Server

Export to Sheets

1. b. Provide strategies for optimizing data models to improve application performance. (10 Marks)

- **Choose the Right NoSQL Database:** Select the database type that best matches your data model and access patterns (e.g., document database for flexible schemas, key-value store for simple lookups).
- **Data Modeling for Access Patterns:** Design your data model based on how your application will query and access the data. Optimize for frequent queries.
- **Denormalization:** Reduce the need for joins by duplicating data across documents or tables. This improves read performance but increases storage and update complexity.

- **Caching:** Implement caching mechanisms to store frequently accessed data in memory, reducing database load and latency.
- **Indexing:** Create indexes on frequently queried fields to speed up data retrieval. Be mindful of index size as it can impact write performance.
- **Sharding:** Distribute data across multiple servers (shards) to improve read and write throughput for large datasets.
- **Connection Pooling:** Reuse database connections to reduce the overhead of establishing new connections for each request.
- **Monitoring and Profiling:** Continuously monitor database performance and use profiling tools to identify bottlenecks and optimize queries.

2. a. Explain the impedance mismatch problem in the context of data storage, and how does it affect application development. (10 Marks)

- **Impedance Mismatch:** The mismatch between the object-oriented paradigm used in application development (objects, classes) and the relational model used in traditional databases (tables, rows).
- **Effects on Application Development:**
    - **Object-Relational Mapping (ORM):** Requires ORM frameworks to translate between objects and relational data, adding complexity and potential performance overhead.
    - **Data Access Code:** Developers write code to map data between objects and database tables, leading to boilerplate code and potential for errors.
    - **Performance Bottlenecks:** ORM can introduce performance issues due to complex mappings and inefficient queries.
    - **Schema Rigidity:** Relational schemas are rigid, making it difficult to adapt to evolving application requirements.

2. b. Describe the concept of schemaless databases and their benefits. (10 Marks)

- **Schemaless Databases:** NoSQL databases that do not require a pre-defined schema. Documents or data entries can have varying structures and fields.
- **Benefits:**
    - **Flexibility:** Easily adapt to changing data requirements without schema migrations.
    - **Agility:** Faster development cycles as developers don't need to spend time designing and managing schemas.
    - **Handling Unstructured Data:** Well-suited for storing and processing unstructured or semi-structured data.
    - **Developer Productivity:** Easier to work with as developers don't need to worry about schema constraints.

Module 2

3. a. Explain the concepts of Master-Slave and Peer-to-Peer replication in distributed databases. (10 Marks)

- **Master-Slave Replication:**
    - A single master node handles all write operations.
    - Slave nodes replicate data from the master and handle read operations.
    - **Pros:** Simple to implement, good for read scaling.
    - **Cons:** Single point of failure (master), write bottleneck on master.
- **Peer-to-Peer Replication:**
    - All nodes are equal and can handle both read and write operations.
    - Data is replicated across all nodes.
    - **Pros:** High availability, no single point of failure, improved write scalability.

    - **Cons:** More complex to implement, requires conflict resolution mechanisms.

3. b. Discuss the importance of version stamps in ensuring consistency across multiple nodes in a distributed database. (10 Marks)

- **Version Stamps (or Vector Clocks):** Used to track the order of updates in a distributed system. Each update is associated with a version stamp that reflects its causal history.
- **Importance:**
    - **Conflict Detection:** Version stamps help detect conflicting updates made concurrently on different nodes.
    - **Conflict Resolution:** They provide information needed to resolve conflicts by determining the order of updates.
    - **Causal Consistency:** Version stamps help ensure causal consistency, where updates are seen in the same order by all nodes.

4. a. Discuss the challenges associated with achieving update consistency in a distributed database system. (10 Marks)

- **Challenges:**
    - **Network Latency:** Delays in communication between nodes can lead to inconsistencies.
    - **Concurrency Control:** Managing concurrent updates from multiple users while maintaining data integrity.
    - **Distributed Transactions:** Ensuring atomicity, consistency, isolation, and durability (ACID properties) across multiple nodes.
    - **Conflict Resolution:** Handling conflicting updates made concurrently on different nodes.
    - **Partial Failures:** Dealing with node failures or network partitions while maintaining data consistency.

4. b. Discuss read consistency in distributed databases, considering factors such as staleness and isolation levels. (10 Marks)

- **Read Consistency:** Ensures that read operations return the most up-to-date data.
- **Factors:**
    - **Staleness:** The degree to which read data is out-of-date. Eventual consistency allows for some staleness.
    - **Isolation Levels:** Control the degree to which transactions are isolated from each other, affecting read consistency.
        - **Read Uncommitted:** Allows reading uncommitted data (can lead to dirty reads).
        - **Read Committed:** Reads only committed data (prevents dirty reads).
        - **Repeatable Read:** Ensures that the same data is read throughout a transaction.
        - **Serializable:** Provides the highest level of isolation, ensuring that transactions appear to execute serially.

Module 3

5. a. Provide an example of composing Map-Reduce calculations to process and analyze data. (10 Marks)

- **Example: Word Count**
    - **Input:** A large text document.
    - **Map Phase:**
        - Split the document into chunks.
        - For each chunk, emit key-value pairs: (word, 1) for each word in the chunk.
    - **Reduce Phase:**
        - Group the key-value pairs by word.

        - For each word, sum the values to get the total count.
    - **Output:** A list of words and their counts.

5. b. Discuss the scalability characteristics of key-value databases. (10 Marks)

- **Scalability Characteristics:**
    - **Horizontal Scalability:** Key-value databases are designed for horizontal scalability. You can easily add more nodes to the cluster to handle increasing data volumes and traffic.
    - **Shared-Nothing Architecture:** Data is partitioned across nodes, and each node is independent. This avoids bottlenecks and allows for linear scalability.
    - **Distributed Data Management:** Key-value databases provide mechanisms for distributing data and managing replication across nodes.

6. a. Discuss the key features of key-value stores and their advantages. (10 Marks)

- **Key Features:**
    - **Simple Data Model:** Data is stored as key-value pairs.
    - **Fast Lookups:** Efficient retrieval of data using keys.
    - **High Scalability:** Designed for horizontal scalability.
    - **Minimal Schema:** No fixed schema, allowing for flexible data structures.
- **Advantages:**
    - **Performance:** Optimized for fast read and write operations.
    - **Scalability:** Handle massive amounts of data and high traffic loads.
    - **Simplicity:** Easy to use and understand.

6. b. Describe the basic structure of data in key-value databases. (10 Marks)

- **Basic Structure:**
    - **Key:** A unique identifier that is used to retrieve the associated value.
    - **Value:** The data associated with the key. It can be any type of data (string, number, JSON, etc.).
    - **Key-Value Pair:** The combination of a key and its corresponding value.

Module 4

7. a. Explain the fundamental principles of document databases. (10 Marks)

Document databases are a type of NoSQL database designed for storing, retrieving, and managing data as collections of documents. These documents are typically semi-structured and can vary in structure, offering flexibility and scalability. Here are the fundamental principles:

- **Document-Oriented Structure:** The core principle is storing data as "documents." A document typically represents a record or entity and is the fundamental unit of storage and retrieval. Documents are often represented in JSON or BSON format, making them human-readable and easily parsed by applications.

- **Schema-less or Flexible Schema:** Unlike relational databases with rigid, pre-defined schemas, document databases are often schema-less or have flexible schemas. This means that documents within the same collection can have different structures and fields. This allows for easy adaptation to evolving data requirements without costly schema migrations. While some structure might be beneficial for querying and indexing, it's not enforced at the database level.

- **Self-Describing Data:** Documents are self-describing, meaning they contain both the data and the metadata (field names) within the same unit. This makes the data more understandable and portable. Applications don't need to consult a separate schema to understand the structure of the data.

- **Collection-Based Organization:** Documents are grouped into "collections," which are analogous to tables in relational databases. Collections help organize and manage large sets of documents.

- **Focus on Semi-structured Data:** Document databases excel at managing semi-structured data, which doesn't conform to the rigid structure of relational databases. This makes them suitable for data that is evolving, nested, or hierarchical.

- **Scalability and Distribution:** Document databases are designed for horizontal scalability, meaning they can be distributed across multiple servers to handle large volumes of data and high traffic loads. Sharding (distributing data across multiple nodes) and replication (creating copies of data for redundancy and availability) are common techniques used.

- **Querying and Indexing:** Document databases provide powerful query mechanisms to retrieve documents based on various criteria. They support indexing on specific fields to improve query performance. Query languages often allow for querying within the nested structures of the documents.

- **ACID Properties (Varying Degrees):** While some document databases offer ACID properties (Atomicity, Consistency, Isolation, Durability) for transactions, others prioritize performance and scalability, offering eventual consistency instead. The level of ACID support can vary depending on the specific database and its configuration. Many offer "BASE" properties (Basically Available, Soft state, Eventually consistent) as an alternative.

- **Native JSON/BSON Support:** Most document databases natively support JSON (JavaScript Object Notation) or BSON (Binary JSON) formats. This simplifies data serialization and deserialization, making it easier for applications to work with the data.

7. b. Discuss the importance of SEO (Search Engine Optimization) in the context of blogging platforms. (10 Marks)

- **Importance of SEO in Blogging:**
  - **Increased Visibility:** SEO helps improve the ranking of blog posts in search engine results pages (SERPs), making them more visible to potential readers.
  - **Organic Traffic:** Higher rankings lead to increased organic (non-paid) traffic to the blog, bringing in more readers and potential customers.
  - **Targeted Audience:** SEO allows bloggers to target specific keywords and topics, attracting readers who are interested in their content.
  - **Brand Building:** Higher visibility and increased traffic can help build brand awareness and establish the blog as an authority in its niche.
  - **Content Promotion:** SEO helps promote blog content and reach a wider audience.

8. a. Provide examples of common query operations performed on document databases and explain their significance. (10 Marks)

- **Common Query Operations:**
  - **Find/Query:** Retrieve documents based on specific criteria (e.g., find all documents where "author" is "John Doe").
    - **Significance:** Essential for retrieving relevant data from the database.
  - **Insert:** Add new documents to the database.
    - **Significance:** Allows for storing new data in the database.
  - **Update:** Modify existing documents.
    - **Significance:** Enables updating data to reflect changes or corrections.

    - **Delete:** Remove documents from the database.
        - **Significance:** Allows for removing outdated or irrelevant data.
    - **Aggregation:** Perform calculations on a set of documents (e.g., count the number of documents, calculate the average value of a field).
        - **Significance:** Enables summarizing and analyzing data.
    - **Indexing:** Create indexes on fields to speed up queries.
        - **Significance:** Improves query performance by allowing for faster data retrieval.

8. b. Explain the importance of event logging in web applications with examples. (10 Marks)

- **Importance of Event Logging:**
    - **Debugging:** Logs provide detailed information about application behavior, making it easier to identify and fix errors.
    - **Monitoring:** Logs can be used to monitor application performance and identify potential issues.
    - **Security:** Logs can track user activity and detect suspicious behavior.
    - **Auditing:** Logs provide a record of events that can be used for auditing purposes.
    - **Analytics:** Logs can be used to gather data about user behavior and application usage.
- **Examples:**
    - **User Login/Logout:** Log successful and failed login attempts, including timestamps and user details.
    - **Order Placement:** Log details of each order placed, including items, customer information, and timestamps.
    - **Error Handling:** Log error messages and stack traces to help debug application issues.
    - **API Calls:** Log details of API calls, including request and response data.
    - **Security Events:** Log suspicious activities, such as unauthorized access attempts or data breaches.

You're right! My apologies. I missed Module 5 in the previous response. Let's address those questions now:

Module 5

9. a. Discuss the key features of graph databases that make them suitable for handling connected data. (10 Marks)

- **Key Features:**

    - **Nodes and Relationships:** Graph databases store data as nodes (entities) and relationships (connections between entities). This directly mirrors real-world relationships.
    - **Relationship-Centric:** Relationships are first-class citizens, not just foreign keys. This allows for efficient traversal and analysis of connections.
    - **Schema-Optional:** While some graph databases have schemas, many are schema-optional, providing flexibility.
    - **Native Graph Processing:** Graph databases use specialized graph query languages (like Cypher in Neo4j) for efficient traversal and manipulation of relationships.
    - **Scalability:** Graph databases can scale horizontally to handle large and complex networks of data.
- **Suitability for Connected Data:**

    - **Social Networks:** Modeling users and their connections.
    - **Recommendation Engines:** Finding related products or content based on user preferences and connections.
    - **Knowledge Graphs:** Representing entities and their relationships in a specific domain.
    - **Network Management:** Analyzing network topology and dependencies.

    - **Supply Chain Management:** Tracking products and their movement through the supply chain.

9. b. Identify scenarios where using a graph database may not be appropriate. What are the limitations? (10 Marks)

- **Scenarios Where Graph Databases May Not Be Appropriate:**
    - **Simple Lookup Tables:** If your data is primarily simple key-value lookups with no complex relationships, a key-value store or relational database might be more efficient.
    - **Transactional Systems Requiring Strong ACID Properties:** While graph databases are improving in transaction management, traditional RDBMS might be better suited for applications with very strict ACID requirements, especially if the relationships are not a primary focus.
    - **Data Warehousing and Analytics with Complex Aggregations:** If your primary need is complex analytical queries involving aggregations across the entire dataset, a columnar database or data warehouse might be more appropriate.
- **Limitations:**
    - **Learning Curve:** Graph query languages (like Cypher) can have a learning curve for those used to SQL.
    - **Tooling Maturity:** The ecosystem of tools and libraries for graph databases is still maturing compared to more established relational databases.
    - **Complex Query Optimization:** Optimizing complex graph queries can be challenging.
    - **Distributed Transactions:** Implementing distributed transactions across a large graph can be complex.

10. a. Describe how transactions are handled in graph databases. What ACID properties are implemented? (10 Marks)

- **Transaction Handling in Graph Databases:**
    - Most graph databases support transactions to ensure data consistency and integrity.
    - Transactions group a set of operations (creates, updates, deletes) into a single unit of work.
    - If any operation within a transaction fails, the entire transaction is rolled back.
- **ACID Properties Implementation:**
    - **Atomicity:** Transactions are atomic, meaning all changes within a transaction are either committed or rolled back entirely.

    - **Consistency:** Transactions maintain the database's consistency by ensuring that all constraints and rules are met.

    - **Isolation:** Transactions are isolated from each other, preventing interference and ensuring that concurrent transactions do not see each other's uncommitted changes.

    - **Durability:** Once a transaction is committed, the changes are persistent and survive even system failures.

    - **Note:** The specific implementation and level of ACID compliance may vary among different graph databases. Some may prioritize performance over strict ACID guarantees in certain scenarios.

10. b. Provide examples of complex queries that can be efficiently executed in graph databases. (10 Marks)

- **Complex Queries Efficiently Executed in Graph Databases:**
  - **Finding the shortest path between two nodes:** Efficiently determine the shortest route in a network (e.g., navigation, social connections).
  - **Identifying communities or clusters:** Discover groups of interconnected nodes (e.g., social network analysis, customer segmentation).
  - **Calculating the influence or centrality of a node:** Determine the importance of a node within a network (e.g., identifying key influencers in social media).
  - **Finding patterns or cycles in a graph:** Detect recurring patterns or loops in relationships (e.g., fraud detection, supply chain analysis).
  - **Performing recommendations based on relationships:** Suggest related items or content based on user preferences and connections (e.g., product recommendations, friend suggestions).