USN :_____ **Internal Assessment Test 2** CMRIT

Sub:	NOSQL Databases	Subject Code:	BCD515C	Branch :	CS & DS
Date:		Duration: 90 min	Marks : 50	Sem : V	OBE

Answer any FIVE question	Marks	CO	RBT
1. Explain Map-Reduce with an example. What are the challenges and use cases associated in map-reduce Technique?	4 + 6	CO3	L2
2. What are the kay value stores? Explain how all the data is stored in a single bucket of key value data store? What are the pros and cons ?	2 + 4+ 4	CO3	L2
3. What are Document databases? Elaborate the suitable cases of document databases.	2 + 8	CO4	L3
4. With suitable diagrams, explain horizontal shading in mongoDB for adding a new node to an existing replica-set and each shard is a replica set.	10	CO4	L3
5. What are graph databases? Write the query performed in graph databases.	2 + 8	CO4	L2
6. With a neat diagram, explain the terms property, relationships and traversing a graph with a query.	5 + 5	CO4	L3

ANSWERS

1. Map-Reduce is a data processing model designed to perform operations on large datasets and produce aggregated results. It provides a flexible and scalable way to handle complex data transformations using two main functions: map and reduce. The map function is used to group all the data based on the key-value and the reduce function is used to perform operations on the mapped data. So, the data is independently mapped and reduced in different spaces and then combined in the function and the result will be saved to the specified new collection. Use Case in E-Commerce: This calculation can help e-commerce platforms:

• Analyze the performance of product categories.
• Identify high-revenue categories for targeted promotions.
• Provide insights into sales distribution for inventory management.

Similar Calculations:

• Average Order Value by Category: Replace sum with average in the reduce function.
• Top-Selling Products: Emit product IDs as keys and calculate totals for each.

2. A Key-Value Store is a type of NoSQL database that stores data as a collection of key-value pairs. Each piece of data (value) is associated with a unique key, which is used to retrieve or update the data efficiently. These stores are widely used for caching, real-time analytics, and distributed applications due to their high performance and scalability. In a key-value store, all data can be stored in a single bucket (or namespace), where each item is uniquely identified by its key. A bucket is a logical container that holds key-value pairs. The key serves as an identifier, and the value can be a simple data type (string, integer) or a complex data structure (JSON, BLOB, XML, etc.).

3. A document database is a type of NoSQL database that stores data in a semi-structured format, usually as JSON, BSON, or XML documents. Unlike relational databases, which store data in

tables with fixed schemas, document databases allow flexible and nested data storage. Each document is self-contained, meaning it stores all necessary information in a single unit. Documents can have different structures and fields, making them ideal for applications requiring flexibility.

4. MongoDB uses horizontal sharding (partitioning data across multiple servers) to handle large datasets and ensure scalability. Each shard in MongoDB is a replica set, meaning it has multiple nodes for redundancy and fault tolerance.
5. A Graph Database is a type of NoSQL database that represents and stores data as nodes (entities) and edges (relationships) instead of tables. It is optimized for handling highly connected data and complex relationships efficiently.

Key Components of a Graph Database:

Nodes: Represent entities (e.g., people, products, locations).

Edges: Represent relationships between nodes (e.g., "friend of," "bought," "connected to").

Properties: Store information about nodes and edges.

Labels: Categorize nodes into types (e.g., Person, Product).

6. Graph databases use a structure of nodes, relationships, and properties to represent and query data efficiently. Let's break down these fundamental concepts:

Query for properties: CREATE (a:Person {name: 'Alice', age: 28, city: 'Kanpur'})
RETURN a

Query for relationships: MATCH (a:Person {name: 'Alice'}), (b:Person {name: 'Bob'})
CREATE (a)-[:FRIEND {since: 2020}]->(b)
RETURN a, b

Query for traversing: MATCH (a:Person {name: 'Alice'})-[:FRIEND]->(b)
RETURN b.name