CAMBRIDGE INSTITUTE OF TECHNOLOGY

An Autonomous Institution

K.R. PURAM, BENGALURU – 560 036, Ph: 080-2561 8798 / 2561 8799

Fax: 080-2561 8789, email: principal@cambridge.edu.in

Affiliated to VTU, Belagavi| Approved by AICTE, New Delhi |NAAC A+ & NBA Accredited|

NIRF Innovation Ranked, UGC 2(f), ISO Certified| Recognized by Govt. of Karnataka

Vision

To become a premier institute transforming our students to be global professionals.

Mission

M1: Develop competent Human Resources, and create state-of-the-art infrastructure to impart quality education and to support research.

M2: Adopt tertiary approach in teaching – learning pedagogy that transforms students to become competent professionals and entrepreneurs.

M3: Nurture and train students to develop the qualities of global professionals.

MongoDB LAB MANUAL
(BDSL456B)

(As per Visvesvaraya Technological University Syllabus)

Compiled by:

Mr. Sreekantha B
Assistant Professor
Dept. of AIML

Name:_____

USN :_____

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

2023-24

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Vision

To enrich the academic acumen in the field of artificial intelligence and machine learning coupled with providing sustainable solutions to global challenges.

Mission

M1: Adhere to Indian pedagogy of Guru Shishya Parambara in modern technological endeavors

M2: Design and develop adequate infrastructure for patentable technologies resulting in useful products

M3: Equip the students in fulfilling duty of serving the nation by ensuring self-reliance in technological prowess

PROGRAM EDUCATIONAL OUTCOMES (PEOs)

PEO1: To provide students learn skills in the domain of artificial intelligence and machine learning.

PEO2: To strengthen students' knowledge for higher studies in premier institutes in India and abroad.

PEO3: To equip students to become entrepreneurs by solving challenging global problems.

PEO4: To augment new frontiers of research in artificial intelligence and machine learning through systematic process.

PEO5: To enable the academic community to render its service to the nation by reducing technological barriers in the domain.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: To gain fundamental knowledge of computing paradigms with the intricacies through laboratory Experiments

PSO2: To design and develop artificial intelligence systems such as mobile robots, drones and intelligence software systems

PSO3: To use intelligence system with machine learning to solve challenging problems in defence, Healthcare and industrial applications.

Program Outcomes

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Sl No	Experiments
1	a. Illustration of Where Clause, AND,OR operations in MongoDB. b. Execute the Commands of MongoDB and operations in MongoDB : Insert, Query, Update, Delete and Projection. (Note: use any collection)
2	a. Develop a MongoDB query to select certain fields and ignore some fields of the documents from any collection. b. Develop a MongoDB query to display the first 5 documents from the results obtained in a. [use of limit and find]
3	a. Execute query selectors (comparison selectors, logical selectors) and list out the results on any collection b. Execute query selectors (Geospatial selectors, Bitwise selectors) and list out the results on any collection
4	Create and demonstrate how projection operators ($, $elematch and $slice) would be used in the MondoDB.
5	Execute Aggregation operations ($avg, $min,$max, $push, $addToSet etc.). students encourage to execute several queries to demonstrate various aggregation operators)
6	Execute Aggregation Pipeline and its operations (pipeline must contain $match, $group, $sort, $project, $skip etc. students encourage to execute several queries to demonstrate various aggregation operators)
7	a. Find all listings with listing_url, name, address, host_picture_url in the listings And Reviews collection that have a host with a picture url b. Using E-commerce collection write a query to display reviews summary.
8	a. Demonstrate creation of different types of indexes on collection (unique, sparse, compound and multikey indexes) b. Demonstrate optimization of queries using indexes.
9	a. Develop a query to demonstrate Text search using catalog data collection for a given word b. Develop queries to illustrate excluding documents with certain words and phrases
10	Develop an aggregation pipeline to illustrate Text search on Catalog data collection.

DATABASE

A database is a structured collection of data that is organized and stored in a computer system. It allows for efficient storage, retrieval, and manipulation of data. Databases are used in a wide range of applications, including websites, business applications, scientific research, and more.

There are various types of databases, including:

1. **Relational Databases**: These are the most common type of database, where data is organized into tables with rows and columns. Relational databases use Structured Query Language (SQL) for querying and managing data. Examples include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.
2. **NoSQL Databases**: NoSQL (Not Only SQL) databases are designed to handle large volumes of unstructured or semi-structured data. They provide more flexibility and scalability compared to relational databases and are often used in big data and real-time web applications. Examples include MongoDB, Cassandra, Redis, and Couchbase.
3. **Graph Databases**: Graph databases are designed to represent and store data as nodes, edges, and properties, which are interconnected. They are used for applications where relationships between data points are important, such as social networks, recommendation systems, and network analysis. Examples include Neo4j, Amazon Neptune, and ArangoDB.
4. **Document Databases**: Document databases store and retrieve data in a semi-structured format, typically using JSON or XML documents. They are suitable for content management systems, e-commerce platforms, and other applications where data varies in structure. Examples include MongoDB, Couchbase, and RavenDB.
5. **Columnar Databases**: Columnar databases store data in columns rather than rows, which allows for efficient data retrieval and analysis, especially for analytical queries on large datasets. They are commonly used in data warehousing and business intelligence applications. Examples include Google BigQuery, Amazon Redshift, and Apache Cassandra.

Databases play a crucial role in modern computing by enabling organizations to efficiently manage and utilize large volumes of data for various purposes, such as decision-making, analytics, and application development.

RDBMS VS DOCUMENT DATABASE

Relational Database Management Systems (RDBMS) and Document Databases are two different types of database management systems, each with its own set of characteristics and use cases. Here are some key differences between the two:

1. **Data Model**:
    - RDBMS: RDBMS uses a structured data model based on tables with rows and columns. Data is organized into tables with predefined schemas, and relationships between tables are established using foreign keys.
    - Document Database: Document databases use a semi-structured data model where data is stored in documents, typically in formats like JSON or BSON. Each document can have its own structure, and there is no fixed schema across the entire database.
2. **Schema**:
    - RDBMS: RDBMS typically requires a predefined schema, where the structure of each table and the relationships between tables must be specified before inserting data. Changes to the schema can be complex and often require downtime.
    - Document Database: Document databases are schema-less or schema-flexible. Documents within the same collection (equivalent to a table in RDBMS) can have different structures, and new fields can be added dynamically without requiring a predefined schema.
3. **Query Language**:
    - RDBMS: RDBMS typically uses SQL (Structured Query Language) for querying and manipulating data. SQL is powerful and standardized across most relational databases.
    - Document Database: Document databases often use query languages specific to the database, such as MongoDB's query language or Couchbase's N1QL. These languages are designed to work with semi-structured data and allow for querying nested fields within documents.
4. **Scaling**:
    - RDBMS: Scaling RDBMS horizontally (across multiple servers) can be challenging, especially for large-scale applications. Vertical scaling (increasing the resources of a single server) is a common approach for RDBMS.
    - Document Database: Document databases are often designed to scale horizontally with ease. They can distribute data across

multiple nodes and handle large volumes of data and high read/write loads efficiently.

5. **Use Cases**:
    - RDBMS: RDBMS are well-suited for applications where data has a clearly defined structure, and ACID (Atomicity, Consistency, Isolation, Durability) transactions are required. Common use cases include traditional transactional systems, such as financial applications and e-commerce platforms.
    - Document Database: Document databases are ideal for applications with rapidly evolving schemas, unstructured or semi-structured data, and where high performance on read and write operations is crucial. Use cases include content management systems, real-time analytics, and applications requiring flexible data models.

In summary, while both RDBMS and Document Databases are types of database management systems, they differ in their data models, schemas, query languages, scaling capabilities, and use cases. The choice between them depends on factors such as the nature of the data, scalability requirements, and the flexibility of the application's data model.

MongoDB is a popular open-source NoSQL database management system that provides high performance, high availability, and easy scalability.
Here's a comprehensive overview of MongoDB:

Introduction to MongoDB:

- MongoDB is a document-oriented NoSQL database.
- Developed by MongoDB Inc.
- Initially released in 2009.
- Written in C++, C, and .

Features:

1. **Document-Oriented:** MongoDB stores data in flexible, JSON-like documents called BSON (Binary JSON), making it easy to store and retrieve complex data structures.
2. **Schema-less:** Unlike traditional relational databases, MongoDB does not require a predefined schema, allowing for flexibility in data structure.
3. **High Performance:** MongoDB provides high-speed read and write operations by utilizing internal memory for storing working sets of data.

4. **High Availability:** Supports replication and automated failover, ensuring data availability in case of hardware failure or network partition.
5. **Horizontal Scalability:** MongoDB allows for horizontal scaling through sharding, distributing data across multiple servers to handle large volumes of data and traffic.
6. **Rich Query Language:** MongoDB supports a rich query language with features like secondary indexes, aggregations, text search, and geospatial queries.
7. **Aggregation Framework:** Provides an aggregation pipeline for processing and transforming documents in the database.
8. **Indexing:** Supports various types of indexes including compound indexes, geospatial indexes, text indexes, and hashed indexes to improve query performance.
9. **Security:** Offers authentication, authorization, encryption, and auditing features to ensure data security and compliance.
10. **Ad Hoc Queries:** Allows ad hoc queries on documents using a query language similar to SQL.

Components:

1. **MongoDB Server:** The core database server responsible for data storage, retrieval, and manipulation.
2. **MongoDB Shell:** A command-line interface for interacting with MongoDB instances using -like syntax.
3. **MongoDB Compass:** A graphical user interface (GUI) tool for visually exploring and interacting with MongoDB databases.
4. **Drivers:** MongoDB provides official and community-supported drivers for various programming languages like Python, Java, Node.js, etc., to interact with the database.
5. **Replica Set:** A group of MongoDB instances that maintain identical data sets for redundancy and fault tolerance.
6. **Sharding:** MongoDB clusters data across multiple servers to support horizontal scalability.

Use Cases:

- **Content Management:** MongoDB's flexible schema and scalability make it suitable for content management systems handling diverse types of content.
- **Real-Time Analytics:** MongoDB's high-performance read and write operations make it suitable for real-time analytics and logging applications.

- **Catalog and Product Data:** MongoDB can handle complex product catalogs with varying attributes and structures efficiently.
- **Mobile and Social Infrastructure:** MongoDB is well-suited for mobile and social applications due to its ability to handle large volumes of unstructured data.
- **IoT Data Storage:** MongoDB can efficiently store and analyze large volumes of data generated by IoT devices.

MongoDB is a powerful NoSQL database system that offers flexibility, scalability, and performance, making it suitable for a wide range of applications from small-scale projects to enterprise-level deployments. Its document-oriented nature, rich query capabilities, and ease of scalability make it a popular choice among developers and organizations seeking a modern database solution

Real Time Applications

MongoDB, being a highly flexible and scalable NoSQL database, is widely used in various real-time applications due to its ability to handle large volumes of data, high availability, and horizontal scalability. Here are some real-time applications where MongoDB is commonly used:

1. **Social Media Platforms**: Social media platforms deal with a huge amount of real-time data such as user interactions, posts, comments, likes, etc. MongoDB's flexibility allows for quick storage and retrieval of such data. It's often used for activity feeds, user profiles, messaging systems, and analytics.
2. **IoT (Internet of Things)**: MongoDB is used in IoT applications for storing sensor data, device telemetry, and real-time analytics. With its ability to handle time-series data efficiently, MongoDB is suitable for applications such as smart home systems, industrial monitoring, and vehicle tracking.
3. **Real-Time Analytics**: MongoDB is used in applications that require real-time analytics such as monitoring user behavior, tracking website traffic, and analyzing application logs. Its ability to handle semi-structured and unstructured data makes it suitable for these use cases.
4. **Online Gaming**: MongoDB is used in online gaming platforms for player profiles, game state management, leaderboards, and real-time messaging. Its scalability allows gaming companies to handle millions of concurrent users and provide a seamless gaming experience.
5. **Financial Services**: MongoDB is used in financial applications for real-time risk analysis, fraud detection, and trade processing. Its

ability to handle high volumes of data with low latency makes it suitable for applications that require real-time decision-making.

6. **Real-Time Collaboration Tools**: MongoDB is used in collaboration tools such as project management platforms, document editing tools, and messaging applications. It allows multiple users to collaborate in real-time by storing and synchronizing data efficiently.
7. **Content Management Systems (CMS)**: MongoDB is used in CMS applications for storing and serving content in real-time. Its flexible schema allows content creators to store various types of content such as text, images, videos, and metadata.
8. **E-commerce**: MongoDB is used in e-commerce applications for product catalog management, inventory tracking, and real-time order processing. Its ability to handle complex product data and high traffic volumes makes it suitable for e-commerce platforms.

These are just a few examples of real-time applications where MongoDB is commonly used. Its flexibility, scalability, and performance make it a popular choice for a wide range of real-time use cases across different industries.

MongoDB is a popular NoSQL database that uses a document-oriented data model. It provides various query capabilities to retrieve and manipulate data. Below, I'll cover MongoDB queries from basic to advanced levels:

1. **Basic Queries**:
   - **Find**: The find() method retrieves documents from a collection. It takes a query object as a parameter to specify selection criteria.

- db.collection_name.find({ key: value })
- **Example**: db.users.find({ name: "John" }) retrieves all documents where the name field is equal to "John".

 Projection:

- The projection parameter in find() allows you to specify which fields to include or exclude in the result set.

- db.collection_name.find({ key: value }, { field1: 1, field2: 1, ... })
- **Example**: db.users.find({}, { name: 1, age: 1 }) retrieves only the name and age fields from the users collection.

 Query Operators:

- MongoDB provides various query operators for complex queries:
  - Comparison Operators ($eq, $gt, $lt, etc.)
  - Logical Operators ($and, $or, $not, $nor)

  - Element Operators ($exists, $type)
  - Array Operators ($in, $all, $elemMatch)
- **Example**: db.users.find({ age: { $gt: 25, $lt: 40 } }) retrieves users with ages between 25 and 40.

Sorting:

- The sort() method sorts documents in the result set based on specified criteria.
- db.collection_name.find().sort({ field: 1 }) // Ascending
- db.collection_name.find().sort({ field: -1 }) // Descending
- **Example**: db.users.find().sort({ age: -1 }) sorts users by age in descending order.

Limiting Results:

- The limit() method limits the number of documents returned in the result set
- db.collection_name.find().limit(number)
- **Example**: db.users.find().limit(10) retrieves the first 10 users.

Skipping Results:

- The skip() method skips a specified number of documents and returns the rest.

- db.collection_name.find().skip(number)
- **Example**: db.users.find().skip(5) skips the first 5 users.

Indexing:

- Indexes improve query performance by making data retrieval faster.
- Create indexes using the createIndex() method.

- db.collection_name.createIndex({ field: 1 })
- **Example**: db.users.createIndex({ name: 1 }) creates an index on the name field.

Aggregation:

MongoDB's aggregation framework allows you to perform advanced data processing operations.

db.collection_name.aggregate([{ stage1 }, { stage2 }, ...])
Stages include $match, $group, $project, $sort, $limit, etc.

Example:

```
db.orders.aggregate([
  { $match: { status: "delivered" } },
  { $group: { _id: "$product", total: { $sum: "$quantity" } } }
])
```

- **Text Search**:

  MongoDB provides text search capabilities for full-text search queries.

  db.collection_name.find({ $text: { $search: "search_query" } })

  Example: db.articles.find({ $text: { $search: "MongoDB tutorial" } })

Geospatial Queries:

- MongoDB supports geospatial queries for location-based data.

```
db.collection_name.find({
  location: {
    $near: {
      $geometry: { type: "Point", coordinates: [longitude, latitude] },
      $maxDistance: distance_in_meters
    }
  }
})
```

- **Example**:

```
db.places.find({
  location: {
    $near: {
      $geometry: { type: "Point", coordinates: [-73.97, 40.77] },
      $maxDistance: 1000
    }
  }
```

Experiments

Experiment 1:

a. Illustration of Where Clause, AND, OR operations in MongoDB:

```
// Find documents where age is greater than 25 and gender is "male"
db.collection.find({ $and: [{ age: { $gt: 25 } }, { gender: "male" }] })

// Find documents where age is greater than 30 or gender is "female"
db.collection.find({ $or: [{ age: { $gt: 30 } }, { gender: "female" }] })
```

b. Execute the Commands of MongoDB and operations in MongoDB:

```
// Insert document
db.collection.insertOne({ name: "John", age: 35, gender: "male" })

// Query document
db.collection.find({ age: { $gt: 30 } })

// Update document
db.collection.updateOne({ name: "John" }, { $set: { age: 40 } })

// Delete document
db.collection.deleteOne({ name: "John" })

// Projection
db.collection.find({}, { name: 1, age: 1 })
```

Experiment 2:

a. Develop a MongoDB query to select certain fields and ignore some fields of the documents:

```
// Select certain fields and ignore some fields
db.collection.find({}, { name: 1, age: 1, _id: 0 })
```

b. Develop a MongoDB query to display the first 5 documents:

```
// Display the first 5 documents
db.collection.find().limit(5)
```

Experiment 3:

a. Execute query selectors (comparison selectors, logical selectors):

```
// Comparison selectors
db.collection.find({ age: { $gt: 30 } })

// Logical selectors
db.collection.find({ $and: [{ age: { $gt: 25 } }, { gender: "male" }] })
```

b. Execute query selectors (Geospatial selectors, Bitwise selectors):

```
// Geospatial selectors
db.collection.find({ location: { $near: { $geometry: { type: "Point", coordinates: [ -73.9667, 40.78 ] }, $maxDistance: 1000 } } })

// Bitwise selectors
db.collection.find({ flags: { $bitsAllSet: 4 } })
```

Experiment 4:

Create and demonstrate how projection operators would be used:

```
// $ projection operator
db.collection.find({}, { "grades.$": 1 })

// $elemMatch projection operator
db.collection.find({}, { grades: { $elemMatch: { score: { $gt: 80 } } } })

// $slice projection operator
db.collection.find({}, { grades: { $slice: 2 } })
```

Experiment 5:

Execute Aggregation operations:

```
// $avg
```

```
db.collection.aggregate([{ $group: { _id: null, avgAge: { $avg: "$age" } } }])

// $min
db.collection.aggregate([{ $group: { _id: null, minAge: { $min: "$age" } } }])

// $max
db.collection.aggregate([{ $group: { _id: null, maxAge: { $max: "$age" } } }])

// $push
db.collection.aggregate([{ $group: { _id: null, allNames: { $push: "$name" } } }])

// $addToSet
db.collection.aggregate([{ $group: { _id: null, uniqueNames: { $addToSet: "$name" } } }])
```

Experiment 6:

Execute Aggregation Pipeline and its operations:

```
// Aggregation Pipeline
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $group: { _id: "$gender", count: { $sum: 1 } } },
  { $sort: { count: -1 } },
  { $skip: 2 },
  { $project: { _id: 0, gender: "$_id", count: 1 } }
])
```

Experiment 7:

a. Find all listings with listing_url, name, address, host_picture_url:

```
db.listingsAndReviews.find({}, { listing_url: 1, name: 1, address: 1, host_picture_url: 1 })
```

b. Using E-commerce collection write a query to display reviews summary:

```
db.eCommerce.aggregate([
```

```
  { $group: { _id: "$product_id", avgRating: { $avg: "$rating" }, totalReviews: { $sum: 1 } } }
])
```

Experiment 8:

a. Demonstrate creation of different types of indexes on collection:

```
// Unique index
db.collection.createIndex({ username: 1 }, { unique: true })

// Sparse index
db.collection.createIndex({ city: 1 }, { sparse: true })

// Compound index
db.collection.createIndex({ age: 1, city: -1 })

// Multikey index
db.collection.createIndex({ tags: 1 })
```

b. Demonstrate optimization of queries using indexes:

```
// Using index
db.collection.find({ username: "john" }).hint({ username: 1 })

// Explain query execution plan
db.collection.find({ age: { $gt: 30 } }).explain()
```

Experiment 9:

a. Develop a query to demonstrate Text search using catalog data collection for a given word:

```
db.catalog.find({ $text: { $search: "keyword" } })
```

b. Develop queries to illustrate excluding documents with certain words and phrases:

```
db.collection.find({ $text: { $search: "keyword -excludedWord" } })
```

Experiment10:

Develop an aggregation pipeline to illustrate Text search on Catalog data collection

```
db.Catalog.aggregate([
  {
    $search:  {
      index: 'default', // The name of the search index to use
      text: {
        query: 'your_search_query_here', // The text to search for
        path: 'description' // The field in the documents to search
      }
    }
  },
  {
    $project:  {
      _id: 0, // Exclude the _id field from the output
      productName: 1, // Include the productName field in the output
      description: 1, // Include the description field in the output
      score: { $meta: 'searchScore' } // Include the search score in the output
    }
  }
])
```

Viva Questions

1. What are some of the advantages of MongoDB?
2. What is a Document in MongoDB?
3. What is a Collection in MongoDB?
4. What are Databases in MongoDB?
5. What is the Mongo Shell?
6. What are some features of MongoDB
7. How do you Update a Document?
8. How do you Delete a Document?
9. What are the data types in MongoDB
10. Explain the term "Indexing" in MongoDB.
11. What are Geospatial Indexes in MongoDB
12. What do you mean by Transactions
13. What is the Aggregation Framework in MongoDB
14. Explain the concept of pipeline in the MongoDB aggregation framework.
15. What are some utilities for backup and restore in MongoDB
16. Explain the Replication Architecture in MongoDB.

****