Name : Bhavani Rajpurohit

Class : AIA-3 Subject: DBMS LAB

Roll No : 2213688 Batch : B

ASSIGNMENT NO: 08

Title:

Basic Operations (CRUD Operations) on NoSQL Databases like MongoDB, Cassandra Graph Database (NEO4j)

Aim:

Understand the basic operations (Create, Read, Update, and Delete - CRUD) on NoSQL databases such as MongoDB, Cassandra, and NEO4j.

Software Required:

- MongoDB
- Cassandra
- NEO4j

Theory:-

NoSQL databases are a type of database management system that provide a flexible and scalable approach to data storage and retrieval. They differ from traditional relational databases by prioritizing horizontal scalability, performance, and ease of development over strict data consistency and structured querying. Here is an overview of NoSQL databases and their characteristics:

Schema-less Structure: NoSQL databases are schema-less, meaning they do not enforce a fixed schema for data. This allows for dynamic and flexible data modeling, where different records can have varying structures, making it easier to handle evolving data requirements.

Horizontal Scalability: NoSQL databases are designed to scale horizontally by distributing data across multiple servers or clusters. This enables them to handle large volumes of data and high

traffic loads by adding more machines to the system.

High Performance: NoSQL databases prioritize performance by employing various techniques, such as in-memory caching and optimized data storage formats. They are often optimized for specific use cases, such as read-heavy or write-heavy workloads.

Flexible Data Models: NoSQL databases support a variety of data models, including key-value, document, columnar, and graph. This allows developers to choose the most suitable data model for their specific application requirements.

Key-Value Stores: Simplest form of NoSQL database, where data is stored as a collection of key-value pairs. They provide fast key-based retrieval but lack complex querying capabilities.

Document Databases: Store semi-structured data as documents, typically in formats like JSON or XML. They provide rich querying and indexing capabilities, allowing for flexible data retrieval.

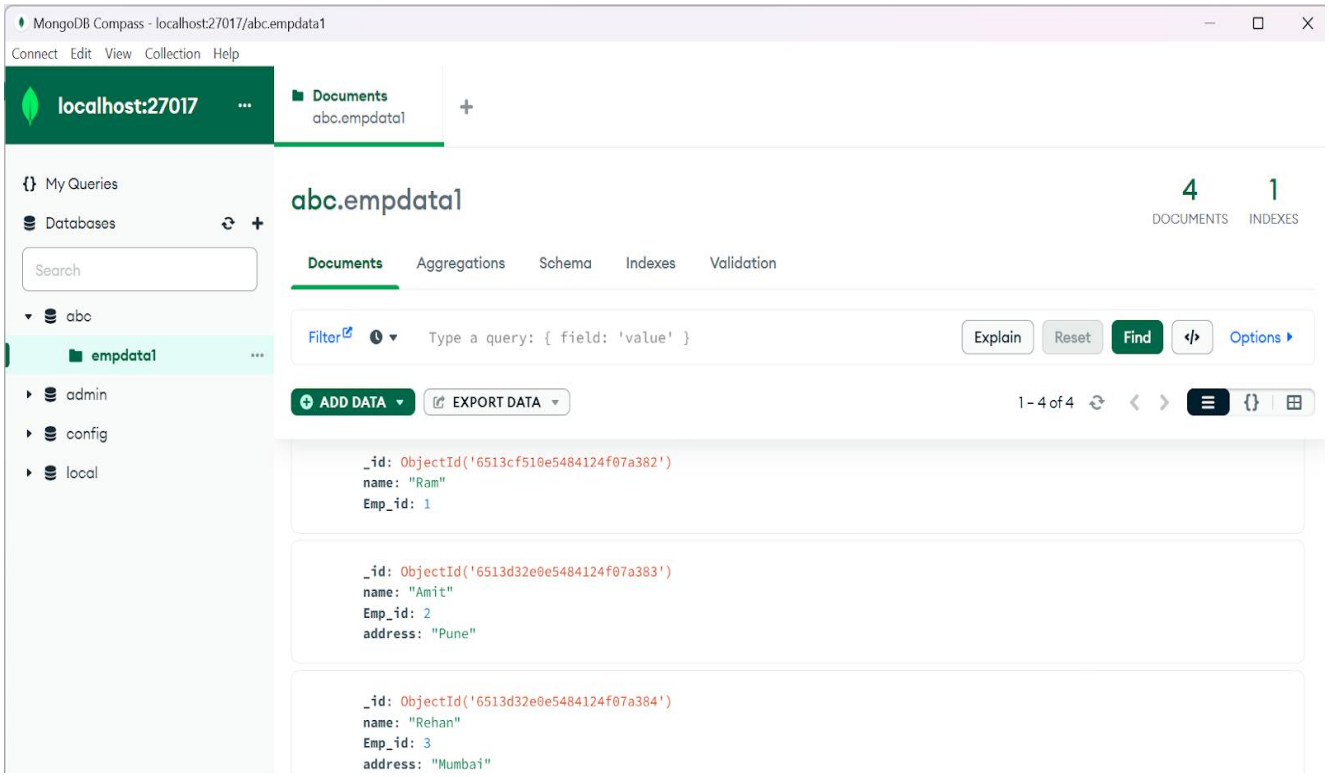
Columnar Databases: Organize data in columns rather than rows, enabling efficient storage and retrieval of large datasets. They are often used for analytical and big data workloads.

Graph Databases: Optimize for managing highly connected data, such as social networks or recommendation engines. They use graph structures to represent relationships between entities and offer powerful graph traversal queries.

Eventual Consistency: NoSQL databases often adopt an eventual consistency model, where data changes propagate asynchronously and may take some time to be fully synchronized across distributed nodes. This trade-off allows for improved scalability and availability but sacrifices immediate data consistency.

Distributed Architecture: NoSQL databases are designed to operate in distributed environments, with data partitioned and replicated across multiple nodes. This distribution ensures fault tolerance, data redundancy, and high availability.

Polyglot Persistence: NoSQL databases embrace the concept of polyglot persistence, which means using different storage technologies for different data needs within an application. Instead of a one-size-fits-all approach, developers can choose the most appropriate NoSQL database for each specific use case.




```
abc> show collections

abc> db.empdata1.insertOne({ name:"Ram",Emp_id:1})
{
  acknowledged: true,
  insertedId: ObjectId("6513cf510e5484124f07a382")
}
abc> show collections
empdata1
abc> db.empdata.find()

abc> show collections
empdata1
abc> db.empdata1.find()
[
  { _id: ObjectId("6513cf510e5484124f07a382"), name: 'Ram', Emp_id: 1 }
]
abc> db.empdata1.find().pretty
[Function: pretty] {
  returnType: 'this',
  serverVersions: [ '0.0.0', '999.999.999' ],
  apiVersions: [ 0, Infinity ],
  topologies: [ 'ReplSet', 'Sharded', 'LoadBalanced', 'Standalone' ],
  returnsPromise: false,
  deprecated: false,
  platforms: [ 'Compass', 'Browser', 'CLI' ],
  isDirectShellCommand: false,
  acceptsRawInput: false,
  shellCommandCompleter: undefined,
  help: [Function (anonymous)] Help
}
abc> db.empdata1.find().pretty()
```

```
abc> db.empdata1.find().pretty()
[
  { _id: ObjectId("6513cf510e5484124f07a382"), name: 'Ram', Emp_id: 1 }
]
abc> db.empdata1.insertMany({name:"Amit",Emp_id:2,address:"Pune"},{name:"Rehan",Emp_id:3,address:"Mumbai"},{name:"Mohit",Emp_id:4,add
ress:"Pune"}])
Uncaught:
SyntaxError: Unexpected token, expected "," (1:140)

> 1 | db.empdata1.insertMany({name:"Amit",Emp_id:2,address:"Pune"},{name:"Rehan",Emp_id:3,address:"Mumbai"},{name:"Mohit",Emp_id:4,ad
dress:"Pune"}])
    |
            ^
  2 |

abc> db.empdata1.insertMany([{name:"Amit",Emp_id:2,address:"Pune"},{name:"Rehan",Emp_id:3,address:"Mumbai"},{name:"Mohit",Emp_id:4,ad
dress:"Pune"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6513d32e0e5484124f07a383"),
    '1': ObjectId("6513d32e0e5484124f07a384"),
    '2': ObjectId("6513d32e0e5484124f07a385")
  }
}
abc> db.empdata1.find().pretty()
[
  { _id: ObjectId("6513cf510e5484124f07a382"), name: 'Ram', Emp_id: 1 },
  {
    _id: ObjectId("6513d32e0e5484124f07a383"),
    name: 'Amit',
    Emp_id: 2,
```

```
    name: 'Amit',
    Emp_id: 2,
    address: 'Pune'
  },
  {
    _id: ObjectId("6513d32e0e5484124f07a384"),
    name: 'Rehan',
    Emp_id: 3,
    address: 'Mumbai'
  },
  {
    _id: ObjectId("6513d32e0e5484124f07a385"),
    name: 'Mohit',
    Emp_id: 4,
    address: 'Pune'
  }
]
```

Conclusion:

Through this experiment, students should learn the basics of NoSQL databases and their usage in performing CRUD operations. They will gain practical experience with popular NoSQL databases like MongoDB, Cassandra, and NEO4j. By comparing these databases, students will understand the strengths and weaknesses of different NoSQL database types.