Experiment No. 8

Name: Sourav Shailesh Toshniwal

Class: TY CSE-8 AIEC-1

Roll no.: 2213047

Batch: A

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.

Procedure:

I. MongoDB:

a. Installation and Setup:

- Visit the official MongoDB website (https://www.mongodb.com) and download the appropriate version for your operating system.
- Follow the installation instructions provided on the MongoDB website to install MongoDB on your machine.
- Start the MongoDB server.

b. CRUD Operations in MongoDB:

Create Operation:

- Open a terminal or command prompt and start the MongoDB shell by entering the command mongo.
- To create a new database, use the command use database_name. Replace database_name with the name of your desired database.
- To create a new collection, use the command db.collection_name.insert(document). Replace collection_name with the name of your collection and document with the JSON document you want to insert.

Read Operation:

- To retrieve all documents in a collection, use the command db.collection_name.find(). This will return all documents in the specified collection.
- To query for specific documents, use the find command with query parameters. For example, db.collection_name.find({ key: value }) will return documents where the specified key has the given value.

Update Operation:

- To update a document, use the command db.collection_name.update(query, update). Replace query with the filter to select the document(s) to update, and update with the modifications you want to make.

Delete Operation:

- To delete documents from a collection, use the command db.collection_name.delete(query). Replace query with the filter to select the document(s) to delete.

II. Cassandra:

a. Installation and Setup:

- Visit the official Apache Cassandra website (https://cassandra.apache.org) and download the appropriate version for your operating system.
- Follow the installation instructions provided on the Cassandra website to install Cassandra on your machine.
- Start the Cassandra server.

b. CRUD Operations in Cassandra:

Create Operation:

- Open a terminal or command prompt and start the Cassandra shell by entering the command cqlsh.
- To create a keyspace (equivalent to a database in Cassandra), use the command CREATE KEYSPACE keyspace_name WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};. Replace keyspace_name with the desired name of your keyspace.

Read Operation:

- To query all rows from a table, use the command SELECT * FROM keyspace_name.table_name;. Replace keyspace_name with your keyspace name and table_name with the name of your table.

Update Operation:

- To update a row in a table, use the command UPDATE keyspace_name.table_name SET column_name = new_value WHERE key = key_value;. Replace keyspace_name, table_name, column_name, new_value, key, and key_value with the appropriate values.

Delete Operation:

- To delete a row from a table, use the command DELETE FROM keyspace_name.table_name WHERE key = key_value;. Replace keyspace_name, table_name, key, and key_value with the appropriate values.

III. NEO4j:

a. Installation and Setup:

- Visit the official NEO4j website (https://neo4j.com) and download the appropriate version for your operating system.
- Follow the installation instructions provided on the NEO4j website to install NEO4j on your machine.
- Start the NEO4j server.

b. CRUD Operations in NEO4j:

Create Operation:

- Open a web browser and navigate to the NEO4j browser interface (usually at http://localhost:7474).
- Use the CREATE clause to create nodes and relationships. For example, CREATE (node_name:Label {key: value}) will create a node with the specified label and properties.

Read Operation:

- To retrieve all nodes in the graph, use the MATCH clause. For example, MATCH (n) RETURN n will return all nodes in the graph.

Update Operation:

- To update a node or relationship, use the SET clause.
- For example, MATCH (n {key: value}) SET n.property = new_value will update the specified property of the matched node.

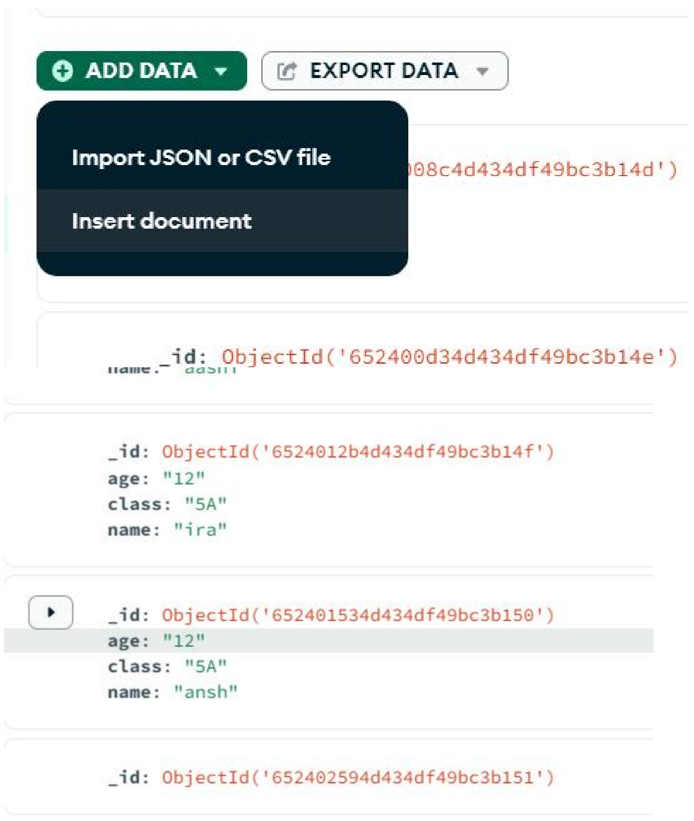
Delete Operation:

- To delete nodes and relationships, use the DELETE clause.
- For example, MATCH (n {key: value}) DELETE n will delete the nodes that match the specified condition.

OUTPUT:

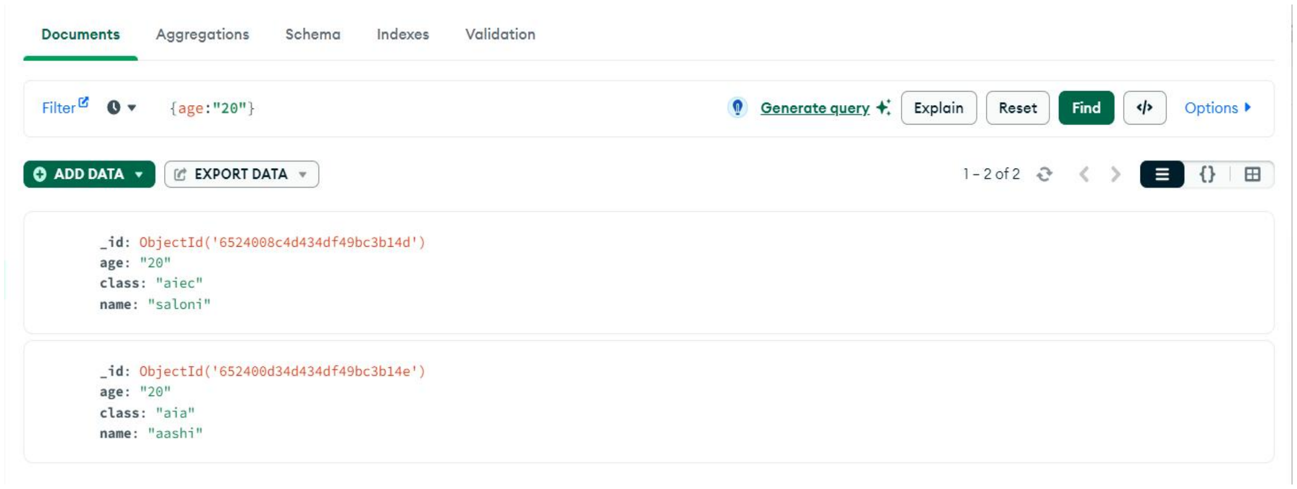
MONGODB USED:

CREATE:



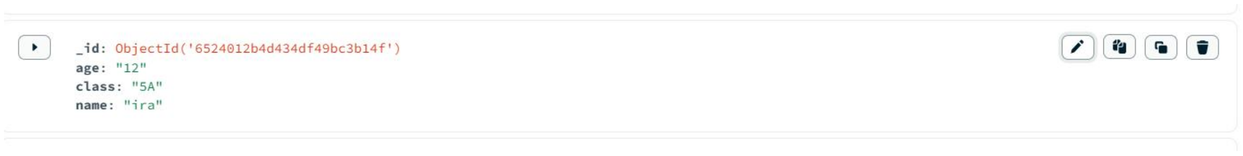
```
)08c4d434df49bc3b14d')

_id: ObjectId('652400d34d434df49bc3b14e')

_id: ObjectId('6524012b4d434df49bc3b14f')
age: "12"
class: "5A"
name: "ira"

_id: ObjectId('652401534d434df49bc3b150')
age: "12"
class: "5A"
name: "ansh"

_id: ObjectId('652402594d434df49bc3b151')
```

READ:

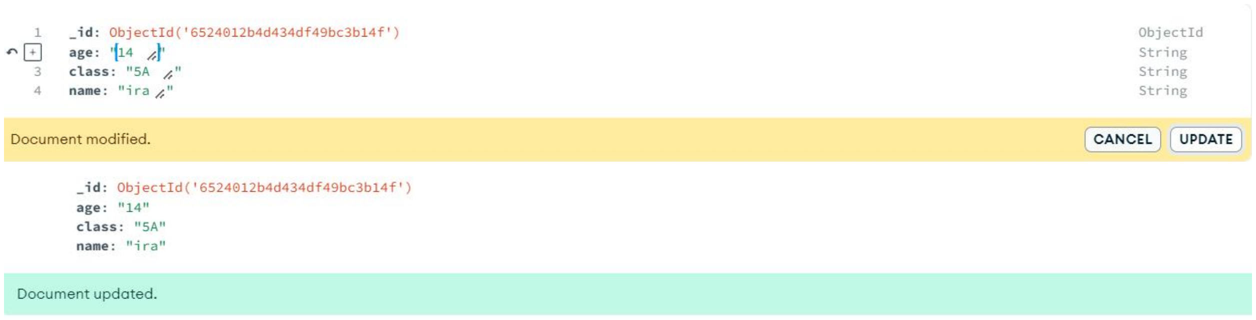


```
{age:"20"}

_id: ObjectId('6524008c4d434df49bc3b14d')
age: "20"
class: "aiec"
name: "saloni"

_id: ObjectId('652400d34d434df49bc3b14e')
age: "20"
class: "aia"
name: "aashi"
```

UPDATE:



```
_id: ObjectId('6524012b4d434df49bc3b14f')
age: "12"
class: "5A"
name: "ira"
```



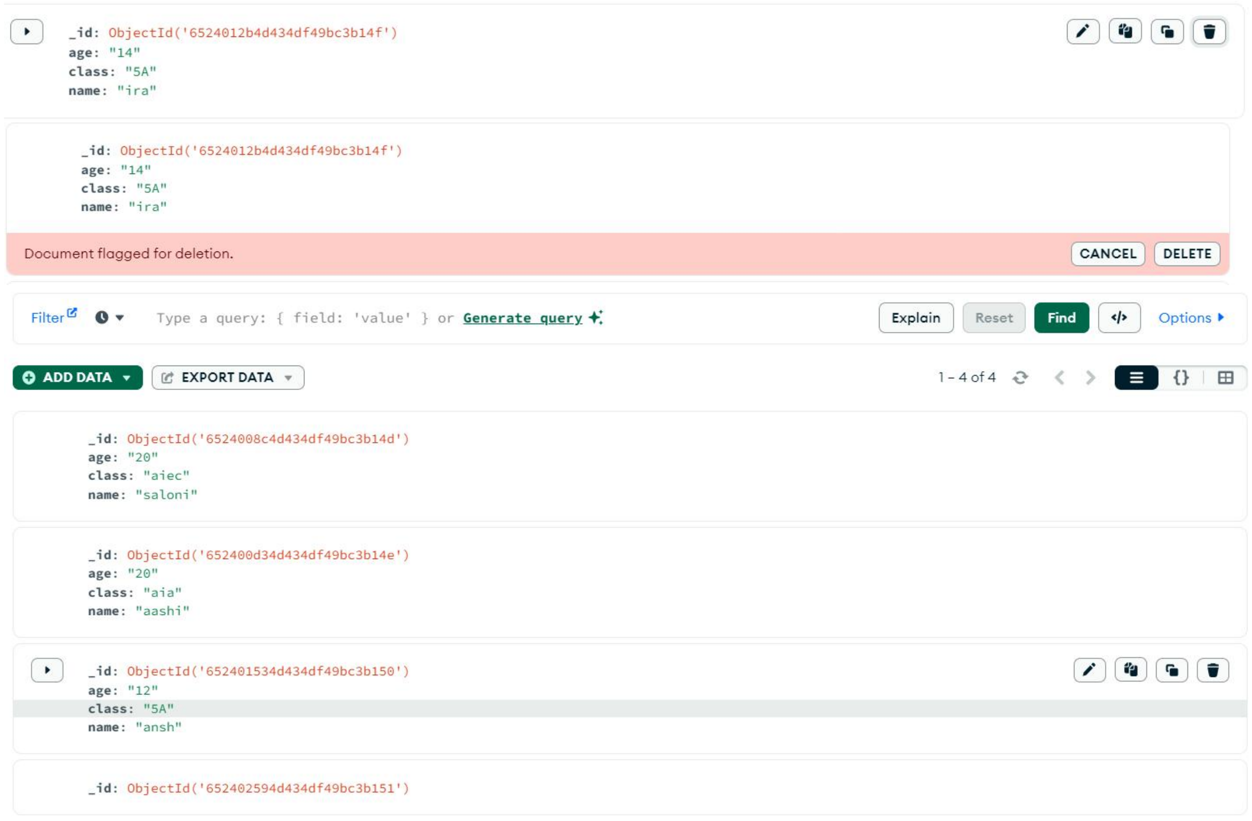
```
_id: ObjectId('6524012b4d434df49bc3b14f')      ObjectId
age: "14"                                      String
class: "5A"                                    String
name: "ira"                                    String
```

Document modified. CANCEL UPDATE

```
_id: ObjectId('6524012b4d434df49bc3b14f')
age: "14"
class: "5A"
name: "ira"
```

Document updated.

DELETE:



```
_id: ObjectId('6524012b4d434df49bc3b14f')
age: "14"
class: "5A"
name: "ira"
```

```
_id: ObjectId('6524012b4d434df49bc3b14f')
age: "14"
class: "5A"
name: "ira"
```

Document flagged for deletion. CANCEL DELETE

Filter Type a query: { field: 'value' } or Generate query Explain Reset Find Options

ADD DATA EXPORT DATA 1 – 4 of 4

```
_id: ObjectId('6524008c4d434df49bc3b14d')
age: "20"
class: "aiec"
name: "saloni"
```

```
_id: ObjectId('652400d34d434df49bc3b14e')
age: "20"
class: "aia"
name: "aashi"
```

```
_id: ObjectId('652401534d434df49bc3b150')
age: "12"
class: "5A"
name: "ansh"
```

```
_id: ObjectId('652402594d434df49bc3b151')
```

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.

Exercise:

1) Implementing CRUD operations on a different NoSQL database like Couchbase.
2) Analyzing the performance of CRUD operations on different NoSQL databases.
3) Integrating NoSQL databases with web applications for CRUD operations.
4) Implementing data synchronization and replication in NoSQL databases.
5) Designing a schema for a specific use case in a NoSQL database.

Frequently Asked Questions:

1. What is the difference between NoSQL databases and traditional relational databases?
2. What are the advantages of using NoSQL databases over relational databases?
3. How does MongoDB handle consistency and scalability?
4. What is the CAP theorem, and how does it relate to NoSQL databases?
5. How can I install and set up MongoDB on my local machine?
6. What are the key differences between MongoDB, Cassandra, and NEO4j?
7. How does Cassandra ensure fault-tolerance and high availability?
8. Can I perform complex queries in NoSQL databases like MongoDB and NEO4j?
9. How does NEO4j handle relationships between data entities?
10. What are the typical use cases for each of the NoSQL databases discussed in this experiment?