EXPERIMENT.NO- 16

FAMILIARIZATION OF NOSQL DATABASES AND CRUD OPERATIONS

AIM:

Familiarization of NoSQL databases and CRUD operations.

THEORY:

NoSQL databases ("not only SQL") are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

NoSQL database features

Each NoSQL database has its own unique features. At a high level, many NoSQL databases have the following features:

1. Flexible schemas
2. Horizontal scaling
3. Fast queries due to the data model
4. Ease of use for developers

Types of NoSQL databases

The four major types of NoSQL databases: document databases, key-value databases, wide-column stores, and graph databases.

- Document databases store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.
- Key-value databases are a simpler type of database where each item contains keys and values.

- Wide-column stores store data in tables, rows, and dynamic columns.
- Graph databases store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

When should NoSQL be used?

When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:

- Fast-paced Agile development
- Storage of structured and semi-structured data
- Huge volumes of data
- Requirements for scale-out architecture
- Modern application paradigms like microservices and real-time streaming

MongoDB is the world's most popular NoSQL database.

Introduction to MongoDB

MongoDB is a document database designed for ease of development and scaling. The Manual introduces key concepts in MongoDB, presents the query language, and provides operational and administrative considerations and procedures as well as a comprehensive reference section.

MongoDB offers both local and cloud-hosted deployment options:

- For locally hosted deployments, MongoDB offers both a *Community* and an *Enterprise* version of the database:
    - o MongoDB Community is the source available and free to use edition of MongoDB. o MongoDB Enterprise is available as part of the MongoDB Enterprise Advanced subscription and includes comprehensive support for your MongoDB deployment. MongoDB Enterprise also adds enterprise-focused features such as LDAP and Kerberos

support, on-disk encryption, and auditing.

- MongoDB Atlas is a hosted MongoDB Enterprise service option in the cloud which requires no installation overhead and offers a free tier to get started.

Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

Collections/Views/On-Demand Materialized Views

MongoDB stores documents in collections. Collections are analogous to tables in relational databases. In addition to collections, MongoDB supports:

- Read-only Views (Starting in MongoDB 3.4)
- On-Demand Materialized Views (Starting in MongoDB 4.2).

Key Features

1. High Performance

MongoDB provides high performance data persistence. In particular,

● Support for embedded data models reduces I/O activity on database system. ● Indexes support faster queries and can include keys from embedded documents and arrays. 2**. Rich Query Language**

MongoDB supports a rich query language to support read and write operations (CRUD) as well as: ● Data Aggregation

● Text Search and Geospatial Queries.

3. High Availability

MongoDB's replication facility, called replica set, provides:

● *automatic* failover

● data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

4. Horizontal Scalability

MongoDB provides horizontal scalability as part of its *core* functionality:

● Sharding distributes data across a cluster of machines.

● Starting in 3.4, MongoDB supports creating zones of data based on the shard key. In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the Zones manual page for more information.

5. Support for Multiple Storage Engines

MongoDB supports multiple storage engines:

● WiredTiger Storage Engine (including support for Encryption at Rest)

● In-Memory Storage Engine.

In addition, MongoDB provides a pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

MongoDB CRUD Operations

CRUD operations *create*, *read*, *update*, and *delete* documents.

Create Operations

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection: • db.collection.insertOne() *New in version 3.2*

• db.collection.insertMany() *New in version 3.2*

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

Read Operations

Read operations retrieve documents from a collection; i.e. query a collection for documents.

MongoDB provides the following methods to read documents from a collection: • db.collection.find()

You can specify query filters or criteria that identify the documents to return.

Update Operations

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

- db.collection.updateOne() *New in version 3.2*
- db.collection.updateMany() *New in version 3.2*
- db.collection.replaceOne() *New in version 3.2*

In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

Delete Operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- db.collection.deleteOne() *New in version 3.2*
- db.collection.deleteMany() *New in version 3.2*

In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document. You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

RESULT:

Familiarized NoSQL databases and CRUD operations and CO5 is attained.