ASSIGNMENT NO. 09
TITLE: STUDY OF OPEN SOURCE NOSQL DATABASE: MONGODB

PROBLEM STATEMENT: To Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution)

OBJECTIVES:
- To study NoSQL databases and types
- To perform installation and configuration of MongoDB environment
- To perform MongoDB CRUD operations

OUTCOMES:
Students will be able to
- Install and configure MongoDB environment.
- Execute MongoDB CRUD operations

SOFTWARE & HARDWARE REQUIREMENTS:
1. 64-bit Open source Linux or its derivative's
2. MongoDB Server

THEORY:
What is NoSql?
A NoSQL database environment is, simply put, a non-relational and largely distributed database system that enables rapid, ad-hoc organization and analysis of extremely high-volume, disparate data types. NoSQL databases are sometimes referred to as cloud databases, non-relational databases, Big Data databases. NoSQL databases have become the first alternative to relational databases, with scalability, availability, and fault tolerance being key deciding factors.

Benefits of NoSQL
When compared to relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several issues that the relational model is not designed to address:
- Large volumes of structured, semi-structured, and unstructured data
- Object-oriented programming that is easy to use and flexible
- Efficient, scale-out architecture instead of expensive, monolithic architecture

Types of NoSQL Databases
There are four general types of NoSQL databases, each with their own specific attributes:

- Key-Value databases
- Column store
- Document database
- Graph database

Key-Value databases

These databases are designed for storing data in a schema-less way. In a key-value store, all of the data within consists of an indexed key and a value. The client can either get the value for the key, put a value for a key, or delete a key from the data store.

Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled.

Examples of this type of database include: Cassandra, DyanmoDB, Azure Table Storage (ATS), Riak, BerkeleyDB.

Document Databases

Expands on the basic idea of key-value stores where "documents" are more complex, in that they contain data and each document is assigned a unique key, which is used to retrieve the document. Documents are the main concept in document databases.

These documents are self-describing, hierarchical tree data structures which can consist of maps, collections, and scalar values. Some of the popular document databases we have seen are MongoDB, CouchDB , Terrastore, OrientDB, RavenDB.

Column family stores

Instead of storing data in rows, these databases are designed for storing data tables as sections of columns of data, rather than as rows of data. Column-family databases store data in column families as rows that have many columns associated with a row key. Various rows do not have to have the same columns, and columns can be added to any row at any time without having to add it to other rows.

Examples include: HBase, Cassandra ,BigTable and HyperTable.

Graph Databases

These databases are based on graph theory. Designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them. Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties. In graph databases, traversing the joins or relationships is very fast. The relationship between nodes is not calculated at query time but is actually persisted as a relationship.

Typical use cases in areas like Semantic Web and RDF, LinkedData, GIS, Genome analysis, modeling of social network data, deep recommendation algorithms and many more.

Examples: Neo4j - Open Source, Java, Property Graph model, AllegroGraph, Closed Source Sones - Closed Source, .NET focused Virtuoso - Closed Source, HyergraphDB - Open Source, Java, HyperGraph model

Introduction to MongoDB

MongoDB is an open-source document database, and leading NoSQL database. MongoDB is written in c++. MongoDB is a cross-platform, document oriented database that provides, high

performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Below given table shows the relationship of RDBMS terminology with MongoDB

RDBMS	**MongoDB**
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

Below given example shows the document structure of a blog site which is simply a comma separated key value pair.

```
{
_id: ObjectId(7df78ad8902c)
title: 'MongoDB Overview',
description: 'MongoDB is no sql database',
by: 'spm',
url: 'http://www.isquareit.edu.in',
tags: ['mongodb', 'database', 'NoSQL'],
likes: 100,
comments: [
{
user:'user1',
message: 'My first comment',
dateCreated: new Date(2011,1,20,2,15),
like: 0
},
{
user:'user2',
message: 'My second comments',
dateCreated: new Date(2011,1,25,7,45),
like: 5
```

```
}
]
}
```

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide _id while inserting the document. If you didn't provide then MongoDB provide a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of mongodb server and remaining 3 bytes are simple incremental value.

Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB there is no concept of relationship.

MongoDB Installation

1. Download the binary files for the desired release of MongoDB.

Download the binaries from the MongoDB Download Center

2. Extract the files from the downloaded archive.

```
tar -zxvf mongodb-linux-x86_64-3.4.7.tgz
```

3. Copy the extracted archive to the target directory.

Copy the extracted folder to the location from which MongoDB will run.

```
mkdir -p mongodb
cp -R -n mongodb-linux-x86_64-3.4.7/ mongodb
```

4. Create the data directory.

Before you start MongoDB for the first time, create the directory to which the mongod process will write data. By default, the mongod process uses the /data/db directory. If you create a directory other than this one, you must specify that directory in the dbpath option when starting the mongod process later in this procedure.

5. Run mongodb server

Now go to Bin directory and run mongod with setting the path as

```
./mongod --dbpath data\db
```

6. Open another terminal goto Bin folder and run

```
$ ./mongo
```

Advantages of MongoDB over RDBMS

- Schema less : MongoDB is document database in which one collection holds different different documents. Number of fields, content and size of the document can be differ from one document to another.
- Structure of a single object is clear
- No complex joins
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL
- Ease of scale-out: MongoDB is easy to scale

- Conversion / mapping of application objects to database objects not needed
- Uses internal memory for storing the (windowed) working set, enabling faster access of data

MongoDB Basic Commands

MongoDB provides rich semantics for reading and manipulating data. CRUD stands for **create, read, update, and delete**. These terms are the foundation for all interactions with the database.

MongoDB Create Operations

MongoDB use DATABASE_NAME is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

```
use DATABASE_NAME
```

To check your currently selected database use the command db

```
>db
```

If you want to check your databases list, then use the command show dbs.

```
>show dbs
```

The createCollection() Method : MongoDB db.createCollection(name, options) is used to create collection.

```
>db.createCollection(name, options)
```

In the command, name is name of collection to be created. Options is a document and used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

The insert() Method : To insert data into MongoDB collection, you need to use MongoDB's insert() or save()method.

```
>db.COLLECTION_NAME.insert(document) OR
>db.COLLECTION_NAME.insert(document name)
```

MongoDB Read Operations

The find() &findOne() Method

To query data from MongoDB collection, you need to use MongoDB's find() method. Find and findOne can be used to query a collection with miner difference. The find() method will display all the documents in a non-structured way & findOne() method will display all the documents in a structured way.

```
>db.COLLECTION_NAME.find()
>db.COLLECTION_NAME.findOne()
```

The pretty() Method : To display the results in a formatted way, you can use pretty() method.
>db.mycol.find().pretty()

MongoDB Update Operations

MongoDB update() and save() Method

MongoDB's update() and save() methods are used to update document into a collection. The update() method update values in the existing document while the save() method replaces the existing document with the document passed in save() method.

The Update() method: The update() method updates values in the existing document.

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

The Save() Method : The save() method replaces the existing document with the new document passed in save() method.

Syntax: >db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})

MongoDB Delete Operations

MongoDB Drop Database : MongoDB **db.dropDatabase ()** command is used to drop a existing database.

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

The drop() Method : MongoDB's db.collection.drop() is used to drop a collection from the database.

```
>db.COLLECTION_NAME.drop()
```

The remove () Method

MongoDB'sremove() method is used to remove document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag

1. deletion criteria : (Optional) deletion criteria according to documents will be removed.
2. justOne : (Optional) if set to true or 1, then remove only one document.

Syntax:

>db.COLLECTION_NAME.remove (DELLETION_CRITTERIA)

CONCLUSION: