Assignment No B1

Title: MongoDB Queries: DesignandDevelopMongoDBQueriesusingCRUDoperations.(UseCRUDoperations,SAVE method, logical operators etc.).

Objective: Understand CRUD operations in MongoDB and execution of basic CRUD operatios.

Theory:
What is NoSQL?

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. The purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like

Twitter, Facebook, Google collect terabytes of user data every single day.

NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would be "NoREL", NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

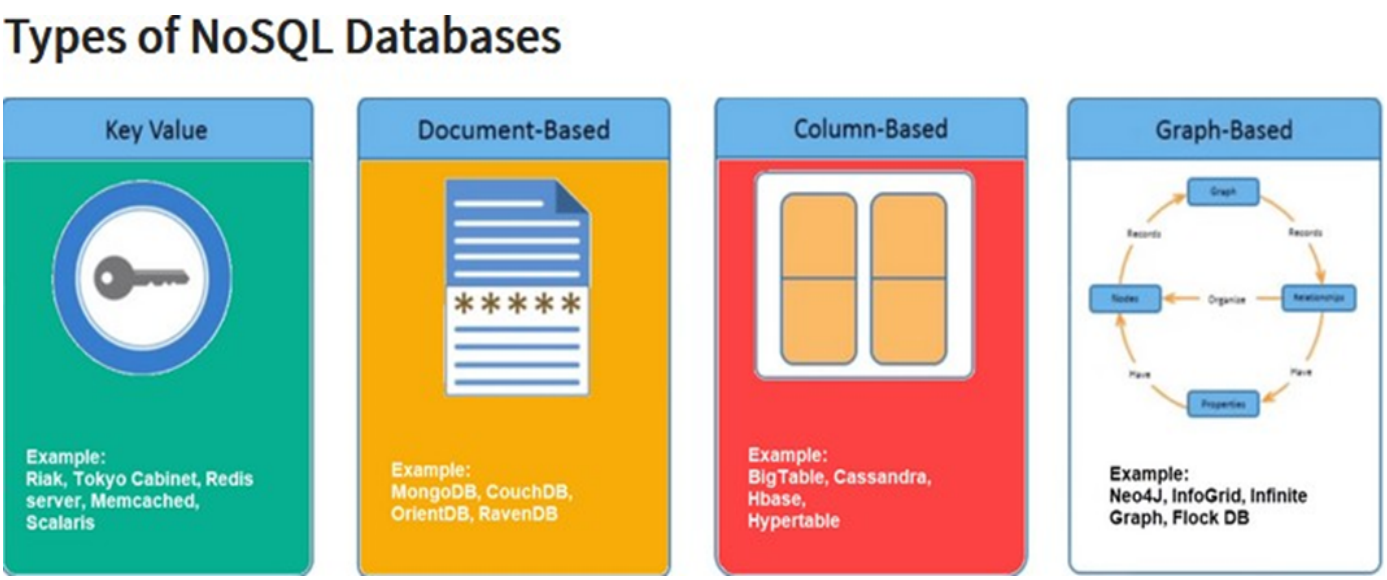
Why NoSQL?

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.

Difference Between SQL and NoSQL

SQL	NOSQL
Relational Database management system	Distributed Database management system
Vertically Scalable	Horizontally Scalable
Fixed or predifined Schema	Dynamic Schema
Not suitable for hierarchical data storage	Best suitable for hierarchical data storage
Can be used for complex queries	Not good for complex queries



Document-Oriented:
Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

MongoDB

Scalable High-Performance Open-source, Document-orientated database.

- Built for Speed
- Rich Document based queries for Easy readability.
- Full Index Support for High Performance.
- Replication and Failover for High Availability.
- Auto Sharding for Easy Scalability.
- Map / Reduce for Aggregation**.**

Advantages of MongoDB

- Schema less : Number of fields, content and size of the document can be differ from one document to another.
- No complex joins
- Data is stored as JSON style
- Index on any attribute
- Replication and High availability

Mongo DB Terminologies for RDBMS concepts

RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (JSON, BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

Data Types of MongoDB

- String : This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.
- Integer : This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- Boolean : This type is used to store a boolean (true/ false) value.
- Double : This type is used to store floating point values.
- Min/ Max keys : This type is used to compare a value against the lowest and highest BSON elements.
- Arrays : This type is used to store arrays or list or multiple values into one key.
- Timestamp : ctimestamp. This can be handy for recording when a document has been modified or added.
- Object : This datatype is used for embedded documents.
- Null : This type is used to store a Null value.

- Symbol : This datatype is used identically to a string however, it's generally reserved for languages that use a specific symbol type.
- Date : This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- Object ID : This datatype is used to store the document's ID.
- Binary data : This datatype is used to store binay data.
- Code : This datatype is used to store javascript code into document.
- Regular expression : This datatype is used to store regular expression

Basic Database Operations

- use <*database name*>

switched to database provided with command

- db

To check currently selected database use the command db

- show dbs

Displays the list of databases

- db.dropDatabase()

To Drop the database

- db.createCollection (name)

Ex:- db.createCollection(Stud)

- To create collection
- >show collection

List out all names of collection in current database

- db.*databasename*.insert

({Key : Value})

Ex:- db.Stud.insert({{Name:"Jiya"})

In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

- db.collection.drop()

Example:- db.Stud.drop()

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

CRUD Operations:

- Insert
- Find
- Update
- Delete

CRUD Operations – Insert

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

Syntax

>db.COLLECTION_NAME.insert(document)

Example

>db.stud.insert({name: “Jiya”, age:15})

_id Field

- If the document does not specify an *_id* field, then MongoDB will add the _id field and assign a unique *ObjectId* for the document before inserting.
- The _id value must be unique within the collection to avoid duplicate key error.

Insert a Document without Specifying an _id Field

- db.stud.insert({ Name : “Reena", Rno: 15 })
- db.stud.find()

{ "_id" : "5063114bd386d8fadbd6b004”, “Name” : “Reena", “Rno”: 15 }

Insert a Document Specifying an _id Field

- db.stud.insert({ _id: 10, Name : “Reena", Rno: 15 })
- db.stud.find()

{ "_id" : 10, “Name” : “Reena", “Rno”: 15 }

Insert Single Documents

db.stud.insert ({Name: “Ankit”, Rno:1, Address: “Pune”})

Insert Multiple Documents

```
db.stud.insert ( [
{ Name: “Ankit”, Rno:1, Address: “Pune”} ,
{ Name: “Sagar”, Rno:2},
{ Name: “Neha”, Rno:3}
] )
```

Insert Multicolumn attribute

```
db.stud.insert( {
   Name: “Ritu",
   Address: { City: “Pune", State: “MH” },
Rno: 6
})
```

Insert Multivalued attribute

```
db.stud.insert( {
    Name : “Sneha",
    Hobbies: [“Singing”, “Dancing” , “Cricket”] ,
Rno:8
})
```

Insert Multivalued with Multicolumn attribute

```
db.stud.insert( {
    Name : “Sneha",
  Awards: [ { Award : “Dancing”, Rank: “1st”, Year: 2008 },
            {Award : “Drawing”, Rank: “3rd”, Year: 2010 } ,
            {Award : “Singing”, Rank: “1st”, Year: 2015 } ],
Rno: 9 })
```

CRUD Operations – **Find**

The find() Method- To display data from MongoDB collection. Displays all the documents in a non structured way.

Syntax

>db.COLLECTION_NAME.find()

The pretty() Method- To display the results in a formatted way, you can use **pretty()** method.

Syntax

>db. COLLECTION_NAME.find().pretty()

Specify Equality Condition

use the query document { <field>: <value> }

Examples:

- db.stud.find(name: “Jiya" })
- db.stud.find({ _id: 5 })

Comparison Operators	
Operator	**Description**
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	values that are greater than or equal to a specified value.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$in	Matches any of the values specified in an array.
$nin	Matches none of the values specified in an array.

Find Examples with comparison operators

- db.stud.find({ rno: { $gt:5} }) *Shows all documents whose rno>5*
- db.stud.find({ rno: { $gt: 0, $lt: 5} }) *Shows all documents whose rno greater than 0 and less than 5*

Examples to show only particular columns

- db.stud.find({name: "Jiya"},{Rno:1}) *To show the rollno of student whose name is equal to Jiya (by default _id is also shown)*
- db.stud.find({name: "jiya"},{_id:0,Rno:1}) *show the rollno of student whose name is equal to Jiya (_id is not shown)*

Examples for Sort function

- db.stud.find().sort({ Rno: 1 })

  Sort on age field in Ascending order (1)
- db.stud.find().sort({ Rno: -1 })

  Sort on age field in Ascending order(-1)

Examples of Count functions

- *db.stud.find().count()*

  Returns no of documents in the collection

Examples of limit and skip

- *db.stud.find().limit(2)*

  Returns only first 2 documents
- *db.stud.find().skip(5)*

  Returns all documents except first 5 documents

CRUD Operations – Update

Syntax

db.*CollectionName*.update (
<query/Condition>,
<update with $set or $unset>,
{
upsert: <boolean>,
multi: <boolean>,

})

upsert

• If set to *True*, creates new document if no matches found.

multi

• If set to *True*, updates multiple documents that matches the query criteria

CRUD Operations – Update Examples

1> Set age = 25 where id is 100, First Whole document is replaced where condition is matched and only one field is remained as age:25

db.stud.update(

{ _id: 100 },
{ age: 25})

2> Set age = 25 where id is 100, Only the age field of one document is updated where condition is matched .

db.stud.update(

{ _id: 100 },
{ $set:{age: 25}})
3> To remove a age column from single document where id=100

db.stud.update(

{ _id: 100 },
{ $unset:{age: 1}})

CRUD Operations – Remove

- Remove All Documents

  db.inventory.remove({})

- Remove All Documents that Match a Condition

  db.inventory.remove({ type : "food" })

- Remove a Single Document that Matches a Condition

  db.inventory.remove({ type : "food" }, 1)