Class: TE IT(A)

Roll No: 35040

Batch: B

Group: C

Assignment No. 1

Title: Insert, update and remove in MongoDB

Problem Statement: Create a database with suitable example using MongoDB and implement

1.Inserting and saving document (batch insert, insert validation)

2. Removing document

3. Updating document (document replacement, using modifiers, upserts, updating multiple documents, returning

Requirements: MongoDB

Prerequisites: Basic knowledge about MongoDB**.**

Theory:

1. **What is NOSQL DB?**
   NoSQL databases (aka "not only SQL") are non tabular, 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.

2. **Difference between SQL and NOSQL.**

SQL	NoSQL
RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have have dynamic schema
These databases are not suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
These databases are best suited for complex queries	These databases are not so good for complex queries
Verticlly Scalable	Horizontally scalable

3. **Types of NOSQL**
    - Key Value Pair Storage Database
    - Column Based Database
    - Graph Based Database
    - Document Oriented Database

4. **Properties of MongoDB**

    MongoDB maintains the most valuable features of relational databases:
    - strong consistency
    - ACID transactions
    - expressive query language
    - secondary indexes.

    As a result, developers can build highly functional applications faster than NoSQL databases.

5. **Syntax of insert, update and remove in detail**

    INSERT:
    - **db.collection.insert({document});**
    - example- db.driver_info.insert({did:1,name:"Akash",address:"Pune"});

    UPDATE:

    - db.collection.update({query},{update},{options});
    - db.collection.update({query},{update},{upsert:true});
    - example- db.driver_info.update({did:2},{$set:{name:"Ravi"}});

    REMOVE:

    - db.collection.remove({query},{justOne});
    - example- db.driver_info.remove({did:8});

Conclusion: Thus, in this assignment how to insert,update and remove in MongoDB.

Roll No : 35040

TE IT(A)

Group : C

Batch : B

Assignment 2

Title: - Queries in MongoDB

Problem Statement: -- Execute at least 10 queries on any suitable MongoDB database that demonstrates following querying techniques:

- find and findOne (specific values)
- Query criteria (Query conditionals, OR queries, $not, Conditional semantics)
- Type-specific queries (Null, Regular expression, Querying arrays)

Requirements: -- MongoDB

Prerequisites: -- Basic Of MongoDB

Theory: --

find and findOne :

find() –
Nomatter number of documents matched, the find() method does not return null, it returns a cursor.
Eg. To select all the documents whose sid is 3
Query : db.staff.find({sid:3}).pretty();

findOne() –
The findOne() returns first document if query matches otherwise returns null.
Eg. To select first document whose sid is 3.

Query : db.staff.findOne({sid:3}).pretty();

Comparison Query Operators :

In MongoDB the conditional operators are :

(>) greater than - $gt

(<) less than - $lt

(>=) greater than equal to - $gte

(<=) less than equal to - $lte

(!=) not equal to - $ne

Syntax : db.collection_name.find({col_name:{comparison_operator:value}}).pretty();

Logical Query Operators :

Name	Description
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

Regular Expression :

Provides regular expression capabilities for pattern matching strings in queries. MongoDB uses Perl compatible regular expressions (i.e. “PCRE”) version 8.42 with UTF-8 support.

To use $regex, use one of the following syntaxes:

{ <field>: { $regex: /pattern/, $options: '<options>' } }

{ <field>: { $regex: 'pattern', $options: '<options>' } }

{ <field>: { $regex: /pattern/<options> } }

1. String Match
   Eg. Find The Customers Whose Name Is “Qwe”

>db.customer_Info.find({Name:{$Regex;"QWE"}}).pretty()
eg. Find The Customers Who Have Orders In The Year 2016
>db.customer_Info.Find({Orddate:/^2016/}).Pretty()

2. Case Insensitive
   Eg. Find The Customers Whose Name Is "Tgb"
   >db.customer_Info.find({Name:{$Regex:"Tgb",$Options:"$I"}}).Pretty()
   >db.customer_Info.update({Cid:2},{$Set:{Email:Null}},{Multi:True})

3. Null
   Eg. Find The Customers Who Do Not Have An Email Id.
   >db.customer_Info.find({Email:Null}).Pretty()

4. Substring
   Eg. Find The Customers Whose Last Name Contains The Substring
   >db.customer_Info.Find({"Name.Lname":{$Regex:/MN?/i}}).Pretty()

Querying Arrays:

1) Size
   Eg. Display the records of customers who have more than two phone numbers
   > db.customer_Info.find({Phno:{"$Size":2}}).Pretty()
2) Push
   Eg. Add a new phone number to a customers record whose cid is 2.
   >db.customer_Info.update({Cid:2},{"$Push":{"Phno":8866543211}})

Conclusion : Thus, we have implemented queries in MongoDB.