MongoDb

Introduction

It is a non sql database. Stored Data in single document based . In document based format, Store data in key value format like json. Rather calling json it's a bson (binary javascript object notation).We have document to store data in key value pair. Document is a group of key value pair. Collection is a group of document. Database is a group of collection.

Document which store data can also be called object.

Steps for making connection with cli

1. Navigate to mongo folder bin/exe and run it
2. Type mongosh for opening connection

some command for mongodb

Point to be noted while creating database

Until you create collection inside databse ,database will no longer list in "show dbs" cmd

After creating database, you need to create collection inside it for listing in in "show dbs" cmd;

1. Databse cmd
   a. show dbs //show all available database
   b. use DATABASE_NAME // select database by name which you want to use
   c. use DATABASE_NAME // use to create and select Database
   d. Drop or deleting collection
      i. Use DATABASE_NAME // To select db which you want to drop
      ii. db.dropDatabase() // run this method to delete databse
2. For collection cmd
   a. show collections; // list all collection of databse
   b. Creating collection
      i. Use Database //select databse first where you want to create collection

ii. Run db.createCollection("collection_name"); //this method take parameter as COLLECTION_NAME to create collection

c. db.student.stats(); //describe collection
d. Creating collection with extra argument
    i. Db.createCollection(“teachers”,{capped:true,size:10000000,max:100,autoIndexId:false});
    ii. Example : db.createCollection("teachers",{capped:true,size:100000,max:100,autoIndexId:false})
    iii. //if capped is true it will take fixed field value
    iv. //size takes maximum amount size of file size
    v. Max used to set how many number of document you want to store
    vi. AutoIndexed used for faster searching of same value by indexing it
e. Inserting data into collection
    i. db.collection_name.insertOne({key:”value1”,key:value2,key:value3});//one data insertion
        1. example ({name:”suraj”,age:23})
            a. after insertion it will return acknowlegde as true and insertion id
    ii. db.insertMany([{},{},{}]) //insert many data obect{} inside array[] to store multiple data.
        1. example db.collection_name.insertMany([{name:"anil",age:20},{name:"manuz",age:32},{name:"subin",age:34,gender:"female"}])
f. To fetch data (db.findOne() and db.find())
    i. db.collection_name.find()
        1. return all inserted data with its insertion id or specific object id.Used to fetch many data at once
    ii. db.findOne(name:”suraj”) // return object from collection which name is suraj. Only specific for single unique data
    iii. db.collection_name.find({query},{projection})// multi parmeter value for fetching data with find() method
        1. {query} part is like where clause what data you need to fetch like{name:”suraj”,age:23}

2. {projection} part is like what key value you need to fetch like{name:true,_id:false}
    a. True set key value will only fetch
3. example: db.collection_name.find({name:"suraj",age:22},{name:true,_id:false});

g. updating data(updateOne() or update())
    i. note updating method can use also for adding new key value pair by $set method while for removing key value there is $unset:{key:value} method
    ii. updateOne({name:value},{$set:{key:vale}}) //for specific data only
        1. 1st pass parameter object will select specific data by given key value pair and second one is $set parmerter which take object as parameter to update in key:value pair
        2. Example: db.student.updateOne({name:"suraj"},{$set:{age:40}});
    iii. updateOne() with object id
        1. update by taking specific object id
            a. example : db.student.updateOne({_id: ObjectId("651009f43ab28f179d7e656d")},{$set:{name:"harry potter"}});
    iv. updateOne() with $unset:{}
        1. $unset method used to remove key value pair from column
        2. Example: db.student.updateOne({_id: ObjectId("651009f43ab28f179d7e656d")},{$unset:{name:"harry potter"}});
    v. updateMany({},{$set:{key:value}})// to update many field in document
        1. takes two parameter 1st parmeter is empty object{} which means you selecting all the field for updation,2nd parameter is {$set;{key:value}} which means what key value you want to update or add to document field
        2. example: db.student.updateMany({},{$set:{hobby:"playing"}});

vi. updateMany({key:{$exist:false}},{$set:{key:value}})//$exist will select document which have its key column false means do not select that key field and another parameter is {$set:{}}which used to update docunet field
    1. example: db.student.updateMany({dob:{$exists:false}},{$set:{hobby:"typing"}});
vii. updateMany({key:value},{$set:{key:value}})// updating may field with selecting available key field
    1. example: db.student.updateMany({name:"suraj"},{$set:{hobby:"programming"}});
viii. updating inner object (db.collection_name.updateOne({“key.key”:value},{$set:{key.key:value}}))
    1. example: db.student.updateOne({"address.temp":"koteshwor"},{$set:{"address.temp":"narephat"}})

h. deleting document (deleteOne() or deleteMany())
    i. deleteOne({key:value}) //take parameter in key value field to delete specific field
        1. example: db.student.deleteOne({name:"surajssss"});
    ii. deleteMany({key:value})// delete multiple document field which have selected by {key:value}parameter
        1. example: db.student.deleteMany({status:0});
    iii. db.student.deleteMany({status:{$exists:true}});
        1. delete all the field which have status key
    iv. deletion with inner key object(db.collection_name.deleteOne({“key.key”:value})
        1. example: db.student.deleteOne({ "status.age": 3 })
            a. find inner key and delete the field

3. Date function
    a. new Date() // this method take current date automatically while insertion
        i. example db.collection_name.insertOne({name:”suraj”,dob: new Date()})
4. null value

    a. null // if we do not have any value for variable but need to further use null value is required
        i. example example db.collection_name.insertOne({name:”suraj”,clgJoin: new Date(),graduationDate:null})
5. muti value variable in collection object
    a. key:[] // use array to store multiple value
        i. example db.collection_name.insertOne({name:"Harry",age:20,subject:[" economics","distributed","science"]})
6. nested document
    a. key:{} // use object to store nested data
        i. example db.collection_name.insertOne({name:"Harry",age:20,address:{t emp:"koteshwor",permanent:"Bardibas"},dob:new Date()});
7. Aggregate method
    a. sort({name:1})// take parameter 1 for ascending and -1 for descending
        i. Example db.collection_name.find().sort({name:1 });// take parameter as -1 or 1 to sort name by key
    b. limit(2) take parameter how much data you need
        i. db.collection_name.find().limit(2);//
    c. sort() with limit()
        i. db.collection_name.find().sort({age:1}).limit(2); // returns 2 data entries by sorting
8. Operator

   Since operatore are denoted by $ symbol
    a. Comparison operator
        i. $ne (key:{$ne:value}}) // ne means not equal return all the name which do not match with your given key value
            1. Example : db.student.find({name:{$ne:"suraj"}});
        ii. $lt (key:{$lt:value) // lt means less than return all the field which is less than your given value
            1. Example : db.student.find({age:{$lt:22}});
        iii. $lte (key:{$lte:value) // lte means less than or equal to, return all the field which is less or equal to than your given value
            1. Example : db.student.find({age:{$lte:22}});

iv. $gt (key:{$gt:value) // gt means greater than ,return all the field which is greater than your given value
    1. Example : db.student.find({age:{$gt:22}});
v. $gte (key:{$gte:value) // gte means greater than or equal to, return all the field which is gretaer or equal to than your given value
    1. Example : db.student.find({age:{$gte:22}});
vi. Between operator(key:{$gte:vale,$lte:value}) // find grater and less than number according to your given value
    1. Example : db.student.find({age:{$gte:20,$lte:20}});
vii. $in (key:{$in:[“value1”,”value2”]}) //in means include,return all field which match with given key includes array value
    1. Example: db.student.find({name:{$in:["suraj","anil"]}});
viii. $nin (key:{$nin:[“value1”,”value2”]}) //nin means not inclue,return all field which match with given key and do not includes array value
    1. Example: db.student.find({name:{$nin:["suraj","anil"]}});

b. Logical opertors

Logical operators return data based on expressions that evaluate true or false

i. $and ($and:[{key:“value1”},{key:”value2”}]) // return true if both condition match then reurn matched data
    1. Example : db.student.find({$and:[{name:"suraj"},{age:40}]});
ii. $and with condtition(db.student.find({$and:[{key:"value"},{key:{$lte:value}}]});) // return all the field which match with given condition
    1. Example : db.student.find({$and:[{name:"suraj"},{age:{$lte:45}}]});
iii. $or ($or:[{key:"value"]},{key:{$lte:value}]})// return data If any condition is true
    1. db.student.find({$or:[{name:"suraj"},{age:{$lte:45}}]});
iv. $nor ($nor:[{key:"value"]},{key:{$lte:value}]})// return data where both condition are false

1. Example :
   db.student.find({$nor:[{hobby:"programming"},{age:{$l
   te:40}}]});

v. $not ({key:{$not:{$lte:value}}}); retrun all the field except given value also returns null assign value
   1. Example : db.student.find({age:{$not:{$lte:30}}});

9. Indexes
   a. db.student.find({name:"suraj"}).explain("executionStats");
      i. examine each document one by one
      ii. perform linear serach
      iii. fast searching
   b. creating or getting index
      i. db.student.createIndex({name:1});
      ii. db.student.getIndexes();
         1. give all the indexes of database
      iii. db.student.dropIndexes("index_name");
         1. db.student.dropIndexes("name_1");
10. join in mongo db

```
db.orders.aggregate([
  {
   $lookup:
   {
    from: "products",
    localField: "product_id",
    foreignField: "_id",
    as: "orderDetails"
   }
  }
])
```

11. multiple join

```
db.users.aggregate([
  {
    $match: {
      _id: ObjectId("user_id") // Replace with the actual user _id
    }
  },
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "user_id",
      as: "user_orders"
    }
  },
  {
    $lookup: {
      from: "products",
      localField: "user_orders.product_id",
      foreignField: "_id",
      as: "user_order_products"
    }
  }
]);
```