Create and Drop database in MongoDB

MongoDB uses indexing in order to make the query processing more efficient. If there is no indexing, then the MongoDB must scan every document in the collection and retrieve only those documents that match the query. Indexes are special data structures that stores some information related to the documents such that it becomes easy for MongoDB to find the right data file. The indexes are order by the value of the field specified in the index.

Creating an Index:

MongoDB provides a method called createIndex() that allows user to create an index.

Syntax:-C:\Users\cbkpc>mongosh

Current Mongosh Log ID: 66cf2d23c3d0b586f22710bb

Connectingto:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.0

Using MongoDB: 7.0.14

Using Mongosh: 2.3.0

Syntax:-test> show dbs

It will display all the running databases in the system

admin 40.00 KiB

college 16.00 KiB

config 96.00 KiB

local 72.00 KiB

Syntax:-test> use college;

switched to db college

CRUD Operation on document

Drop() method is used to delete the database from the system.

Create and Drop Collections

It will be created implicitly. We can also create collection explicity using "createCollection()" command. We will have to follow the syntax as discussed below.

After learning to MongoDB create collection, let's see how we drop collection in MongoDB. MongoDB Drop Collection is even more easy than creating it. To drop a collection, we need to execute the following command.

Syntax:-college> db.student.insertOne({name: "Ajju", branch: "CSE", qualification: "Diploma"});

```
{
  acknowledged: true,
  insertedId: ObjectId('66cf2e8fc3d0b586f22710bc')
}
```

Syntax:-college> db.student.insertMany([{name: "Dinu", branch: "CSE", qualification: "Diploma"},{name: "kishan", branch: "CSE", qualification: "Diploma"}]);

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66cf2f5cc3d0b586f22710bd'),
    '1': ObjectId('66cf2f5cc3d0b586f22710be')
  }
}
```

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:

Syntax:-college> db.student.find();

```
[
```

```
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'CSE',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710be'),
    name: 'kishan',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Update Operations

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

db.collection.updateOne()

db.collection.updateMany()

db.collection.replaceOne()

Syntax:-
college>db.student.updateOne({name:"Ajju"},{$set:{branch:"EC"}});

{
 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0
}

MongoDB-sort() Method

The sort() method specifies the order in which the query returns the matching documents from the given collection. You must apply this method to the cursor before retrieving any documents from the database. It takes a document as a parameter that contains a field value pair that defines the sort order of the result set. The value is 1 or -1 specify an ascending or descending sort respectively.

Syntax:-college> db.student.find();

[
 {
  _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
  name: 'Ajju',
  branch: 'EC',
  qualification: 'Diploma'
 },
 {
  _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
  name: 'Dinu',
  branch: 'CSE',
  qualification: 'Diploma'
 },
 {
  _id: ObjectId('66cf2f5cc3d0b586f22710be'),

```
    name: 'kishan',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Delete Operations

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

db.collection.deleteOne()

db.collection.deleteMany()

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.

Syntax:-college> db.student.deleteOne({name:"kishan"});

{ acknowledged: true, deletedCount: 1 }

}

qualification: 'M.Tech'

MongoDB Cursor

In MongoDB, when the find() method is used to find the documents present in the given collection, then this method returned a pointer which will points to the documents of the collection, now this pointer is known as cursor. Or in other words we can say that a cursor is a pointer, and using this pointer we can access the document. By default, cursor iterate automatically, but you can iterate a cursor manually.

Manually iterating a cursor

In MongoDB, the find() method return the cursor, now to access the document we need to iterate the cursor. In the mongo shell, if the cursor is not assigned to a var keyword then the mongo shell automatically iterates the cursor up to 20 documents. MongoDB also allows you to iterate cursor manually. So, to iterate a

cursor manually simply assign the cursor return by the find() method to the var keyword Or JavaScript variable.

Note: If a cursor inactive for 10 min then MongoDB server will automatically close that cursor.

Syntax:-college> db.student.find();

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Syntax:-college> db.student.find().limit(1);

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
    qualification: 'Diploma'
  }
]
```

Syntax:-college> db.student.find().limit(2);

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Syntax:-college> db.student.find().sort({name:-1});

```
[
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
```

```
    qualification: 'Diploma'
  }
]
```

Parameter:

The parameter contains a field: value pair that defines the sort order of the result set. The value is 1 or 1 that specifies an ascending or descending sort respectively. The type of parameter is a document.

Syntax:-college> db.student.find().sort({name:1});

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Syntax:-college> varmycursor=db.student.find();

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
```

```
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Count cursor:

In order to get the correct documents, we need to know how many documents. are present for that collection. To get that we can use the count() method which returns the total number of documents are present in the given collection.

Syntax:-college> db.student.find().count();

2

Cursor.toArray():

In order to have as an array that contains all documents returned by the cursor. we can use toArray() method.

Syntax:-college> db.student.find().toArray();

```
[
  {
    _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
    name: 'Ajju',
    branch: 'EC',
    qualification: 'Diploma'
  },
  {
    _id: ObjectId('66cf2f5cc3d0b586f22710bd'),
```

```
    name: 'Dinu',
    branch: 'CSE',
    qualification: 'Diploma'
  }
]
```

Syntax:-college> db.student.find().next();

```
{
  _id: ObjectId('66cf2e8fc3d0b586f22710bc'),
  name: 'Ajju',
  branch: 'EC',
  qualification: 'Diploma'
}
```

It will retrieve all the description of the created within collection.

Create user and add role in MongoDB

In MongoDB, we are allowed to create new users for the database. Every MongoDB user only accesses the data that is required for their role. A role in MongoDB grants privileges to perform some set of operations on a given resource. In MongoDB, users are created using createUser() method. This method creates a new user for the database, if the specified user is already present in the database then this method will return an error.

How to create a normal user without any roles?

In MongoDB, we can create a user without any roles by specifying an empty array[] in the role field in createl Iser() method.

Syntax:-college> db.student.createIndex({Key:1});

Key_1

Indexing in MongoDB:

MongoDB uses indexing in order to make the query processing more efficient. If there is no indexing, then the MongoDB must scan every document in the

collection and retrieve only those documents that match the query. Indexes are special data structures that stores some information related to the documents such that it becomes easy for MongoDB to find the right data file. The indexes are order by the value of the field specified in the index.

Creating an Index:

MongoDB provides a method called createIndex() that allows user to create an index.

clientSource: If the value of this field is present, so when a user is authenticating the server verifies the client IP by checking the IP address in the given list or CIDR range in the list. If the client IP present in the list then the server authenticate the client or if not then server will not authenticate the user.

serverAddress: It is a list of IP addresses or CIDR ranges to which the client can connect. If the value of this field is present in the list, then the server verify the client connection and if the connection was established via unrecognized ip

address, then the server does not authenticate the user. Let us discuss this concept with the help of an example:

Example:

In this example, we are going to create a user with authentication restrictions:

Syntax:-college> db.createUser(

... {

... user:"kishan",

... pwd:"kishan@123",

... roles:[]

... }

...);

{ ok: 1 }

Syntax:-college> db.getUsers();

{

users: [

```
    {
      _id: 'college.kishan',
      userId: UUID('04994dcc-8ac0-431c-879e-3bfa77a9b266'),
      user: 'kishan',
      db: 'college',
      roles: [],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
```

How to create a user with authentication restrictions?

In MongoDB. authentication is a process which checks whether the user/client who is trying to access the database is known or unknown. If the user is known then it allows them to connect with server. We can also create a user with authentication vestrictions using createUsert) method by setting the value of authentication Restrictions field. This field provides authentication permission of the user and contains the following fields:

Syntax:-
college>db.grantRolesToUser("kishan",[{role:"readWrite",db:"college"}])
;

{ ok: 1 }

Syntax:-college> db.getUsers();

```
{
  users: [
    {
      _id: 'college.kishan',
      userId: UUID('04994dcc-8ac0-431c-879e-3bfa77a9b266'),
      user: 'kishan',
      db: 'college',
```

```
    roles: [ { role: 'readWrite', db: 'college' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
 ],
 ok: 1
}
```

How to drop a user?

In Mongodb, we can also drop a user using dropUser() method. This method returns true when the user is deleted otherwise return false.

How to create a user for a single database?

in MongoDB, we can also create a user for single database using createUser() method. Let us discuss this concept with the help of an example:

```
Syntax:-college> db.createUser(
... {
... user:"restrict",
... pwd:passwordPrompt(),
... roles:[{role:"readWrite",db:"example"}],
... authenticationRestrictions:[{
... clientSource:["192.168.56.0"],
... serverAddress:["198.157.54.0"]
... }]
... }
... );
Enter password
**********{ ok: 1 }
```