MONGODB

Handwritten

Notes

Prepared By: TOPPERWORLD LEARN & GROW

Index :-

MongoDB

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

Collection:-

Collection is a group of MongoDB document. It is an eqivalent to RDBMS table. A collection exists within a single database. Collections do not enforce a schema document within a collection can have diffrent fields.

Document:-

A document is a set of key-value pair. Document have dynamic schema. Dynamic schema means that document in the same collection do not need to have the same set of fields and structure and common fields in a collection document may hold diffrent types of data.

Advantages of MongoDB over RDBMS :-

- Schema less :- MongoDB is a document database in which one collection holds diffrent documents Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is a clear
- No complex join
- Deep query-ability. MongoDB supports dynamic queries on document using a document-based query language that nearly as powerfull as SQL
- Tuning
- Conversion/

Why use MongoDB :

- Document Oriented Storaye.
- Index on any attribute
- Replication and availability
- Auto-Sharding
- Rich Queries
- Fast in-place updates
- Proffesional support by MongoDB

Where to use MongoDB :

① MongoDB → Big Data
Content Management and Delivery
Mobile and Social Infrastructure
User Data Management
Data hub

Start MongoDB

sudo service mongodb start

Stop MongoDB

Sudo service mongodb stop

Restart MongoDB

Sudo service mongodb restart

MongoDB – Data Modelling.

Data in MongoDB has flexible schema documents in the same collection. They do not need to have the same set of fields or structure common fields in the collections document may hold diffrent types of data

Data Model Design:-

MongoDB provides 2 types of data models.

① Embedded Data Model

In this model, you can have all the related data in a single document. It is also known as de-normalized data model.

Example:-

```
{
   _id:,
   Emp_ID: "10025AE336"
   Personal_details: {
         First_Name : "Radhika",
```

```
        Last_Name : "Sharma",
        Dato_of_Birth : "1995-09-26"
    },
    Contact : {
        email : "radhika_sharma.123@gmail.com",
        phone : "9848022338"
    },
    Address : {
        city : "Hydrabad",
        Area : "MadApur",
        State : "Telangana"
```

Normalized Data Model :-

In this model, you can refer the sub-documents in the original document, using referances.

Example :-

```
Employee :-
  {
     _id : <ObjectId01>,
     Emp_ID : "10025AE336"
  }

Personal_details :
    {
       _id : <ObjectId102>
       empDOCID : "ObjectId101",
       First_Name : "Radhika",
       Last_Name : "Sharma",
       Date_of_Birth = "1995-09-26"
    }
```

Contact:
{
_id : <ObjectId103>,
empDocID : "ObjectId101",
email : "radhika_sharma.123@gmail.com",
phone : "9848022338"
}

Address:-
{
_id:-<ObjectId104>,
empDocID : "ObjectId101",
City : "Hydrabad",
Area : "Madapur",
State : "Telangana"

Considerations while designing Schema in MongoDB

- Design your schema according to user requirement
- Combine objects into one document if you will use them together.
- Do joins while write, not on read.
- Optimize your schema for most frequent use cases
- Do Complex aggregation in schema.
- Duplicate the data because disk space is cheap as compared to compute time.

MongoDB - Create Database

The use Command

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

Syntax:-
Basic Syntax of use DATABASE statement as follows:-
use DATABASE_NAME

Example:-
If you want to use a database with name <mydb>, then use DATABASE statement
>use mydb
switched to db mydb

To check your currently selected database, use the command db
>db
mydb

If you want to check your database list, use the command show dbs
>Show dbs
local 0.78125GB
test 0.23012 GB

Your created database is not present in list
To display database you need to insert atleast

One document into it.

```
>db.movie.Insert({"name":"---."})
>Show dbs
  local:      0.78125 GB
  mydb        0.23012 GB
  test        0.23012 GB
```

MongoDB - Drop Database

The dropDatabase() method.

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax :-

Basic syntax of dropDatabase() command is as follow

```
db.dropDatabase()
```

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

Example :-

Check the list of available databases by using the command, show dbs

```
>Show dbs
local   0.78125GB
mydb    0.23012SGB
test    0.23012 GB
>
```

If you want to delete new database <mydb> then dropDatabase() command would be follows

```
>use mydb
switched to db mydb
>db.dropDatabase()
>{"dropped":"mydb","ok":1}
>
```

MongoDB - Create Collection

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

Syntax:-
Basic syntax of createCollection() command

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

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

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

Options parameter is optional, so you need to specify only name of collection.

Field	Type	Description
Capped	Boolean	If true, enables a caped collection. Capped collection is a fixed size collection that automatically overwrite its oldest entries. when it reaches to its maximum size. If you specify true, you need to specify size parameter also
autoIndexed	Boolean	If true, automatically create index on _id fields Default value is false
Size	number	Specifies a maximum size in bytes for a capped collection If capped is true then you need to specify this field also.
max	number	Specifies the maximum number of document allowed in the capped collection

While Inserting document, MongoDB first checks size field of capped collection, then it checks max field.

Example:-
Basic syntax of createCollection() method without option is as follows.

```
>use test
switched to db test
>db.createCollection("mycollection")
{ "Ok" : 1 }
>
```

You can check the created collection by using the command show collections.

```
> show collections
mycollections
system.indexes
```

The following example shows the syntax of create collection() method will few important options.-

```
>db.createCollection("mycol", { capped : true,
        autoIndexID: true, size: 6142800,
        max:10000}) {
            "Ok": 0,
    "errmsg": "BSON field 'create.autoIndexID'
        is an unknown field", "code" : 40415,
  "codeName":"Location 40415" }
  >
```

In MongoDB, you don't need to create collection, MongoDB create collection automatically.

```
>db.tutorialspoint.insert({"name":"tutorialspoint"})
WriteResult({"nInserted":1})
> Show collections
 mycol
 mycollection
 System.Indexes
 tutorialspoint
 >
```

MongoDB - Drop Collection.

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

Syntax :-
Basic syntax of drop command

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

Example.

```
> use mydb
 Switched to db mydb
 > Show collections
 mycol
 mycollection
 System.indexes
 tutorialspoint
 >
```

Now drop the collection with the name mycollection.

```
>db.mycollection.drop()
true
>
```

Again check the list of collection into database.

```
>show collections
mycol
System.Indexes.
tutorialspoint
>
```

drop() method will return true. If the selected collection is dropped successfully, otherwise it will return false.

MongoDB - Datatypes.

- **String**
  This is most commonly used datatypes to store the data. String 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 server.

- **Boolean**
  This type is used to store boolean 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 in 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 document

- Null :-
  This type is used to store Null value

- Symbol :-
  This datatype is used identically to a string its 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 time by creating object of Date and passing day, month, year onto it

- Object ID -
  This datatype is used to store document ID.

- **Binary data :-**
  This datatype is used to store binary data.

- **Code :-**
  This datatype is used to store javascript code into the document.

- **Regular Expression :-**
  This datatype is used to store regular expression

MongoDB - Insert Document

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.users.insert({
... _id : ObjectId("507f191e810c19e810c19729de860"),
... title : "MongoDB Overview",
... description : "MongoDB is noSQL Database",
... by : "tutorial point",
... url : "http://www.tutorialspoint.com",
... tags: ['mongodb', 'database', 'NoSQL'],
... likes : 100
... })
WriteResult({"nInserted" : 1})
>
```

Here mycol is our collection name, as created in this. If the collection doesn't exist in database then MongoDB will create this collection and then insert document into it.

In the inserted document, if we don't specify the _id parameter then MongoDB assign a unique ObjectId for this document.

_id is 12 bytes hexadecimal number unique for every document in a collection 12 bytes are divided as follows

_id : ObjectId (4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)

You can also pass an array of document into the Insert() method as shown below.

```
>db.createCollection("post")
>db.post.insert([
    {
      titele :"MongoDB Overiview ",
     description:"MongoDB is no SQL Database",
      by: "Tutorials Point"
      url: "http//www.tutorialpoint.com")
     tags : ["mongodb","database","NoSQL"],
     likes : 100
   },
   {
   title : "NoSQL Database",
  description:"NoSQL database doesn't have table",
```

```
tags : ["Mongodb", "database", "NoSQL" ],
likes : 20,
Comments : [
    {
        user: "user1",
        message :"My first comment",
        dateCreated : new Date(2013,11,10,2,35),
        like =0
        }
    ]
}
])
BulkWriteResult ({
    "WriteErrors" : [ ],
    "WriteConcernErrors" : [ ],
    "nInserted" : 2,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ],
    } ]
>
```

To insert the document you can use db.post.save(document) also. If you don't specify _id in the document then save() method will work same as insert() method. If you specify _id then. It will replace whole data of document containing _id specified in save() method.

The insertOne() method.
If you need to insert only one document into a collection you can use this method

Syntax:-
>db.COLLECTION_NAME.insertOne(document)

Example:-
>db.createCollection("empDetails")
{"ok" : 1}

>db.empDetails.insertOne(
{
First_Name :"Radhika",
Last-Name :- "Sharma",
Date_Of_Birth "-":"1995-09-26",
email :"radhika_sharma.123@gmail.com",
phone :"9848022338"
})
{
"acknowledged" : true,
"insertedId" : ObjectId("5dd62b4070fb13eec")
}
>

The insertMany() method :-
You can insert multiple document using the insertMany() method. To this method you need to pass an array of documents.

Example :-

```
>db.empDetails.insertMany(
    [
        {
            First-Name : "Radhika",
            Last-Name : "Sharma"
          Date_of_Birth : "1995-09-26",
          email : "radhika-sharma.123@gmail.com",
          phone : "9000123456"
        },
        {
            First_Name : "Rachel",
            Last_Name : "Christopher",
          Date_of_Birth : "1990-02-16",
          email : "Rachel.Christopher.123@gmail.com"
          phone : "9000054321"
        },
        {
            First_Name : "Fathima",
            Last_Name : "Sheik",
         Date_of_Birth : "1990-02-16"
         email : "Fathima.Sheik.123@gmail.com",
          phone : "9000054321"
        }
    ]

)
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5dd631f270fb13eec396"),
        ObjectId("5dd631f270fb13eec396"),
        ObjectId("5dd631f270fb13eecbef"),
    ]
}
>
```