MongoDB

Learning MongoDB

Praveen Nair

Introduction to MongoDB

MongoDB is a document database.

MongoDB is a non-relational, non-tabular database.

Relational data is stored differently.

Instead of having multiple tables all the related data are stored together.

In MongoDB, tables are called collections.

MongoDB can be installed locally or in cloud called MongoDB Atlas

Mongosh or Compass can be used to query MongoDB

MongoDB Installation

https://www.mongodb.com/try/download/community

Choose MSI

Connect to local mongodb

Install Mongosh (https://www.mongodb.com/try/download/shell)

Type mongosh –version

Type mongosh to get prompt

show dbs

use myproj to create or access new db

db.dropDatabase(“dbname”) to delete database (or db.dropDatabase())

show collections

db.createCollection(“employees")

db.createCollection(“employees“,{capped:true,size:100,max:100) //max 100 employees, size max 100 bytes. Delets oldest document

db.emploees.drop() to delete collection

db.restaurant.renameCollection('restaurants’)   //rename collection

Case sensitive

Inserting Data

```
db.employees.insertOne({
  name: "John Smith",
  email: "john@gmail.com",
  department: "IT",
  salary: 1456,
  location: ["FL", "OH"],
  date: Date()
})
db.employees.find()
```

Inserting Multiple Data

```
db.employees.insertMany([{
  name: "Mike Joseph",
  email: "mike@gmail.com",
  department: "IT",
  salary: 2456,
  location: ["FL", "TX"],
  date: Date()
},
{ name: "Cathy G",
  email: "cathy@gmail.com",
  department: "IT",
  salary: 3456,
  location: ["AZ", "TX"],
  date: Date()
}])
```

Find Data

returns first 20, then type it for more documents

```
db.employees.find()  //returns first 20, then type it for more documents
db.employees.find().skip(2)
db.employees.findOne()
db.users.find().sort({name:1})  //sorting -1 for reverse
 db.users.find().limit(1)  //returns 1 document sort by object id
db.users.find().sort({name:1}).limit(3)
db.employees.find( {department: "IT"} )
db.users.find({name:"Cathy",pass:"1234"})  //two condition
db.employees.find({}, {_id: 0, salary: 1, date: 1})  //cannot give 0
db.users.find({},{_id:false,name:true}) //cannot give false
db.employees.find({}, {_id: 0, salary: 0, date: 1}) //either use 0 or 1, can’t use both
 db.users.find({'address.city':'Gwenborough’})  //query nested documents
db.users.find({address.geo.lat:'-37.3159’})
db.employees.find({'location':'TX’})  //where location : [‘FL’,’TX’]
db.users.find().count()
db.employees.find({},{"dept":"$department",email:1,salary:1})  //dept is alias
```

Update Document

```
db.users.find({'address.city':'Gwenborough’})  //query nested documents
db.users.find({address.geo.lat:'-37.3159’})
db.employees.find({'location':'TX’})  //where location : [‘FL’,’TX’]
```

Update Document

```
db.employees.updateOne({email:'cathy@gmail.com'},{$set:{department:'HR'}})

db.employees.updateOne(
  { email: "ria@gmail.com" },
  {
    $set:
      {
        name: "Ria K",
        email: "ria@gmail.com",
        department: "HR",
        salary: 5000,
        location: ["FL", "LA"],
        date: Date()
      }
  },
  { upsert: true }
)

db.employees.updateMany({}, { $set: { date: Date() } })
```

Delete Document

```
db.employees.deleteOne({email:'ria@gmail.com'})
db.employees.deleteMany({email:'ria@gmail.com'})
```

Query Operators

```
db.employees.find({department:{$eq:'HR’}})
db.users.find({email:{$ne:'cathy@gmail.com'}})
db.employees.find({salary:{$gt:3000}})
db.employees.find({salary:{$gte:3000}})
db.employees.find({salary:{$gte:3000,$lt:5000}})
db.employees.find({salary:{$gt:1000},department:{$eq:'HR'}})
db.employees.find({salary:{$gt:2000},department:{$in:['HR','IT']}})
db.employees.find({salary:{$gt:2000},department:{$nin:['HR','IT']}})
db.employees.find({$or:[{salary:{$gt:2000}},{department:{$eq:'HR'}}]})
db.employees.find({$and:[{salary:{$gt:2000}},{department:{$eq:'HR'}}]})
db.employees.find({$nor:[{salary:{$gt:2000}},{department:{$eq:'HR'}}]})    //like and but both should be false
db.employees.find({department:{$not:{$eq:'HR'}}})
db.users.find({email1:{$exists:false}})
```

Update Operators(fields)

```
db.employees.updateOne({email:'cathy@gmail.com'},{$set:{email:'cathy@hot
mail.com'}})

db.employees.updateMany({},{$set:{points:0}})    -- new field

db.employees.updateMany({},{$inc:{points:70}})

db.employees.updateMany({},{$rename:{points:'score'}})

db.employees.updateMany({},{$unset:{score:""}})  //deletes the field
```

Summary - CRUD

```
db.users.find({filter},{projection})
db.users.insertOne({document})
db.users.insertMany([{document},{document}]
db.users.deleteMany({filter})
db.users.updateMany({filter},{$set:{flag:false}})
db.users.updateMany({filter},{$unset:{flag:""}})
db.users.updateMany({filter},{$inc:{score:20}})  //increment by 20
db.users.updateMany({filter},{$rename:{flag:"indicator"}})
db.users.find({$and:[{},{}]})
```

Update Operators (arrays)

db.employees.updateOne({email:'cathy@hotmail.com'},{$addToSet:{location:'FL’}}) //duplicates won’t be added, use push instead

db.employees.updateOne({email:'cathy@hotmail.com'},{$pop:{location:1}}) – try -1

db.employees.updateMany({email:'cathy@hotmail.com'},{$pull:{points:{$gt:1}}})

db.employees.updateMany({email:'cathy@hotmail.com'},{$push:{points:5}})

Misc – skip and limit

```
db.employees.find().skip(2)
db.employees.find().skip(2).limit(1)
```

Used for pagination

Aggregation Operations

Processes multiple documents and return computed results

```
db.employees.count({department:'IT’})
db.employees.distinct("department")
```

Aggregation - $match

```
db.employees.aggregate([
  {
    $match: {}   //stage 1
  },
  {
    $group: { _id: "$department", total: { $sum: "$salary" } }  //stage 2
  },
{
    $sort: { “department": -1 }
  },
])
```

Aggregation - $match

```
db.employees.aggregate([
 {
   $match: { salary: { $gt: 1000 } }   //state 1
 },
 {
   $group: { _id: "$department", total: { $sum: "$salary" } }  //stage 2
 }
])
```

Aggregation - $group

An aggregation pipeline return results for groups of documents. For example, return the total, average, maximum, and minimum values.

```
db.employees.aggregate([
  {
    $group: {
      _id: "$department",
      Total: { $sum: "$salary" },
      Hightest: { $max: "$salary" },
      Lowest: { $min: "$salary" },
      Average: { $avg: "$salary" },
    },
  },
]);
```

Aggregation - $limit

```
db.employees.aggregate([
  { $group: { _id: "$department", Total: { $sum: "$salary" } } },
  { $limit: 1 },
]);
```

Aggregation - $project

```
db.employees.aggregate([
 {
   $project: {
     "name": 1,
     "email": 1,
     "salary": 1
   }
 },
 {
   $limit: 2
 }
])
```

$project – remove field

```
db.employees.aggregate([{ $project: { _id: 0, name: 0 } }]);
```

$project – rename & add calc

```
db.employees.aggregate([
  {
    $project: {
      empname: "$name",
      email: 1,
      salary: 1,
      AnnualSalary: { $multiply: [12, "$salary"] },
    },
  },
]);
```

Aggregation - $sort

```
db.employees.aggregate([
 {
   $sort: { "name": -1 }
 },
 {
   $project: {
     "name": 1,
     "email": 1,
     "salary":1
   }
 },
 {
   $limit: 5
 }
])
```

Aggregation - $count

```
db.employees.aggregate([
 {
   $match: { "department": "IT" }
 },
 {
   $count: "totalEmp"
 }
])
```

Aggregation - $addFields

```
db.employees.aggregate([
  {
    $addFields: {
      avgPoints: { $avg: "$points" }
    }
  },
  {
    $project: {
      "department": 1,
      "avgPoints": 1
    }
  },
  {
    $limit: 5
  }
])
```