In this chapter, we will learn how to query document from MongoDB collection.

## The find() Method

To query data from MongoDB collection, you need to use MongoDB's **find()** method.

### Syntax

The basic syntax of **find()** method is as follows −

```
>db.COLLECTION_NAME.find()
```

**find()** method will display all the documents in a non-structured way.

### Example

Assume we have created a collection named mycol as −

```
> use sampleDB
switched to db sampleDB
> db.createCollection("mycol")
{ "ok" : 1 }
>
```

And inserted 3 documents in it using the insert() method as shown below −

```
> db.mycol.insert([
        {
                title: "MongoDB Overview",
			description: "MongoDB is no SQL database",
			by: "authorname",
			url: "http://www.tutorialspoint.com",
			tags: ["mongodb", "database", "NoSQL"],
			likes: 100
		},
		{
			title: "NoSQL Database",
			description: "NoSQL database doesn't have tables",
			by: "tutorials point",
			url: "http://www.tutorialspoint.com",
			tags: ["mongodb", "database", "NoSQL"],
			likes: 20,
			comments: [
				{
					user:"user1",
					message: "My first comment",
					dateCreated: new Date(2013,11,10,2,35),
					like: 0
				}
			]
		}
])
```

Following method retrieves all the documents in the collection −

```
> db.mycol.find()
{ "_id" : ObjectId("5dd4e2cc0821d3b44607534c"), "title" : "MongoDB
Overview", "description" : "MongoDB is no SQL database", "by" :
"tutorials point", "url" : "http://www.tutorialspoint.com", "tags" : [
"mongodb", "database", "NoSQL" ], "likes" : 100 }
{ "_id" : ObjectId("5dd4e2cc0821d3b44607534d"), "title" : "NoSQL
Database", "description" : "NoSQL database doesn't have tables", "by" :
"tutorials point", "url" : "http://www.tutorialspoint.com", "tags" : [
"mongodb", "database", "NoSQL" ], "likes" : 20, "comments" : [ {
```

```
"user" : "user1", "message" : "My first comment", "dateCreated" :
ISODate("2013-12-09T21:05:00Z"), "like" : 0 } ] }
>
```

## The pretty() Method

To display the results in a formatted way, you can use pretty() method.

Syntax

```
>db.COLLECTION_NAME.find().pretty()
```

Example

Following example retrieves all the documents from the collection named mycol and arranges them in an easy-to-read format.

```
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5dd4e2cc0821d3b44607534c"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5dd4e2cc0821d3b44607534d"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-
09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
```

The findOne() method

Apart from the find() method, there is **findOne()** method, that returns only one document.

Syntax

```
>db.COLLECTIONNAME.findOne()
```

Example

Following example retrieves the document with title MongoDB Overview.

```
> db.mycol.findOne({title: "MongoDB Overview"})
{
        "_id" : ObjectId("5dd6542170fb13eec3963bf0"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
```

RDBMS Where Clause Equivalents in MongoDB

To query the document on the basis of some condition, you can use following operations.

| Operation | Syntax | Example |
|---|---|---|
| Equality | {<key>:{$eg;<value>}} | db.mycol.find({"by":"tutorials point"}).pretty() |
| Less Than | {<key>:{$lt:<value>}} | db.mycol.find({"likes":{$lt:50}}).pretty |
| Less Than Equals | {<key>:{$lte:<value>}} | db.mycol.find({"likes":{$lte:50}}).pret |
| Greater Than | {<key>:{$gt:<value>}} | db.mycol.find({"likes":{$gt:50}}).prett |
| Greater Than Equals | {<key>:{$gte:<value>}} | db.mycol.find({"likes":{$gte:50}}).pre |
| Not Equals | {<key>:{$ne:<value>}} | db.mycol.find({"likes":{$ne:50}}).pret |
| Values in an array | {<key>:{$in:[<value1>, <value2>,……<valueN>]}} | db.mycol.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty() |

| | | |
|---|---|---|
| | | |
| Values not in an array | {<key>:{$nin:<value>}} | db.mycol.find({"name":{$nin:["Ramu "Raghav"]}}).pretty() |

## AND in MongoDB

### Syntax

To query documents based on the AND condition, you need to use $and keyword. Following is the basic syntax of AND −

```
>db.mycol.find({ $and: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

### Example

Following example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'.

```
> db.mycol.find({$and:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty()
{
        "_id" : ObjectId("5dd4e2cc0821d3b44607534c"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
>
```

For the above given example, equivalent where clause will be **' where by = 'tutorials point' AND title = 'MongoDB Overview' '**. You can pass any number of key, value pairs in find clause.

OR in MongoDB

Syntax

To query documents based on the OR condition, you need to use **$or** keyword. Following is the basic syntax of **OR** −

```
>db.mycol.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

Example

Following example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'.

```
>db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB
Overview"}]}).pretty()
{
   "_id": ObjectId(7df78ad8902c),
   "title": "MongoDB Overview",
   "description": "MongoDB is no sql database",
   "by": "tutorials point",
   "url": "http://www.tutorialspoint.com",
   "tags": ["mongodb", "database", "NoSQL"],
   "likes": "100"
}
>
```

## Using AND and OR Together

### Example

The following example will show the documents that have likes greater than 10 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is **'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'**

```
>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},
   {"title": "MongoDB Overview"}]}).pretty()
{
   "_id": ObjectId(7df78ad8902c),
   "title": "MongoDB Overview",
   "description": "MongoDB is no sql database",
   "by": "tutorials point",
   "url": "http://www.tutorialspoint.com",
   "tags": ["mongodb", "database", "NoSQL"],
   "likes": "100"
}
>
```

## NOR in MongoDB

### Syntax

To query documents based on the NOT condition, you need to use $not keyword. Following is the basic syntax of **NOT −**

```
>db.COLLECTION_NAME.find(
	{
		$not: [
			{key1: value1}, {key2:value2}
		]
	}
)
```

Example

Assume we have inserted 3 documents in the collection **empDetails** as shown below −

```
db.empDetails.insertMany(
	[
		{
			First_Name: "Radhika",
			Last_Name: "Sharma",
			Age: "26",
			e_mail: "radhika_sharma.123@gmail.com",
			phone: "9000012345"
		},
		{
			First_Name: "Rachel",
			Last_Name: "Christopher",
			Age: "27",
			e_mail: "Rachel_Christopher.123@gmail.com",
			phone: "9000054321"
		},
		{
			First_Name: "Fathima",
			Last_Name: "Sheik",
			Age: "24",
			e_mail: "Fathima_Sheik.123@gmail.com",
			phone: "9000054321"
		}
	]
)
```

Following example will retrieve the document(s) whose first name is not "Radhika" and last name is not "Christopher"

```
> db.empDetails.find(
        {
                $nor:[
                        40
                        {"First_Name": "Radhika"},
                        {"Last_Name": "Christopher"}
                ]
        }
).pretty()
{
        "_id" : ObjectId("5dd631f270fb13eec3963bef"),
        "First_Name" : "Fathima",
        "Last_Name" : "Sheik",
        "Age" : "24",
        "e_mail" : "Fathima_Sheik.123@gmail.com",
        "phone" : "9000054321"
}
```

NOT in MongoDB

Syntax

To query documents based on the NOT condition, you need to use $not keyword following is the basic syntax of **NOT −**

```
>db.COLLECTION_NAME.find(
        {
                $NOT: [
                        {key1: value1}, {key2:value2}
                ]
        }
).pretty()
```

Example

Following example will retrieve the document(s) whose age is not greater than 25

```
> db.empDetails.find( { "Age": { $not: { $gt: "25" } } } )
{
        "_id" : ObjectId("5dd6636870fb13eec3963bf7"),
        "First_Name" : "Fathima",
        "Last_Name" : "Sheik",
        "Age" : "24",
        "e_mail" : "Fathima_Sheik.123@gmail.com",
        "phone" : "9000054321"
}
```

MongoDB's **update()** and **save()** methods are used to update document into a collection. The update() method updates the values in the existing document while the save() method replaces the existing document with the document passed in save() method.

MongoDB Update() Method

The update() method updates the values in the existing document.

Syntax

The basic syntax of **update()** method is as follows −

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA,
UPDATED_DATA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB
Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL
Overview"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point
Overview"}
```

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New
MongoDB Tutorial'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB
Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL
Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point
Overview"}
>
```

By default, MongoDB will update only a single document. To update multiple documents, you need to set a parameter 'multi' to true.

```
>db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

## MongoDB Save() Method

The **save()** method replaces the existing document with the new document passed in the save() method.

### Syntax

The basic syntax of MongoDB **save()** method is shown below −

```
>db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
```

### Example

Following example will replace the document with the _id '5983548781331adf45ec5'.

```
>db.mycol.save(
   {
      "_id" : ObjectId("507f191e810c19729de860ea"),
                    "title":"Tutorials Point New Topic",
      "by":"Tutorials Point"
   }
)
WriteResult({
           "nMatched" : 0,
           "nUpserted" : 1,
           "nModified" : 0,
           "_id" : ObjectId("507f191e810c19729de860ea")
})
>db.mycol.find()
{ "_id" : ObjectId("507f191e810c19729de860e6"), "title":"Tutorials
Point New Topic",
   "by":"Tutorials Point"}
{ "_id" : ObjectId("507f191e810c19729de860e6"), "title":"NoSQL
Overview"}
{ "_id" : ObjectId("507f191e810c19729de860e6"), "title":"Tutorials
Point Overview"}
>
```

## MongoDB findOneAndUpdate() method

The **findOneAndUpdate()** method updates the values in the existing document.

### Syntax

The basic syntax of **findOneAndUpdate()** method is as follows −

```
>db.COLLECTION_NAME.findOneAndUpdate(SELECTIOIN_CRITE
RIA, UPDATED_DATA)
```

### Example

Assume we have created a collection named empDetails and inserted three documents in it as shown below −

```
> db.empDetails.insertMany(
	[
		{
			First_Name: "Radhika",
			Last_Name: "Sharma",
			Age: "26",
			e_mail: "radhika_sharma.123@gmail.com",
			phone: "9000012345"
		},
		{
			First_Name: "Rachel",
			Last_Name: "Christopher",
			Age: "27",
			e_mail: "Rachel_Christopher.123@gmail.com",
			phone: "9000054321"
		},
		{
			First_Name: "Fathima",
			Last_Name: "Sheik",
			Age: "24",
			e_mail: "Fathima_Sheik.123@gmail.com",
			phone: "9000054321"
		}
	]
)
```

Following example updates the age and email values of the document with name 'Radhika'.

```
> db.empDetails.findOneAndUpdate(
	{First_Name: 'Radhika'},
	{ $set: { Age: '30',e_mail: 'radhika_newemail@gmail.com'}}
)
```

```
{
        "_id" : ObjectId("5dd6636870fb13eec3963bf5"),
        "First_Name" : "Radhika",
        "Last_Name" : "Sharma",
        "Age" : "30",
        "e_mail" : "radhika_newemail@gmail.com",
        "phone" : "9000012345"
}
```

MongoDB updateOne() method

This methods updates a single document which matches the given filter.

Syntax

The basic syntax of updateOne() method is as follows −

```
>db.COLLECTION_NAME.updateOne(<filter>, <update>)
```

Example

```
> db.empDetails.updateOne(
        {First_Name: 'Radhika'},
        { $set: { Age: '30',e_mail: 'radhika_newemail@gmail.com'}}
)
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
>
```

MongoDB updateMany() method

The updateMany() method updates all the documents that matches the given filter.

Syntax

The basic syntax of updateMany() method is as follows −

```
>db.COLLECTION_NAME.update(<filter>, <update>)
```

Example

```
> db.empDetails.updateMany(
	{Age:{ $gt: "25" }},
	{ $set: { Age: '00'}}
)
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
```

You can see the updated values if you retrieve the contents of the document using the find method as shown below −

```
> db.empDetails.find()
{ "_id" : ObjectId("5dd6636870fb13eec3963bf5"), "First_Name" :
"Radhika", "Last_Name" : "Sharma", "Age" : "00", "e_mail" :
"radhika_newemail@gmail.com", "phone" : "9000012345" }
{ "_id" : ObjectId("5dd6636870fb13eec3963bf6"), "First_Name" :
"Rachel", "Last_Name" : "Christopher", "Age" : "00", "e_mail" :
"Rachel_Christopher.123@gmail.com", "phone" : "9000054321" }
{ "_id" : ObjectId("5dd6636870fb13eec3963bf7"), "First_Name" :
"Fathima", "Last_Name" : "Sheik", "Age" : "24", "e_mail" :
"Fathima_Sheik.123@gmail.com", "phone" : "9000054321" }
>
```

In this chapter, we will learn how to delete a document using MongoDB.

The remove() Method

MongoDB's **remove()** method is used to remove a document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

- **deletion criteria** − (Optional) deletion criteria according to documents will be removed.
- **justOne** − (Optional) if set to true or 1, then remove only one document.

Syntax

Basic syntax of **remove()** method is as follows −

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

Example

Consider the mycol collection has the following data.

```
{_id : ObjectId("507f191e810c19729de860e1"), title: "MongoDB
Overview"},
{_id : ObjectId("507f191e810c19729de860e2"), title: "NoSQL
Overview"},
{_id : ObjectId("507f191e810c19729de860e3"), title: "Tutorials Point
Overview"}
```

Following example will remove all the documents whose title is 'MongoDB Overview'.

```
>db.mycol.remove({'title':'MongoDB Overview'})
WriteResult({"nRemoved" : 1})
> db.mycol.find()
{"_id" : ObjectId("507f191e810c19729de860e2"), "title" : "NoSQL
Overview" }
{"_id" : ObjectId("507f191e810c19729de860e3"), "title" : "Tutorials
Point Overview" }
```

Remove Only One

If there are multiple records and you want to delete only the first record, then set **justOne** parameter in **remove()** method.

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. **This is equivalent of SQL's truncate command.**

```
> db.mycol.remove({})
```

```
WriteResult({ "nRemoved" : 2 })
> db.mycol.find()
>
```