Understanding Query Filters and Query Operators

Dan Geabunea
SENIOR SOFTWARE DEVELOPER

@romaniancoder www.romaniancoder.com

Overview

Finding documents in a collection

Query filter documents

Comparison query selectors

Geospatial query selectors

Demo: Filtering data in Mongo collections

The Collection Find Methods

find()

Selects multiple documents from a collection

findOne()

Selects a single document from a collection

The Collection Find Method

```
find(query, projection): cursor
```

INPUT:

query	(Optional) Filtering using query operators
projection	(Optional) Specify which fields to display/hide

OUTPUT:

cursor	Cursor to the documents that match the query criteria

The Collection Find One Method

```
findOne(query, projection): document
```

INPUT:

query	(Optional) Filtering using query operators
projection	(Optional) Specify which fields to display/hide

OUTPUT:

document	The found document

Find One Results

By default, it returns the document that matches the criteria

If multiple documents match the criteria, the method returns the first document according to the order the documents are stored on disk

If no documents match the criteria, the method returns null

```
// Incorrect
> db.aircraft.findOne().pretty()
> db.aircraft.findOne().count()
```

Find One Returns a Document

The cursor methods are not available because they do not make sense

If you try to use them, you will get an exception

```
db.aircraft.find()          // Returns all the documents from the collection

db.aircraft.find({})        // Returns all the documents from the collection

db.aircraft.find({}, {_id: 0})     // Obliged to use a query filter document

db.aircraft.find({_id: 0})         // Incorrect, interpreted as an equality query
```

Empty Query

No filtering applied; all documents returned

- It has no effect from a filtering perspective
- But you are obliged to use it when you have projections

Comparison Query Selectors

Query Filter Documents

A JSON object which consists of criteria which determine weather a document should be included or excluded from the result set

Syntax

```
{ field : { $operator : value } }
```

Comparison Query Operators

$eq

$ne

$in

$nin

Comparison Query Operators

$lt

$lte

$gt

$gte

You will use the comparison query operators a lot

Equality

Syntax

```
find({field : value}, projection)

findOne({field : value}, projection)
```

INPUT:

field	Name of the field containing the value
value	The value you wish to compare against

Examples

Comparing Strings

```
// compare strings, multiple results

db.aircraft.find({model: "Boeing 737-900"})


// compare strings, single result

db.aircraft.findOne({code: "eede6be6-f716-4e2e-bf81-885f0a16a50c"})
```

Examples

Comparing Numbers

```
// compare numbers

db.aircraft.find({range: 5600})


// compare numbers, implicit conversion, same output as above

db.aircraft.find({range: 5600.00})
```

Examples

Comparing Booleans

```
// compare Booleans, true

db.aircraft.find({underMaintenance: true})


// compare Booleans, false

db.aircraft.find({underMaintenance: false})
```

Examples

Comparing Dates

```
// compare dates

db.flights.find({departureDate: ISODate("2020-02-20T23:00:00Z")})


db.flights.find({departureDate: new Date("2020-02-20T23:00:00Z")})


// don’t forget about the time

db.flights.find({departureDate: new Date("2020-02-20")})
```

Examples

Other Types

```
// compare ids

db.aircraft.find({_id: ObjectId("5e8aa971e1562c14d031a021")})


// compare arrays

db.crew.find({skills: ["engineering"]})

db.crew.find({skills: ["engineering“, “planning”]})


// compare objects

db.crew.find({address: {city: “Paris”, country: “France”}})
```

“Wait a minute…I only saw equality comparisons. What about inequality?”

You

Another Syntax

```
find({field: { $eq : value } }, projection)

find({field: { $ne : value } }, projection)
```

INPUT:

field	Name of the field containing the value
operator	Equality ($eq), or non-equality ($neq)
value	The value you wish to compare against

Examples

Equality Equivalence

```
// compare strings with ‘$eq’

db.aircraft.find({model: { $eq : "Boeing 737-900“ }})


// compare numbers with ‘:’

db.aircraft.find({model: “Boeing 737-900”})
```

Examples

Non-equality

```
// compare strings with $ne

db.aircraft.find({model: { $ne : "Boeing 737-900“ }})


// compare numbers with $ne

db.aircraft.find({range: { $ne : 5600 }})


// compare Booleans with $ne

db.aircraft.find({inMaintenance: { $ne : true }})
```

Less Than / Greater Than

Query Operators

$lt

Compare fields that are less than a specific value

$lte

Compare fields that are less than or equal to a specific value

$gt

Compare fields that are greater than a specific value

$gte

Compare fields that are greater than or equal to a specific value

Strict Comparison Syntax

```
find({field: { $gt : value } }, projection)

find({field: { $lt : value } }, projection)
```

INPUT:

field	Name of the field containing the value
operator	Greater than ($gt) or less than ($lt)
value	The value you wish to compare against

Loose Comparison Syntax

```
find({field: { $gte : value } }, projection)

find({field: { $lte : value } }, projection)
```

INPUT:

field	Name of the field containing the value
operator	Greater than or equal ($gte) / less than or equal ($lte)
value	The value you wish to compare against

Examples

Comparing Numbers

```
db.aircraft.find({capacity: { $gt : 200 }})
```

```
db.aircraft.find({capacity: { $lt : 200 }})
```

```
db.aircraft.find({capacity: { $gte : 200 }})
```

```
db.aircraft.find({capacity: { $lte : 200 }})
```

Examples

Comparing Floating Point and Integers

```
// range is an integer number
db.aircraft.find({range: { $gt : 900.00 }})

// range is an integer number
db.aircraft.find({range: { $lt : 1199.99 }})
```

Examples

Comparing Dates

```
db.aircraft.find({nextMaintenance: { $gt : ISODate("2020-02-20") }})
```

```
db.aircraft.find({nextMaintenance: { $lt : ISODate("2020-02-20") }})
```

```
db.aircraft.find({nextMaintenance: { $gte : ISODate("2020-02-20T10:15:00Z") }})
```

```
db.aircraft.find({nextMaintenance: { $lte : ISODate("2020-02-20T10:15:00Z") }})
```

In / Not In

Query Operators

$in

Select documents where the value of a field equals any value in a specified array

$nin

Select documents where the value of a field is not found in a specified array

Syntax

```
find({field: { $in : [v1,v2] } }, projection)

find({field: { $nin : [v1,v2] } }, projection)
```

INPUT:

field	Name of the field containing the value
operator	Is in ($in) / is not in ($nin)
value	An array of values you wish to compare against

Examples

Using Non-array Fields

```
// Aircraft { “model" : “Boeing 747", …}

db.aircraft.find({ model: { $in: ["Airbus A350“, “Boeing 747”] }})


db.aircraft.find({ model: { $nin: ["Airbus A350“, “Boeing 747”] }})
```

Examples

Using Array Fields

```
// Crew Member { "name" : "Anna Smith", "skills" : [ "technical", "management" ] }

db.crew.find({skills: { $in: ["engineering", "management"] } }) // match
db.crew.find({skills: { $in: ["engineering", “analysis"] } }}  // no match

db.crew.find({skills: { $nin: [“advanced landing techniques”] } })
```

Examples

Using Regular Expressions

```
// All aircraft that start with ‘A’
db.aircraft.find({ model: { $in: [/^A/] } })

// All aircraft that start with ‘A’ or contain 737
db.aircraft.find({ model: { $in: [/^A/, /737/] } })
```

Geospatial Queries

Geo Spatial Data Types

GeoJSON

Legacy

Geo JSON vs. Legacy

Geo JSON Coordinates

Calculate geometry on a sphere

MongoDB uses the WGS84 projection system for queries

Needs a ‘2dsphere’ index on the geographical field

Legacy Coordinates

Calculate geometry on a plane

Needs a ‘2dindex’

However, MongoDB supports spherical calculations by using a ‘2dsphere’ index

```
{
    callsign: “RO123”,
    position: { type: “Point”, coordinates: [35.7, 47.5] },
    altitude: 10000
}
db.radar.createIndex( { “position” : "2dsphere" } )
```

Geo JSON Point

Geo JSON data is stored in a sub-document containing

- A field named ‘type’
- A field named ‘coordinates’
  - If you want to coordinates, longitude is first, latitude after

$near

Filters documents where a location field is between a min and max value in meters from a specified geometry

Syntax

```
find({ field : { $near :{

   }
}, projection)
```

Syntax

```
find({ field : { $near :{
    $geometry: {
        type: “Point”, coordinates: [lon,lat]
    }
}
}, projection)
```

Syntax

```
find({ field : { $near :{
    $geometry: {
        type: “Point”, coordinates: [lon,lat]
    },
    $minDistance : value_meters,
    $maxDistance : value_meters
  }
}, projection)
```

Find the Aircraft within 10km of a Point

```
db.radar.find({position: {$near : {
    $geometry: {type: "Point", coordinates: [26.2, 44.4]},
    $maxDistance: 10000
    }}
}).pretty()
```

Demo

Filtering data in Mongo collections

- Comparison
- Geo spatial

Summary

Anatomy and uses cases for the find methods

Comparison query selectors

- Equality
- Greater than / less than
- In / not in

Geo spatial queries

- Near

“I am now confident in creating some queries...but what if I want to add multiple criteria to my filters or search sub documents?”

You

Up Next
Creating Complex Queries