**Understanding Basics and CRUD Operations**

* **Understanding Databases, Collections & Documents**
  + Command to run the mongoDB in different port
    - **mongod --port 27018**
  + Databases, Collections and Documents

A screenshot of a cell phone

Description automatically generated

* **Creating Databases and Collections**

show dbs – lists all available dbs.

use db – to use the DB

Note: We can use this command to use the non-existing DB also, if the DB is not exist it will create a new one. But after creating it until unless you add a collection to the DB, it will not be visible (it will not list in the show dbs command)

use flights

To insert data:

db.flightData.insertOne({{departureAirport: "MUC", arrivalAirport:"SFO", aircraft: "Airbus A380",distance: 12000, intercontinental: true}})

db.flightData.find() – Reads all the data from the collection. And the result will not be formatted (JSON format).

db.flightData.find().pretty() – Reads and format the result.

* **Comparing JSON & BSON:**

**A screenshot of a cell phone

Description automatically generated**

It identifies some predefined Types in JSON file.

Whenever we insert a document on MongoDB, it will automatically create an id, if you don’t want to use the auto-generated id, you can create your own, the name must be this \_id and give any value.

{

"\_id" : ObjectId("5f15cee7efd591cc5955acce"),

"departureAirport" : "MUC",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

"intercontinental" : true

}

{

"\_id" : "txs-sfo-1",

"departureAirport" : "TXS",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

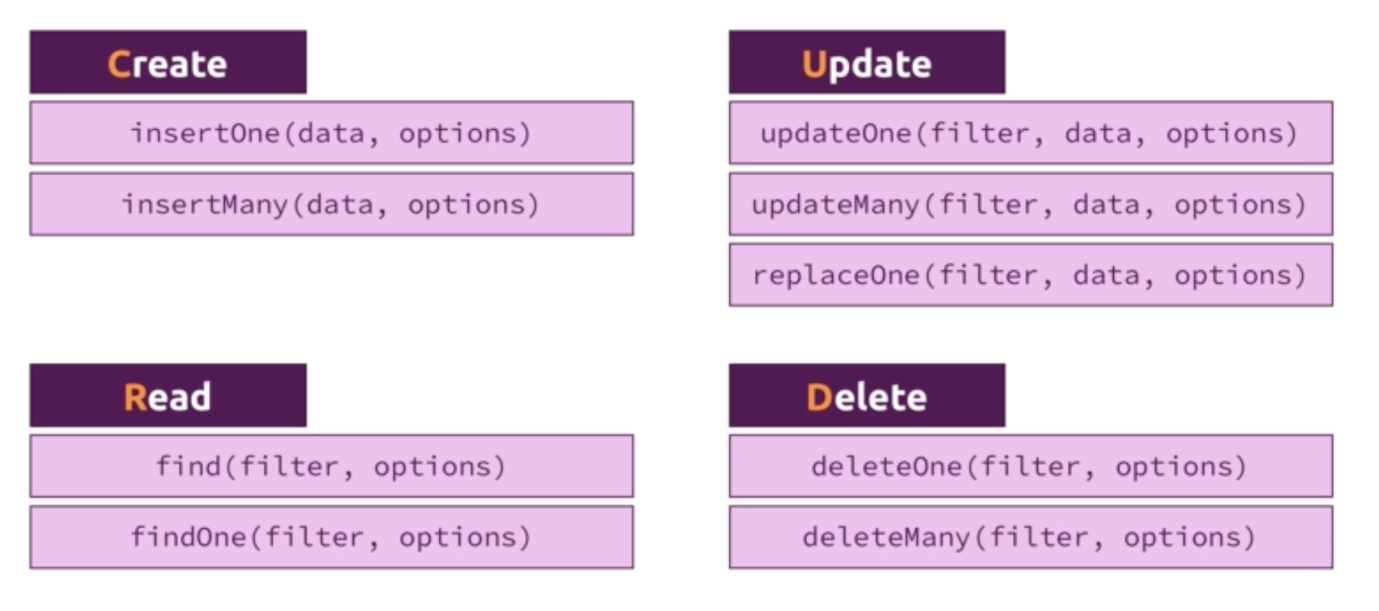
"intercontinental" : true

}

* **Create, Read, Update & Delete on MongoDB:**

**A screenshot of a social media post

Description automatically generated**

****

**A screenshot of a cell phone

Description automatically generated**

Delete an entry(document) in the collection:

db.flightData.deleteOne({“departureAirport”: “MUC”})

Note: I faced an issue with filter if I don’t pass the filter criteria within double quotes.

Update an entry:

db.flightData.updateOne({distance: 12000}, {$set: {marker:”delete”}})

db.flightData.updateMany({}, {$set: {marker: “toDelete”}})

Delete many:

db.flightData.deleteMany({“marker”: “toDelete”})

* **insertMany()**

dbflightDatainsertMany([

{

"departureAirport": "MUC",

"arrivalAirport": "SFO",

"aircraft": "Airbus A380",

"distance": 12000,

"intercontinental": true

},

{

"departureAirport": "LHR",

"arrivalAirport": "TXL",

"aircraft": "Airbus A320",

"distance": 950,

"intercontinental": false

}

])

* **Deep dive in finding data:**

Filter in find:

db.flightData.find({“intercontinental”: true}).pretty()

Filter with greater than:

db.flightData.find({“distance”: {$gt: 9000 }}).pretty()

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54d"),

"departureAirport" : "MUC",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

"intercontinental" : true

}

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54e"),

"departureAirport" : "LHR",

"arrivalAirport" : "TXL",

"aircraft" : "Airbus A320",

"distance" : 950,

"intercontinental" : false

}

Finding the first occurrence in the filtered data:

db.flightData.findOne({“distance”: {$gt: 900}})

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54d"),

"departureAirport" : "MUC",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

"intercontinental" : true

}

* **update vs updateMany:**

db.flightData.updateOne({\_id: ObjectId("5f1b3414f4ade3fbde32a54d")}, {$set: {delayed: true}})

db.flightData.update({\_id: ObjectId("5f1b3414f4ade3fbde32a54d")}, {$set: {delayed: false}})

Update method can be used without set also, when we use $set, it adds the given attribute to the existing attributes. But without set if we use update(), it clears / removes all the attribute and update the given attribute.

Example:

db.flightData.update({\_id: ObjectId("5f1b3414f4ade3fbde32a54d")}, {delayed: true})

{ "\_id" : ObjectId("5f1b3414f4ade3fbde32a54d"), "delayed" : true }

Replace:

dbflightDatareplaceOne({\_id: ObjectId("5f1b3414f4ade3fbde32a54d")}, {"departureAirport": "MUC",

"arrivalAirport": "SFO",

"aircraft": "Airbus A380",

"distance": 12000,

"intercontinental": true})

* **Understanding find() and Cursor object:**

**Cursor:**

Whenever we use find() on a collection, it will not return us the collection itself, it will return the Cursor, let say it if returns the entire collection, in case of large volumes of data, it needs more memory and it has to sent through wires.

A screenshot of a cell phone

Description automatically generated

**Note:** By default, mongoDB gives first 20 documents from the given collection.

If you use toArray() method with find(), it will not returns the cursor, it will return all the documents.

If your collection have more than 20 documents and if you just use find(), it will return only first 20 documents. Do get the more number documents, you have to type it.

Or else you can use toArray() or forEach()

db.passengers.find().toArray()

db.passengers.find().forEach((passengerData) => {printjson(passengerData)})

Note: findOne() will not gives a cursor, it will return the document directly.

* **Understanding Projection:**

**A picture containing screenshot

Description automatically generated**

**db.passengers.find({}, {name: 1}).pretty()**

-> Above query will still return the \_id as it a default one.

{

"\_id" : ObjectId("5f1be4d2f4ade3fbde32a54f"),

"name" : "Max Schwarzmueller"

}

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a550"), "name" : "Manu Lorenz" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a551"), "name" : "Chris Hayton" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a552"), "name" : "Sandeep Kumar" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a553"), "name" : "Maria Jones" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a554"), "name" : "Alexandra Maier" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a555"), "name" : "Dr. Phil Evans" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a556"), "name" : "Sandra Brugge" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a557"), "name" : "Elisabeth Mayr" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a558"), "name" : "Frank Cube" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a559"), "name" : "Karandeep Alun" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55a"), "name" : "Michaela Drayer" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55b"), "name" : "Bernd Hoftstadt" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55c"), "name" : "Scott Tolib" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55d"), "name" : "Freddy Melver" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55e"), "name" : "Alexis Bohed" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a55f"), "name" : "Melanie Palace" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a560"), "name" : "Armin Glutch" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a561"), "name" : "Klaus Arber" }

{ "\_id" : ObjectId("5f1be4d2f4ade3fbde32a562"), "name" : "Albert Twostone" }

If you want to omit the id from projection, use below query.

**db.passengers.find({}, {name: 1, \_id: 0}).pretty()**

{ "name" : "Max Schwarzmueller" }

{ "name" : "Manu Lorenz" }

{ "name" : "Chris Hayton" }

{ "name" : "Sandeep Kumar" }

{ "name" : "Maria Jones" }

{ "name" : "Alexandra Maier" }

{ "name" : "Dr. Phil Evans" }

{ "name" : "Sandra Brugge" }

{ "name" : "Elisabeth Mayr" }

{ "name" : "Frank Cube" }

{ "name" : "Karandeep Alun" }

{ "name" : "Michaela Drayer" }

{ "name" : "Bernd Hoftstadt" }

{ "name" : "Scott Tolib" }

{ "name" : "Freddy Melver" }

{ "name" : "Alexis Bohed" }

{ "name" : "Melanie Palace" }

{ "name" : "Armin Glutch" }

{ "name" : "Klaus Arber" }

{ "name" : "Albert Twostone" }

* **Embedded Documents & Array:**

**Embedded Document:**

**A picture containing screenshot

Description automatically generated**

**Arrays:**

**A picture containing screenshot

Description automatically generated**

**Adding nested document:**

**db.flightData.updateMany({}, {$set: {status: {description: "on-time", lastUpdated: "1 hour ago"}}})**

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54d"),

"departureAirport" : "MUC",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

"intercontinental" : true,

"status" : {

"description" : "on-time",

"lastUpdated" : "1 hour ago"

}

}

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54e"),

"departureAirport" : "LHR",

"arrivalAirport" : "TXL",

"aircraft" : "Airbus A320",

"distance" : 950,

"intercontinental" : false,

"status" : {

"description" : "on-time",

"lastUpdated" : "1 hour ago"

}

}

**Adding another nested document inside status document:**

**db.flightData.updateMany({}, {$set: {status: {description: "on-time", lastUpdated: "1 hour ago", details: {responsible: "Saravana Pandiyan"}}}})**

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54d"),

"departureAirport" : "MUC",

"arrivalAirport" : "SFO",

"aircraft" : "Airbus A380",

"distance" : 12000,

"intercontinental" : true,

"status" : {

"description" : "on-time",

"lastUpdated" : "1 hour ago",

"details" : {

"responsible" : "Saravana Pandiyan"

}

}

}

Adding array to the document:

**db.passengers.updateOne({"name": "Albert Twostone"}, {$set: {hobbies: ["sprots", "cooking"]}})**

Before Adding:

{

"\_id" : ObjectId("5f1be4d2f4ade3fbde32a562"),

"name" : "Albert Twostone",

"age" : 68

}

After Adding:

{

"\_id" : ObjectId("5f1be4d2f4ade3fbde32a562"),

"name" : "Albert Twostone",

"age" : 68,

"hobbies" : [

"sprots",

"cooking"

]

}

* **Accessing Structured Data:**

If you want to access only hobbies from the above link, we can use below query.

**db.passengers.findOne({"name": "Albert Twostone"}).hobbies**

[ "sprots", "cooking" ]

Filtering records based on Array value:

**db.passengers.find({"hobbies": "sprots"}).pretty()**

{

"\_id" : ObjectId("5f1be4d2f4ade3fbde32a562"),

"name" : "Albert Twostone",

"age" : 68,

"hobbies" : [

"sprots",

"cooking"

]

}

**Filtering nested document:**

{

"\_id" : ObjectId("5f1b3414f4ade3fbde32a54e"),

"departureAirport" : "LHR",

"arrivalAirport" : "TXL",

"aircraft" : "Airbus A320",

"distance" : 950,

"intercontinental" : false,

"status" : {

"description" : "on-time",

"lastUpdated" : "1 hour ago",

"details" : {

"responsible" : "Saravana Pandiyan"

}

}

}

**db.flightData.find({"status.description": "on-time"}).pretty()**

**db.flightData.find({"status.details.responsible": "Saravana Pandiyan"}).pretty()**