MongoDB

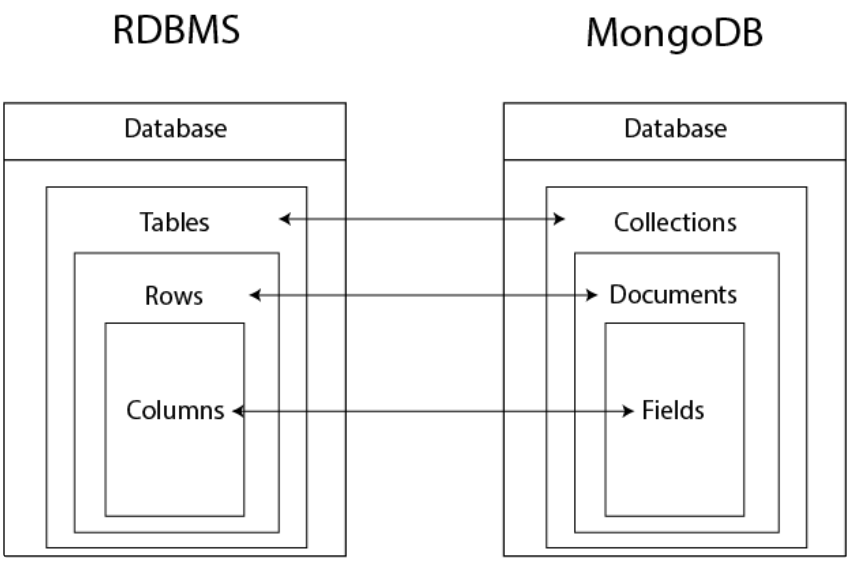
Introduction

MongoDB is a **NoSql database**, which stores the data in groups of **collections** that contain **documents** (also known as **records**).

Each document is a JSON which contains a group of **fields** in a structure of **key-value pairs**.

A document can includes also sub-documents hierarchically.

(MongoDB represents JSON documents in binary-encoded format so we call it BSON behind the scenes).



Relational

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Cars

ID	user_id	model	year
20	1	Bentley	1973
21	1	Rolls Royce	1965

MongoDB

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
            {
                model: "Bentley",
                year: 1973
            },
            {
                model: "Rolls Royce",
                year: 1965
            }
    ]
}
```

@JoeKarlsson1

MongoDB Schema Design

See very recommended slides about this topic here (or the video version here)

MongoDB operations

Connecting to Mongodb

- open mongo docker cli (if you want to work on your local terminal machine, find here how to install and run mongo)

- connect to mongo `mongo -u root -p root --host localhost`

Create database

`use <db name>`

```
> use acme
switched to db acme
```

Get all databases

In mongo shell enter `show dbs`

```
> show dbs
admin          0.000GB
config         0.000GB
local          0.000GB
media-service  0.000GB
```

NOTE: if there isn't any collection in the database it won't be shown in the databases list

Select database

`use <db name>`

NOTE: after you enter this command, you can start working on the database you have just selected. generally commands will start with `db.<command name>`

```
> use media-service
switched to db media-service
```

Delete database

`db.dropDatabase()`

```
> db.dropDatabase()
{ "ok" : 1 }
```

Get current database

`db`

```
> db
media-service
```

Create collection

`db.createCollection(<collection name>)`

```
> db.createCollection('posts')
{ "ok" : 1 }
```

Get all collections

`show collections`

```
> show collections
posts
```

Create document

`db.<collection name>.insert(<object>)`

```
> db.posts.insert({
...   title: 'Post One',
...   body: 'Body of post one',
...   tags: ['news', 'events'],
...   user: {
...     name: 'John Doe',
...     status: 'author'
...   },
...   date: Date()
... })
WriteResult({ "nInserted" : 1 })
```

Create many documents

`db.<collection name>.insertmany(<objects array>)`

```
> db.posts.insertMany([{
...   title: 'Post Two',
...   body: 'Body of post two'
... },
... {
...   title: 'Post Three',
...   body: 'Body of post three'
... },
... {
...   title: 'Post Four',
...   body: 'Body of post four'
... }])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61cb38fb528cfb26bda1b22a"),
                ObjectId("61cb38fb528cfb26bda1b22b"),
                ObjectId("61cb38fb528cfb26bda1b22c")
        ]
}
```

Get all documents

`db.<collection name>.find()`

```
> db.posts.find()
{ "_id" : ObjectId("61cb3298528cfb26bda1b229"), "title" : "Post One", "body" : "Body of post o
ne", "tags" : [ "news", "events" ], "user" : { "name" : "John Doe", "status" : "author" }, "da
te" : "Tue Dec 28 2021 15:51:52 GMT+0000 (UTC)" }
{ "_id" : ObjectId("61cb38fb528cfb26bda1b22a"), "title" : "Post Two", "body" : "Body of post t
wo" }
{ "_id" : ObjectId("61cb38fb528cfb26bda1b22b"), "title" : "Post Three", "body" : "Body of post
 three" }
{ "_id" : ObjectId("61cb38fb528cfb26bda1b22c"), "title" : "Post Four", "body" : "Body of post
four" }
```

you can prettify the response by adding pretty()

`db.<collection name>.find().pretty()`

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post One",
        "body" : "Body of post one",
        "tags" : [
                "news",
                "events"
        ],
        "user" : {
                "name" : "John Doe",
                "status" : "author"
        },
        "date" : "Tue Dec 28 2021 15:51:52 GMT+0000 (UTC)"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four"
}
```

Get documents by condition

`db.<collection name>.find(<condition object>).pretty()`

```
> db.posts.find({title: 'Post One'}).pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post One",
        "body" : "Body of post one",
        "tags" : [
                "news",
                "events"
        ],
        "user" : {
                "name" : "John Doe",
                "status" : "author"
        },
        "date" : "Tue Dec 28 2021 15:51:52 GMT+0000 (UTC)"
}
```

Get documents and sort

for ascending sort: `db.<collection name>.find().sort({<field name>: 1}).pretty()`

for descending sort `db.<collection name>.find().sort({<field name>: -1}).pretty()`

```
> db.posts.find().sort({title: 1}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four"
}
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post One",
        "body" : "Body of post one",
        "tags" : [
                "news",
                "events"
        ],
        "user" : {
                "name" : "John Doe",
                "status" : "author"
        },
        "date" : "Tue Dec 28 2021 15:51:52 GMT+0000 (UTC)"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
```

Get first document

`db.<collection name>.findOne(<condition object>)`

```
> db.posts.findOne({tags: null})
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
```

Count documents

`db.<collection name>.find().count()`

```
> db.posts.find().count()
4
```

Get limited documents

`db.<collection name>.find().limit(<limit number>).pretty()`

```
> db.posts.find().limit(2).pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post One",
        "body" : "Body of post one",
        "tags" : [
                "news",
                "events"
        ],
        "user" : {
                "name" : "John Doe",
                "status" : "author"
        },
        "date" : "Tue Dec 28 2021 15:51:52 GMT+0000 (UTC)"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
```

Update document

`db.<collection name>.update(<condition object>,<new document object>,<upsert:<boolean>)`

```
> db.posts.update({title: 'Post One'},{title: 'Post #1', body: 'body #1'})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post #1",
        "body" : "body #1"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four"
}
```

NOTE: if the condition is true for more than one document, it will update only the first one. for updating many documents look at “Update many documents” section

upsert

upsert: true - creates a new document in case of condition is not true for any existing document

```
}
> db.posts.update({title: 'Not existed title'},{title: 'Post #6', body: 'body #6'}, {upsert: true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("61cb4b4490dc3a288039a919")
})
> db.posts.find().pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post #1",
        "body" : "body #1"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three"
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four"
}
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "body #5"
}
{
        "_id" : ObjectId("61cb4b4490dc3a288039a919"),
        "title" : "Post #6",
        "body" : "body #6"
}
```

set parameter

Without using set parameter, the new object will replace the current object. for updating specific fields in the document without losing data use parameter `$set`

```
> db.posts.update({title: 'Post #5'},{$set: {body: 'New Body Five', like: 4}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.posts.find({title: 'Post #5'}).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "like" : 4
}
```

inc parameter

We can update a numeric field by increment parameter `$inc`

```
> db.posts.find({title: 'Post #5'}).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "like" : 4
}
> db.posts.update({title: 'Post #5'},{$inc: {like: 2}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.posts.find({title: 'Post #5'}).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "like" : 6
}
```

rename parameter

We can rename a field name by parameter

```
> db.posts.update({title: 'Post #5'},{$rename: {like: 'likes'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.posts.find({title: 'Post #5'}).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 6
}
```

Update many documents

`db.<collection name>.updateMany(<condition object>,<new document object>)`

for updating all, leave the condition object empty

```
> db.posts.updateMany({},{$set: {likes: 10}})
{ "acknowledged" : true, "matchedCount" : 7, "modifiedCount" : 7 }
> db.posts.find().pretty()
{
        "_id" : ObjectId("61cb3298528cfb26bda1b229"),
        "title" : "Post #1",
        "body" : "body #1",
        "likes" : 10
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 10
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 10
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 10
}
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
```

Remove document

`db.<collection name>.remove(<condition object>)`

```
> db.posts.remove({title: 'Post #6'})
WriteResult({ "nRemoved" : 2 })
```

Sub documents

We can add sub-documents to a document by simply update a document to include an array of the sub-documents

```
> db.posts.update({title: 'Post #5'},{$set: {comments: [
...   {
...     user: 'Yossi',
...     body: 'Yossi\'s comments'
...   },
...   {
...     user: 'Ivgeny',
...     body: 'Ivgeny\'s comments'
...   }
... ]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Get document by sub-document

1. `db.<collection name>.find(<condition by one of the sub-document>)`

```
> db.posts.find({
...   comments:{user:'Yossi', body:'Yossi\'s comments'}
... }).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 10,
        "comments" : [
                {
                        "user" : "Yossi",
                        "body" : "Yossi's comments"
                },
                {
                        "user" : "Ivgeny",
                        "body" : "Ivgeny's comments"
                }
        ]
}
```

2. `db.<collection name>.find(<condition by $elemMatch parameter>`

```
> db.posts.find({   comments:{$elemMatch: {user:'Yossi'}} }).pretty()
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 10,
        "comments" : [
                {
                        "user" : "Yossi",
                        "body" : "Yossi's comments"
                },
                {
                        "user" : "Ivgeny",
                        "body" : "Ivgeny's comments"
                }
        ]
}
```

Indexes

In order to get a document by field text, we first need to create an index.

```
> db.posts.find({$text: {$search: "Five"}})
Error: error: {
        "ok" : 0,
        "errmsg" : "text index required for $text query",
        "code" : 27,
        "codeName" : "IndexNotFound"
```

In the example above, we can't get "Post Five" document, since we didn't create an index for "title" field which contains "Five".
In order to manage doing this we should create an index for "title" field

```
> db.posts.createIndex({title: 'text'})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
```

Before we run this command, we had only the id index ("numIndexesBefore" : 1)
now we can search by text

```
> db.posts.find({$text: {$search: 'Four Two'}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 10
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 10
}
```

If we want to get a document by the exact text, we should add a quotes

```
> db.posts.find({$text: {$search: '"Post F"'}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 10
}
```

Comparison Query Operations

We have some parameters for getting documents by comparison query

Assume this is the collection:

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 2
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 3
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 4
}
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 5,
        "comments" : [
                {
                        "user" : "Yossi",
                        "body" : "Yossi's comments"
                },
                {
                        "user" : "Ivgeny",
                        "body" : "Ivgeny's comments"
                }
        ]
}
```

gt parameter

```
> db.posts.find({likes: {$gt: 2}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 3
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 4
}
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 5,
        "comments" : [
                {
                        "user" : "Yossi",
                        "body" : "Yossi's comments"
                },
                {
                        "user" : "Ivgeny",
                        "body" : "Ivgeny's comments"
                }
        ]
}
```

gte parameter

```
> db.posts.find({likes: {$gte: 2}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 2
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 3
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 4
}
{
        "_id" : ObjectId("61cb4aa690dc3a288039a8f0"),
        "title" : "Post #5",
        "body" : "New Body Five",
        "likes" : 5,
        "comments" : [
                {
                        "user" : "Yossi",
                        "body" : "Yossi's comments"
                },
                {
                        "user" : "Ivgeny",
                        "body" : "Ivgeny's comments"
                }
        ]
}
```

lt parameter

```
> db.posts.find({likes: {$lt: 4}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 2
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 3
}
```

lte parameter

```
> db.posts.find({likes: {$lte: 4}}).pretty()
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22a"),
        "title" : "Post Two",
        "body" : "Body of post two",
        "likes" : 2
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22b"),
        "title" : "Post Three",
        "body" : "Body of post three",
        "likes" : 3
}
{
        "_id" : ObjectId("61cb38fb528cfb26bda1b22c"),
        "title" : "Post Four",
        "body" : "Body of post four",
        "likes" : 4
}
```

Functions

Print function

You can print to the mongo shell by `print(<expression>)`

```
> print(!true)
false
```

```
> print('I love mongo')
I love mongo
```

forEach function

`db.<collection name>.find().forEach(<function>)`

```
> db.posts.find().forEach(function(record) {print('Blog post: ' + record.title)})
Blog post: Post One
Blog post: Post Two
Blog post: Post Three
Blog post: Post Four
```