MongoDB - Aula 01 – Resumo

Autores: Erni Augusto Fonseca Souza (Sistema_On)
Diego Araújo (DiihFilipe)

NOMECLATURAS

RELACIONAL	-	MONGODB
Base de Dados	-	Coleção
Registro	-	Documento
Índice / Index	-	Índice / Index
Join	-	Documento Embarcado
Foreing Key	-	Referência

Import e Export

```
mongoimport --db be-mean-moto --collection motos --drop --file /home/sistemaon/Documents/motos.json
/**
mongoimport -> comando do MongoDB para importar.
--db be-mean-moto -> comando para selecionar o banco, logo em seguida com o nome do banco.
--collection motos -> comando para escrever o nome da coleção (collection), logo em seguida o nome da coleção.
--drop -> comando para dropar os dados ao banco.
--file /home/sistemaon/Documents/motos.json-> comando para selecionar o arquivo (file), logo em seguida o nome do arquivo com a extensão (.json), nesse caso estou passando o caminho (dir) completo onde o arquivo se encontra e o nome do arquivo com a extensão.
 **/

 mongoexport --db be-mean-moto --collection motos --out /home/moto_exportada_banco.json
/**
mongoexport -> comando do MongoDB para exportar.
--db be-mean-moto -> comando para selecionar o banco, logo em seguida com o nome do banco.
--collection motos -> comando para escrever o nome da coleção (collection), logo em seguida o nome da coleção.
--out /home/moto_exportada_banco.json -> especifica o arquivo para escrever (write) ao exportar (export), logo em seguida o nome do arquivo com a extensão (.json), nesse caso estou passando o caminho (dir) completo, na qual onde o MongoDB vai vai criar e escrever (write) ao exportar (export) e o nome do arquivo com a extensão.
 **/
```

//OBS: Os comandos mongoimport e mongoexport **DEVEM SEMPRE SER FEITOS NO SEU TERMINAL/CMD/PROMPT** nunca no console do mongo.

Importando as motos

```
sistemaon@sistemaon-VirtualBox:~$ mongoimport --db be-mean-moto --collection motos --drop --file
/home/sistemaon/Documents/motos.json
2015-11-19T23:07:41.559-0500connected to: localhost
2015-11-19T23:07:41.559-0500dropping: be-mean-moto.motos
2015-11-19T23:07:41.566-0500imported 13 documents
```

Entrando no banco be-mean-moto
sistemaon@sistemaon-VirtualBox:~$ mongo be-mean-moto
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> //pronto pra sentar query! rsrsrsrs...

//OBS: Para mudar de banco basta rodar o comando use nome_do_banco. EX: use be-mean-moto

Contando as motos
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.count()
13

root@sistemaon-VirtualBox:~# mongoexport --db be-mean-moto --collection motos --out /home/moto_exportada_banco.json
2015-11-20T14:37:56.838-0500connected to: localhost
2015-11-20T14:37:56.884-0500exported 13 records

mongo be-mean-moto
/**
mongo be-mean-moto-> comando do MongoDB para entrar, iniciar o banco (mongo), logo em seguida com o nome do banco.
**/

db.motos.count()
/**
db. -> pré-chamada para utilização dos comandos do mongo, como find(), count(), drop(), insert(), etc...
motos. -> seleciona a coleção (collection) para gerenciamento.
count() -> chamada do comando na qual conta (count()) a quantidade de documentos que tenho em minha coleção (collection).
**/

 # MongoDB - Aula 02 – Resumo

Autores: Erni Augusto Fonseca Souza (Sistema_On)
Diego Araújo (DiihFilipe)

/**
Método insert():
Insere um documento ou documentos em uma coleção (collection).
**/

Crie uma database chamada be-mean-motos
root@sistemaon-VirtualBox:~# mongo be-mean-moto //comando do MongoDB para entrar, iniciar o banco (mongo), logo em seguida com o nome do banco.

Liste quais databases você possui nesse servidor
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> show dbs //comando para mostrar (show) os bancos (dbs) que possui.

be-mean-test	0.078GB
be-mean-moto	0.078GB
moto-test	0.078GB
local	0.078GB
be-mean-mega	0.078GB
be-mean-pokemon	0.078GB
be-mean-restaurante	0.078GB

Liste quais coleções você possui nessa database
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> show collections // comando para mostrar (show) as coleções (collections) que possui.

motos	0.003MB / 0.008MB
system.indexes	0.000MB / 0.008MB

Troca o banco que deseja gerenciar
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> use be-mean-moto /** comando para usar, utilizar, trocar (use) o banco que queremos gerenciar, neste caso o (be-mean-moto).
**/
switched to db be-mean-moto

Insira 1 moto como exemplo utilizando o mesmo padrão de campos, utilizado:
{fabricante: string ,nome: string, motor: [array], cilindradas: int ou float, modelo: string, cor: [array], importada: boolean, nacional: boolean, topSpeed: int ou float}
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var moto = {fabricante: "Honda", nome: "Crf-450x", motor: ["4 tempos", "1 cilindros"], cilindradas: 449.5, modelo: "terra", cor: ["vermelha e preta", "vermelha e branca"], importada: true, nacional: false, topSpeed: 140}
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> moto //mostrando no console o que foi criado com a variável (var) chamado (moto)

```
{
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "motor": [
    "4 tempos",
    "1 cilindros"
  ],
  "cilindradas": 449.5,
  "modelo": "terra",
  "cor": [
    "vermelha e preta",
    "vermelha e branca"
```

```
  ],
  "importada": true,
  "nacional": false,
  "topSpeed": 140
}
```

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.insert(moto) /** método de inserção (insert) de documento, variável (var) chamada (moto) setada acima é passada como pârametro no método (insert(moto)) para inserir o novo documento na coleção (collection) de motos.
**/

```
Inserted 1 record(s) in 62ms
WriteResult({
  "nInserted": 1
})
```

Busque a moto a sua escolha, pelo nome, e armazene-o em uma variável qualquer.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var moto = {nome: /crf/i} /** variável (var) com nome (moto) na qual recebe como valor (/crf/i)
em seu nome (nome:)
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(moto) /** encontrar (find()) algum documento em minha coleção, neste caso quero encontrar (find()) um documento que possui valor (/crf/i) em seu nome (nome:) setado na variável (var) chamada (moto), que passa como parâmetro dentro do find (find(query))
**/

```
{
  "_id": ObjectId("564f804ee4919d0a1085a406"),
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "motor": [
    "4 tempos",
    "1 cilindros"
  ],
  "cilindradas": 449.5,
  "modelo": "terra",
  "cor": [
    "vermelha e preta",
    "vermelha e branca"
  ],
  "importada": true,
  "nacional": false,
  "topSpeed": 140
}
```

/**
O find() retorna um Cursor.
O que é Cursor? Um apontador, indicação para o conjunto de resultados de uma consulta (query).
Por isso com o find() não conseguimos acessar diretamente nosso objeto pois ele é retornado na forma de Cursor.
O método primário para operações de leitura é o método db.collection.find()
**/

Modifique seu `motor` e salve-o.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var moto = {nome: /crf/i} /** variável (var) com nome (moto) na qual recebe como valor (/crf/i) em seu nome (nome:)
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var m = db.motos.find(moto) /** variável (var) com nome (m) na qual recebe como valor o método do MongoDB de encontrar (find()) dentro da minha coleção (collection) de (motos) que setei na variável (var) chamada (m), e que estou passando como parâmetro dentro do find (find(moto)) **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m /** neste caso estou fazendo uma chamada do método do MongoDB que foi setado especificamente como (db.motos.find(moto)) pela variável com nome de (m) que foi setada, e mostrando no console o que foi encontrado (find()), que é a mesma coisa de chamar diretamente db.motos.find(moto) sem setar em nenhuma variável.
**/


```
{
  "_id": ObjectId("564f804ee4919d0a1085a406"),
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "motor": [
    "4 tempos",
    "1 cilindros"
  ],
  "cilindradas": 449.5,
  "modelo": "terra",
  "cor": [
    "vermelha e preta",
    "vermelha e branca"
  ],
  "importada": true,
  "nacional": false,
  "topSpeed": 140
}
```


sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m.motor /** como a variável (var) (m) está setado com o método do mongoDB (find()) dentro da coleção (collection) (moto), o find() ao chamar o (motor) retornou NADA do motor (m.motor). Ele retornou NADA dos valores do motor pois o findOne() retorna retorna um Cursor.
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var m = db.motos.findOne(moto) /** variável (var) com nome (m) na qual recebe como valor o método do MongoDB de encontrar (findOne()) dentro da minha coleção (collection) de (motos) que setei na variável (var) chamada (m), e que estou passando como parâmetro dentro do findOne (findOne(moto))
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m /** neste caso estou fazendo uma chamada do método do MongoDB que foi setado especificamente como (db.motos.findOne(moto)) pela variável com nome de (m) que foi setada, e mostrando no console o que foi encontrado (findOne()), que é a mesma coisa de chamar diretamente db.motos.find(moto) sem setar em nenhuma variável.
**/


```
{
  "_id": ObjectId("564f804ee4919d0a1085a406"),
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "motor": [
    "4 tempos",
    "1 cilindros"
  ],
  "cilindradas": 449.5,
  "modelo": "terra",
  "cor": [
    "vermelha e preta",
```

```
    "vermelha e branca"
  ],
  "importada": true,
  "nacional": false,
  "topSpeed": 140
}

/**
findOne() retorna um objeto comum (único documento).
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m.motor /** como a variável (var) (m) está setado com o método do mongoDB (findOne()) dentro da coleção (collection) (moto), o findOne() ao chamar o (motor) retornou os valores(["4 tempos", "1 cilindros"]) do motor (m.motor). Ele retornou os valores pois o findOne() retorna um objeto comum.
**/
[
  "4 tempos",
  "1 cilindros"
]

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m.motor = ["2 tempos", "1 cilindros"] /** Aqui está sendo setada na variável (m) o novo valor do motor que é o objeto comum.
**/
[
  "2 tempos",
  "1 cilindros"
]

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> m /** neste caso estou fazendo uma chamada do método do MongoDB (db.motos.findOne(moto)) pela variável com nome de (m) que foi setada, e mostrando no console o que foi encontrado (findOne()), que é a mesma coisa de chamar diretamente db.motos.find(moto) sem setar em nenhuma variável. E que está com o novo valor do motor.
**/
{
  "_id": ObjectId("564f804ee4919d0a1085a406"),
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "motor": [
    "2 tempos",
    "1 cilindros"
  ],
  "cilindradas": 449.5,
  "modelo": "terra",
  "cor": [
    "vermelha e preta",
    "vermelha e branca"
  ],
  "importada": true,
  "nacional": false,
  "topSpeed": 140
}
```

/**
Método save():
	Atualiza um documento existente ou insere um novo documento, dependendo do parâmetro do documento.
**/
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.save(m) /** chamada do método (save) MongoDB dentro da minha coleção (collection) passando como parâmetro a variável (var) (m), para salvar (save) o novo valor setado (["2 tempos", "1 cilindros"]) do motor.
**/
Updated 1 existing record(s) in 64ms
WriteResult({
  "nMatched": 1,
  "nUpserted": 0,
  "nModified": 1
})

 # MongoDB - Aula 03 – Resumo

Autores: Erni Augusto Fonseca Souza (Sistema_On)
Diego Araújo (DiihFilipe)

/** Limite de campos (field) para retornar de uma pesquisa, query.
<field>: <1 or true> Especifica a INCLUSÃO do campo (field).
<field>: <0 or false> Especifica a OMISSÃO do campo (field).
Por padrão o campo (field) _id é sempre e o único incluso nos resultados, caso queira omiti-lo, deve seta-lo como; _id: 0.
Por padrão o restante dos campos (field) é sempre omitido nos resultados, caso queira incluí-los, deve seta-los como; nomeDoCampo: 1. EX: fabricante: 1.
**/

1. Liste todas as motos com topSpeed MENOR que 211
connecting to: be-mean-pokemon
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {topSpeed: {$lt: 211}} /** variável (var) chamada (query) recebe topSpeed menor que 211 ($lt: 211) - Less Than ($lt).
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, fabricante: 1, nome: 1, modelo: 1, topSpeed: 1} /** INCLUI no campo (field) fabricante, nome, modelo e topSpeed, OMITE no campo (field) _id.
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field),na qual retorna o que foi setado na query acima e no campo (field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false).
**/

{
 "fabricante": "Yamaha",
 "nome": "Xj-6",
 "modelo": "naked",
 "topSpeed": 192
}
{
 "fabricante": "Yamaha",
 "nome": "Mt-09",
 "modelo": "naked",
 "topSpeed": 208
}
{
 "fabricante": "Honda",
 "nome": "Hornet",
 "modelo": "naked",
 "topSpeed": 209
}
{
 "fabricante": "Ducati",
 "nome": "Monster-796",
 "modelo": "naked",
 "topSpeed": 204
}
{

```
  "fabricante": "Harley-Davidson",
  "nome": "Xr-1200x",
  "modelo": "chopper",
  "topSpeed": 201
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Sportster-883",
  "modelo": "chopper",
  "topSpeed": 191
}
{
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "modelo": "terra",
  "topSpeed": 140
}
Fetched 7 record(s) in 52ms // retornou 7 records.
```

2. Liste todas as motos com topSpeed MAIOR IGUAL que 211

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {topSpeed: {$gte: 211}} /** retorna as motos com topSpeed maior igual que 211 ($gte: 211) - Greater Than Iqual ($gte). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, fabricante: 1, nome: 1, modelo: 1, topSpeed: 1} /** INCLUI no campo (field) fabricante, nome, modelo e topSpeed, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na query acima e no campo (field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false). **/

```
{
  "fabricante": "Yamaha",
  "nome": "R1",
  "modelo": "sport",
  "topSpeed": 238
}
{
  "fabricante": "Honda",
  "nome": "Fireblade",
  "modelo": "sport",
  "topSpeed": 223
}
{
  "fabricante": "Kawazaki",
  "nome": "Z800",
  "modelo": "naked",
  "topSpeed": 217
}
{
  "fabricante": "Suzuki",
  "nome": "Gsx-r1000",
  "modelo": "sport",
  "topSpeed": 235
}
{
  "fabricante": "Ducati",
  "nome": "1199",
  "modelo": "sport",
  "topSpeed": 275
}
```

```
{
  "fabricante": "Triumph",
  "nome": "Daytona-675r",
  "modelo": "sport",
  "topSpeed": 222
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Fat-boy",
  "modelo": "chopper",
  "topSpeed": 211
}
Fetched 7 record(s) in 11ms // retornou 7 records.
```

##3. Liste todas as motos com cilindradas menor igual que 700 E modelo sport.

```
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {$and: [{cilindradas: {$lte: 700}}, {modelo:
"sport"}]} /** retorna as motos com cilindradas menor igual ($lte) que 700 e ($and) modelo sport. */

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, fabricante: 1, nome: 1, modelo: 1,
topSpeed: 1} /** INCLUI no campo (field) fabricante, nome, modelo e topSpeed, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção
de motos passando como parâmetros (query, field), na qual retorna o que foi setado na query acima e no campo
(field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false). **/
{
  "fabricante": "Triumph",
  "nome": "Daytona-675r",
  "modelo": "sport",
  "topSpeed": 222
}
Fetched 1 record(s) in 2ms // retornou 1 records.
```

4. Liste todas motos com o modelo chopper OU com motor com 2 cilindros.

```
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {$or: [{modelo: "chopper"}, {motor: "2
cilindros"}]} /** retorna as motos com modelo chopper ou ($or) **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, fabricante: 1, nome: 1, modelo: 1, motor:
1, topSpeed: 1} /** INCLUI no campo (field) fabricante, nome, modelo, motor e topSpeed, OMITE no campo (field)
_id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção
de motos passando como parâmetros (query, field), na qual retorna o que foi setado na query acima e no campo
(field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false). **/
{
  "fabricante": "Ducati",
  "nome": "Monster-796",
  "motor": [
    "4 tempos",
    "2 cilindros"
  ],
  "modelo": "naked",
  "topSpeed": 204
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Fat-boy",
  "motor": [
```

```
    "4 tempos",
    "2 cilindros"
  ],
  "modelo": "chopper",
  "topSpeed": 211
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Xr-1200x",
  "motor": [
    "4 tempos",
    "2 cilindros"
  ],
  "modelo": "chopper",
  "topSpeed": 201
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Sportster-883",
  "motor": [
    "4 tempos",
    "2 cilindros"
  ],
  "modelo": "chopper",
  "topSpeed": 191
}
Fetched 4 record(s) in 5ms // retornou 4 records

## 5. Liste todas as motos com motor 3 cilindros E com cor preta
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {$and: [{motor: "3 cilindros"}, {cor: "preta"}]}
/** retorna motos com motor de 3 cilindrose ($and) com cor preta.**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, fabricante: 1, nome: 1, modelo: 1, motor:
1, cor: 1} /** INCLUI no campo (field) fabricante, nome, modelo, motor e cor, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção
de motos passando como parâmetros (query, field), na qual retorna o que foi setado na query acima e no campo
(field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false). **/
{
  "fabricante": "Yamaha",
  "nome": "Mt-09",
  "motor": [
    "4 tempos",
    "3 cilindros"
  ],
  "modelo": "naked",
  "cor": [
    "preta",
    "roxa",
    "cinza"
  ]
}
{
  "fabricante": "Kawazaki",
  "nome": "Z800",
  "motor": [
    "4 tempos",
```

```
    "3 cilindros"
  ],
  "modelo": "naked",
  "cor": [
    "preta",
    "verde",
    "preta e verde"
  ]
}
{
  "fabricante": "Triumph",
  "nome": "Daytona-675r",
  "motor": [
    "4 tempos",
    "3 cilindros"
  ],
  "modelo": "sport",
  "cor": [
    "preta",
    "azul",
    "branca"
  ]
}
Fetched 3 record(s) in 5ms // retornou 3 records
```

6. Liste todas as motos com motor 3 cilindros OU modelo naked E cilindradas maior que 600 OU topSpeed menor igual 200

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {$and: [{ $or: [{ motor: "3 cilindros"}, { modelo: "naked"}]}, {$or: [{ cilindradas: {$gt: 600}}, {topSpeed: {$lte: 200}}]}]} /** variável (var) chamada (query) recebendo motor 3 cilindros OU ($or) modelo naked E ($and) cilindradas maior que ($gt) 600 OU ($or) topSpeed menor igual ($lte) 200 **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {_id: 0, nome: 1, modelo: 1, cilindradas: 1, topSpeed: 1, motor: 1} /** INCLUI no campo (field) nome, modelo, cilindradas, topSpeed e motor, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na query acima e no campo (field) INCLUI o que foi setado como 1 (true) e OMITE o que foi setando como 0 (false). **/

```
{
  "nome": "Xj-6",
  "motor": [
    "4 tempos",
    "4 cilindros"
  ],
  "cilindradas": 600,
  "modelo": "naked",
  "topSpeed": 192
}
{
  "nome": "Z800",
  "motor": [
    "4 tempos",
    "3 cilindros"
  ],
  "cilindradas": 806,
  "modelo": "naked",
```

```
  "topSpeed": 217
}
{
  "nome": "Monster-796",
  "motor": [
    "4 tempos",
    "2 cilindros"
  ],
  "cilindradas": 796,
  "modelo": "naked",
  "topSpeed": 204
}
{
  "nome": "Daytona-675r",
  "motor": [
    "4 tempos",
    "3 cilindros"
  ],
  "cilindradas": 675,
  "modelo": "sport",
  "topSpeed": 222
}
Fetched 4 record(s) in 3ms // retorna 4 records
```

 # MongoDB - Aula 04 – Resumo

Autores: Erni Augusto Fonseca Souza (Sistema_On)
Diego Araújo (DiihFilipe)

/**
Operador $pushAll:
O operador $pushAll acrescenta os valores especificados em um array.
Se você especificar um valor único ao operador $pushAll, ele comportará como o operador $push.
Forma do operador $pushAll: { $pushAll: { <field>: [<value1>, <value2>, ...] } }
Exemplo da Forma do operador $pushAll: {$pushAll: {cor: ['roxa', 'laranja']}}

Operador $push:
O operador $push acrescenta um valor especificado para um array.
Forma do operador $push: { $push: { <field1>: <value1>, ... } }
Exemplo da Forma do operador $push: {$push: {motor: 'gasolina'}}

Operador $inc:
O operador $inc INCREMENTA e/ou DECREMENTA um campo pelo valor especificado.
Aceitando valores POSITIVOS e NEGATIVOS
Forma do operador $inc: { $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }
Exemplo da Forma do operador $inc: { $inc: { cilindradas: -25 , topSpeed: 35} }

Operador $setOnInsert:
O operador $setOnInsert atribui os valores especificados para os campos no documento.
Forma do operador $setOnInsert:

```
db.collection.update (
   <query>,
   { $setOnInsert: { <field1>: <value1>, ... } },
   { upsert: true }
)
```

Exemplo da Forma do operador $setOnInsert:

```
var query = {nome: /H2r/i}
var mod = {$setOnInsert:
 {
   fabricante: 'Sem Informações',
   nome: 'H2r',
   motor:[null],
   cilindradas: null,
   modelo: null,
   cor: [null],
   importada: null,
   nacional: null,
   topSpeed: null
 }
}
var opts = {upsert: true}
db.motos.update(query, mod, opts)
```

Operador $in:
O operador $in seleciona os documentos onde o valor de um campo é igual a qualquer valor especificado no array.
Forma do operador $in: { field: { $in: [<value1>, <value2>, ... <valueN>] } }
Exemplo da Forma do operador $in: {cor: {$in: [/cinza/i, /azul/i] }}

Operador $ne:
	O operador $ne seleciona os documentos onde o valor do campo não é igual (!=) para o valor especificado.
	Isto inclui documentos que não contêm o campo.
	Forma do operador $ne: { field: { $ne: value } }
	Exemplo da Forma do operador $ne: {modelo: {$ne: 'sport' , 'naked'}}

Método update():
	Modifica um documento existente ou documentos em uma coleção.
	O método pode modificar os campos específicos de um documento ou
	documentos ou substituir um documento existente, dependendo do parâmetro do update().
	Por padrão, o método update() atualiza um único documento.

Método remove():
	Remove documentos da coleção (collection).
	O método remove() pode receber uma consulta (query) e opcionalmente um boleano (boolean).
	OBS: Neste resumo/exemplo, apenas está sendo utilizado a consulta (query) no método remove().

Parâmetro upsert:
	Opcional. Se setado como true, cria um novo documento quando nenhum documento corresponde com os critérios de consulta (query). O valor padrão setado é false, na qual não insere um novo documento quando nenhuma combinação for encontrada.

Parâmetro multi:
	Opcional. Se setado como true, atualiza vários documentos que atendem aos critérios da consulta (query).
	Se setado como false, atualiza um documento. O seu valor padrão é false.
	A operação de atualização multi (multi update) pode unir-se com outras operações, tanto operações de leitura (read) e/ou escrita (write).

Consulta vazia ({}):
	Um documento de consulta vazia ({}) automaticamente seleciona todos os documentos da coleção (collection).
**/

// Add 2 cores ao mesmo tempo para: a Xj-6.
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {nome: /xj/i} // variável (var) chamada query, recebe como valor (/xj/i) em seu (nome:).

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var mod = {$pushAll: {cor: ['roxa', 'laranja']}} /** variável (var) chamada mod, recebe como valor(es) ('roxa') e ('laranja') em seu campo (cor:), na qual são valores especificados para acrescentar no array de (cor:) pelo operador ($pushAll). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.update(query, mod) /** método de atualização (update()) em que recebe como parâmetros a variável (var) chamada (query) como consulta e a variável (var) chamada (mod) como modificador, na qual o método (update()) irá modificar os campos do documento conforme o que foi setado na variável (var) (query) e variável (mod). **/
Updated 1 existing record(s) in 4ms
WriteResult({ 		// ResultadoDeEscrita
  "nMatched": 1,
  "nUpserted": 0,
  "nModified": 1
})

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query) /** método de busca (find()) que é um apontador para o conjunto de resultados dessa consulta, em que recebe como pârametro a variável (var) chamada (query), na qual irá buscar (find()) o que foi setado na variável (var) chamada (query). **/


```
{
  "_id": ObjectId("564e9c8db6ef4196b7cd5189"),
  "fabricante": "Yamaha",
  "nome": "Xj-6",
  "motor": [
    "4 tempos",
    "4 cilindros"
  ],
  "cilindradas": 600,
  "modelo": "naked",
  "cor": [
    "preta",
    "azul",
    "branca",
    "roxa",
    "laranja",
    "roxa",
    "laranja"
  ],
  "importada": false,
  "nacional": true,
  "topSpeed": 192
}
```


Fetched 1 record(s) in 3ms // retornou 1 record

// $inc! Nova motoca chegando no mercado! Mais poderosa e mais econômica! decrementa 25 cilindradas e incrementa 35 topSpeed para a Xj-6.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {nome: /xj/i} // variável (var) chamada (query), recebe como valor (/xj/i) em seu (nome:).

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var mod = { $inc: { cilindradas: -25 , topSpeed: 35} } /** variável (var) chamada (mod), em que recebe como parâmetro o operador de incremento/decremento ($inc), na qual especifica um valor negativo de DECREMENTO (-25) no campo (cilindradas:) e um valor positivo de INCREMENTO (35) no campo (topSpeed:). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.update(query, mod) /** atualiza (update()) na coleção de motos passando como parâmetros (query, mod). Na qual primeiramente pesquisa e busca o documento da variável (var) chamada (query) que recebe como valor (/xj/i) em seu campo (nome:), e logo após, atualiza/modifica o documento especificando os campos e seus determinados valores na variável (var) chamada (mod) com o operador ($inc) que recebe como valor de decremento (-25) no campo (cilindradas:) e um valor de incremento (35) no campo (topSpeed:). **/

Updated 1 existing record(s) in 92ms // atualizado 1 record existente.


```
WriteResult({              // ResultadoDeEscrita
  "nMatched": 1,
  "nUpserted": 0,
  "nModified": 1
})
```


// Busque a moto Xj-6 e veja se sua cilindradas e seu topSpeed foram atualizados (incrementado e decrementado).

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {fabricante: 1, nome: 1, modelo: 1, cilindradas: 1, topSpeed: 1, _id: 0} /** INCLUI no campo (field) fabricante, nome, modelo, cilindradas e topSpeed, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na variável (var) chamada (query) acima, e no campo (field) INCLUI os campos que foi setado como 1 (true) e
OMITE os campos que foi setando como 0 (false). **/

```
{
  "fabricante": "Yamaha",
  "nome": "Xj-6",
  "cilindradas": 575,
  "modelo": "naked",
  "topSpeed": 227
}
```

Fetched 1 record(s) in 2ms // encontrou 1 record.

//adiciona - add motor: ['gasolina'], especificar tipo de combustível no motor, adicionar 'gasolina' no motor de todas as motos.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {} /** variável (var) chamada (query), recebe nada, ou seja, consulta vazia ({}), na qual seleciona todos os documentos da coleção (collection). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var mod = {$push: {motor: 'gasolina'}} /** variável (var) chamada mod, recebe como valor ('gasolina') em seu campo (motor:), na qual é valor especificado para acrescentar no array de (motor:) pelo operador ($push). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var opts = {multi: true} /** variável (var) chamada (opts), recebe como valor (true) em seu PARÂMETRO (multi), na qual, atualiza todos os documentos que atendem aos critérios da consulta (query). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.update(query, mod, opts) /** atualiza (update()) na coleção de motos passando como parâmetros (query, mod, opts). Na qual primeiramente pesquisa e busca o documento da variável (var) chamada (query) que recebe nada, ou seja, consulta vazia ({}), selecionando automaticamente todos os documentos da collection e, logo após, atualiza/modifica o documento especificando o campo e seu determinado valor na variável (var) chamada (mod) com o operador ($push) que recebe como valor ('gasolina') em seu campo array (motor:), e que atualiza todos os documentos que atendem aos critérios da consulta (query), na qual foi setada na variável (var) chamada (opts) recebendo como valor (true) em seu PARÂMETRO (multi). **/

Updated 14 existing record(s) in 76ms // atualizado 14 records existentes

```
WriteResult({              // ResultadoDeEscrita
  "nMatched": 14,
  "nUpserted": 0,
  "nModified": 14
})
```

```
"motor": [
   "4 tempos",
   "4 cilindros",
   "gasolina"              // agora temos o valor 'gasolina' no nosso campo array (motor: []).
 ]
```

// add nova moto com informacoes null

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {nome: /h2r/i} // variável (var) chamada (query), recebe como valor (/h2r/i) em seu (nome:).

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var mod = {$setOnInsert:

```
                {
                        fabricante: 'Sem Informações',
                        nome: 'H2r',
                        motor:[null],
                        cilindradas: null,
```

```
                    modelo: null,
                    cor: [null],
                    importada: null,
                    nacional: null,
                    topSpeed: null
                }
        }
        // variável (var) chamada (mod), na qual ATRIBUI os valores especificados:
        // ('Sem Informações', 'H2r', 'null')  para os campos (nome:, motor:[], ...),
        // pelo operador ($setOnInsert).
```

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var opts = {upsert: true} /** variável (var) chamada (opts), recebe como valor (true) em seu PARÂMETRO (upsert), na qual, cria um novo documento quando nenhum documento corresponde com os critérios de consulta (query). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.update(query, mod, opts) /** atualiza (update()) na coleção de motos passando como parâmetros (query, mod, opts). Na qual primeiramente pesquisa e busca o documento da variável (var) chamada (query) que recebe como valor (/h2r/i) em seu campo (nome:), logo após, atribuindo os valores especificados ('Sem informações', 'H2r', 'null') para os campos (nome:, motor:[], ...) na variável (var) chamada (mod) com o operador ($setOnInsert), e que cria um novo documento quando nenhum documento corresponde com os critérios da consulta (query), na qual foi setada na variável (var) chamada (opts) recebendo como valor (true) em seu PARÂMETRO (upsert). **/

```
Updated 1 new record(s) in 35ms
WriteResult({        // ResultadoDeEscrita
  "nMatched": 0,
  "nUpserted": 1,
  "nModified": 0,
  "_id": ObjectId("56552215e205d0fffba8f328")
})
```

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query) /** encontra (find()) na coleção de motos passando como parâmetros (query), na qual retorna o que foi setado na variável (var) chamada (query) acima, valor (/h2r/i) no campo (nome:). **/

```
{
  "_id": ObjectId("56552215e205d0fffba8f328"),
  "fabricante": "Sem Informações",
  "nome": "H2r",
  "motor": [
    null
  ],
  "cilindradas": null,
  "modelo": null,
  "cor": [
    null
  ],
  "importada": null,
  "nacional": null,
  "topSpeed": null
}
```

// buscar motos que possuam cor roxa e laranja

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {cor: {$in: [/roxa/i, /laranja/i] }} /** variável (var) chamada (query) que recebe como valor ([/roxa/i, /laranja/i]) pelo operador de seleção ($in) no campo array (cor:). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {fabricante: 1, nome: 1, cor: 1, _id: 0} /** INCLUI no campo (field) fabricante, nome, cor, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na variável (var) chamada (query) acima, no campo (field) INCLUI os campos que foi setado como 1 (true) e OMITE os campos que foi setando como 0 (false). **/


```
{
  "fabricante": "Harley-Davidson",
  "nome": "Fat-boy",
  "cor": [
    "laranja",
    "preta",
    "bege"
  ]
}
{
  "fabricante": "Yamaha",
  "nome": "Xj-6",
  "cor": [
    "preta",
    "azul",
    "branca",
    "roxa",
    "laranja",
    "roxa",
    "laranja"
  ]
}
{
  "fabricante": "Yamaha",
  "nome": "Mt-09",
  "cor": [
    "preta",
    "roxa",
    "cinza"
  ]
}
{
  "fabricante": "Honda",
  "nome": "Fireblade",
  "cor": [
    "preta",
    "laranja",
    "pérola"
  ]
}
```


Fetched 4 record(s) in 14ms // retornou 4 records.

// Pesquisar todas as motos que não são do modelo naked!

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {modelo: {$ne: 'naked'}} /** variável (var) chamada (query) que recebe como valor ('naked') pelo operador não é igual/not equals ($ne) no campo (modelo:), ou seja, campo (modelo:) Not Equals ($ne) valor ('naked'). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = { nome: 1, modelo: 1, _id: 0 }/** INCLUI no campo (field) nome, modelo, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na variável (var) chamada (query) acima, no campo (field) INCLUI os campos que foi setado como 1 (true) e OMITE os campos que foi setando como 0 (false). **/


```
{
  "nome": "R1",
  "modelo": "sport"
}
{
  "nome": "1199",
  "modelo": "sport"
}
{
  "nome": "Fat-boy",
  "modelo": "chopper"
}
{
  "nome": "Crf-450x",
  "modelo": "terra"
}
{
  "nome": "Fireblade",
  "modelo": "sport"
}
{
  "nome": "Gsx-r1000",
  "modelo": "sport"
}
{
  "nome": "H2r",
  "modelo": null
}
{
  "nome": "Daytona-675r",
  "modelo": "sport"
}
{
  "nome": "Xr-1200x",
  "modelo": "chopper"
}
{
  "nome": "Sportster-883",
  "modelo": "chopper"
}
```


Fetched 10 record(s) in 8ms // retornou 10 records

// Pesquisar todas as motos que não são do modelo sport e nem naked! Como fazer isso?? Usando o operador chamado $nin.
/**
O operador $nin:
Seleciona os documentos em que:
O valor do campo não é especificado no array ou
O campo não existe.
Ou seja, caso possua mais de um (1) valor dentro de um (1) campo que não podem ser iguais (!=), é quando usa o operador NOT IN ($nin).
Forma do operador $nin: { field: { $nin: [<value1>, <value2> ... <valueN>]} }
Exemplo da Forma do operador $nin: { modelo: {$nin: ["naked", "sport"]}}

**/
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = { modelo: {$nin: ["naked", "sport"]}} /** variável (var) chamada (query) que recebe como valores (["naked", "sport"]) pelo operador não em/not in ($nin) no campo (modelo:), ou seja, campo (modelo:) Not In ($nin) valor (["naked", "sport"]). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {fabricante: 1, nome: 1, modelo: 1, _id: 0} /** INCLUI no campo (field) fabricante, nome, modelo, OMITE no campo (field) _id. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na variável (var) chamada (query) acima, no campo (field) INCLUI os campos que foi setado como 1 (true) e OMITE os campos que foi setando como 0 (false). **/

```
{
  "fabricante": "Harley-Davidson",
  "nome": "Fat-boy",
  "modelo": "chopper"
}
{
  "fabricante": "Honda",
  "nome": "Crf-450x",
  "modelo": "terra"
}
{
  "fabricante": "Sem Informações",
  "nome": "H2r",
  "modelo": null
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Xr-1200x",
  "modelo": "chopper"
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Sportster-883",
  "modelo": "chopper"
}
```

Fetched 5 record(s) in 2ms // retornou 5 records.

// Pesquisar todas as motos que tenham o motor 4 tempos E tenham a cilindradas não menor ou igual a 999.
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = { $and:[{ motor: { $in: [/4 tempos/i]} }, { cilindradas: {$not: {$lte: 999 }}}]} /** variável (var) chamada (query) que recebe valor ([/4 tempos/i]) pelo operador IN ($in) em seu campo (motor:) E ($and) recebe valor (999) pelo operador Menor Igual Que / Less Than Equals ($lte) junto com outro operador NOT/NEGAÇÃO ($not) em seu campo (cilindradas:).**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> query // mostrando a variável (var) chamada (query) no conosole.

```
{
  "$and": [
    {
      "motor": {
        "$in": [
          /4 tempos/i
        ]
      }
    },
    {
      "cilindradas": {
        "$not": {
          "$lte": 999
        }
      }
    }
  ]
}
```

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field) /** encontra (find()) na coleção de motos passando como parâmetros (query, field), na qual retorna o que foi setado na variável (var) chamada (query) acima, no campo (field) INCLUI os campos que foi setado como 1 (true) e OMITE os campos que foi setado como 0 (false). **/

```
{
  "fabricante": "Yamaha",
  "nome": "R1",
  "motor": [
    "4 tempos",
    "4 cilindros",
    "gasolina"
  ]
}
{
  "fabricante": "Ducati",
  "nome": "1199",
  "motor": [
    "4 tempos",
    "4 cilindros",
    "gasolina"
  ]
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Fat-boy",
  "motor": [
    "4 tempos",
    "2 cilindros",
    "gasolina"
  ]
}
{
  "fabricante": "Honda",
  "nome": "Fireblade",
  "motor": [
```

```
   "4 tempos",
   "4 cilindros",
   "gasolina"
  ]
}
{
  "fabricante": "Harley-Davidson",
  "nome": "Xr-1200x",
  "motor": [
   "4 tempos",
   "2 cilindros",
   "gasolina"
  ]
}
Fetched 5 record(s) in 4ms // retornou 5 records.


// Remova todas as motos do modelo naked E com cilindradas maior que 700.
sistemaon-VirtualBox(mongod-3.0.7) be-mean-test> var query = { $and: [ { modelo: 'naked' }, { cilindradas: { $gt: 700 } } ] } /** variável (var) chamada (query) que recebe valor ('naked') em seu campo (modelo:) E ($and) recebe valor (700) pelo operador Maior Que/Greater Than ($gt) em seu campo (cilindradas:). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> query // mostrando a variável (var) chamada (query) no conosole.
{
  "$and": [
   {
    "modelo": "naked"
   },
   {
    "cilindradas": {
     "$gt": 700
    }
   }
  ]
}


sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query) /** encontra (find()) na coleção de motos passando como parâmetros (query), na qual retorna o que foi setado na variável (var) chamada (query) acima. **/
{
  "_id": ObjectId("564e9c8db6ef4196b7cd518e"),
  "fabricante": "Kawazaki",
  "nome": "Z800",
  "motor": [
   "4 tempos",
   "3 cilindros",
   "gasolina"
  ],
  "cilindradas": 806,
  "modelo": "naked",
  "cor": [
   "preta",
   "verde",
   "preta e verde"
  ],
```

```
  "importada": false,
  "nacional": true,
  "topSpeed": 217
}
{
  "_id": ObjectId("564e9c8db6ef4196b7cd5191"),
  "fabricante": "Ducati",
  "nome": "Monster-796",
  "motor": [
    "4 tempos",
    "2 cilindros",
    "gasolina"
  ],
  "cilindradas": 796,
  "modelo": "naked",
  "cor": [
    "vermelha",
    "vermelha e branca"
  ],
  "importada": false,
  "nacional": true,
  "topSpeed": 204
}
Fetched 2 record(s) in 4ms // retornou 2 records.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.remove(query)  /** método de remoção/remove (remove()) na coleção de motos passando como parâmetros (query). Na qual primeiramente pesquisa e busca o documento da variável (var) chamada (query) que foi setado acima, caso correspondendo a consulta realizada que foi setada na variável acima, então remove (remove()) todos os documentos.  **/
Removed 2 record(s) in 48ms // removido 2 records
WriteResult({
  "nRemoved": 2
})

// Verifica se removeu!
sistemaon-VirtualBox(mongod-3.0.7) be-mean-test> var query = { $and: [ { modelo: 'naked' }, { cilindradas: { $gt: 700 } } ] } /** variável (var) chamada (query) que recebe valor ('naked') em seu campo (modelo:) E ($and) recebe valor (700) pelo operador Maior Que/Greater Than ($gt) em seu campo (cilindradas:). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query) /** encontra (find()) na coleção de motos passando como parâmetros (query), na qual retorna o que foi setado na variável (var) chamada (query) acima. **/
Fetched 0 record(s) in 1ms // retornou 0 records, ou seja, foi removido conforme o exemplo acima! ;)
```

MongoDB - Aula 05 – Exercício

Autores: Erni Augusto Fonseca Souza (Sistema_On)
Diego Araújo (DiihFilipe)

1. Importar as collections moto

sistemaon@sistemaon-VirtualBox:~$ mongoimport --host 127.0.0.1 --db be-mean-moto --collection motos --drop --file /home/sistemaon/Documents/motos.json

/**
mongoimport -> comando do MongoDB para importar.
--host 127.0.0.1 -> Especifica um nome de host para o mongod na qual conectar. Por padrão, o mongoimport se conecta no localhost (127.0.0.1) na porta (port) 27017.
--db be-mean-moto -> comando para selecionar o banco, logo em seguida com o nome do banco.
--collection motos -> comando para escrever o nome da coleção (collection), logo em seguida o nome da coleção.
--drop -> comando para dropar os dados ao banco.
--file /home/sistemaon/Documents/motos.json-> comando para selecionar o arquivo (file), logo em seguida o nome do arquivo com a extensão (.json), nesse caso estou passando o caminho (dir) completo onde o arquivo se encontra e o nome do arquivo com a extensão. **/

2015-11-19T00:08:37.966-0500connected to: 127.0.0.1
2015-11-19T00:08:37.967-0500dropping: be-mean-moto.motos
2015-11-19T00:08:43.767-0500imported 13 documents // importou 13 documentos.

2. Distinct por modelo na motos

/**
Método distinct():
Localiza os valores distintos (distinct()) em um campo específico através de uma única coleção e retorna os resultados em um array.
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = "modelo" // variável (var) chamada (query) recebe como valor uma string com nome de ("modelo").

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.distinct(query) /** localiza os valores distintos (distinct()) pelo parâmetro chamado (query) passado dentro do método, na qual foi setado acima, ou seja, localiza pelo "modelo". **/

```
[
  "sport",
  "chopper",
  "terra",
  "naked",
  null
]
```

3. As primeiras 2 páginas com .limit() e .skip() de motos (2 em 2)

/**
// Cursor, visto no resumo 2.

Método Cursor limit():
Use o método limit() para especificar o número máximo de documentos que o cursor retornará.
limit() é analogia a declaração LIMIT em um banco de dados SQL.

Método Cursos skip():
Use o método skip() em um cursor para controlar onde o MongoDB inicia/começa seus resultados retornados.
Esta abordagem pode ser extremamente útil em implementações dos resultados "paginados" ("paged" results).
**/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {} // consulta vazia ({}), visto no resumo 4.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var field = {nome: 1, _id: 0} // campo (field), visto no resumo 3.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field).limit(2).skip(2 * 0) /** método find() passando como pârametros (query, field), (vistos nos resumos 2, 3 e 4), recebendo 2 métodos cursor, limita (limit()) que recebe valor especificado (2) como pârametro, que irá retornar os documentos, e o saltar (skip()) que recebe valor (2 * 0) como parâmtro, na qual controla o MongoDB para iniciar/começar os resultados retornados. **/

```
{
  "nome": "R1"
}
{
  "nome": "1199"
}
```

Fetched 2 record(s) in 1ms // retornou 2 records.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.find(query, field).limit(2).skip(2 * 1) /** método find() passando como pârametros (query, field), (vistos nos resumos 2, 3 e 4), recebendo 2 métodos cursor, limita (limit()) que recebe valor especificado (2) como pârametro, que irá retornar os documentos, e o saltar (skip()) que recebe valor (2 * 1) como parâmtro, na qual controla o MongoDB para iniciar/começar os resultados retornados. **/

```
{
  "nome": "Fat-boy"
}
{
  "nome": "Crf-450x"
}
```

Fetched 2 record(s) in 6ms // retornou 2 records.

4 . Group e Aggregate contando a quantidade de motos de cada cor

/**
Método group():
Agrupa (group()) documentos de uma coleção pelas chaves especificadas e executa funções de agregação simples (simple aggregation), como contagens (counts) e somas (sums).
O método assemelha a um SELECT <...> GROUP BY no SQL. O método de grup() retorna um array.
**/

// GROUP

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var group = { initial: {total: 0}, // inicia (initial) minha variável chamado total recebendo valor 0.
reduce: function(curr, result) { // uma função (function) chamado reduce, que recebe como parâmetros current (curr) uma moto de cada vez, e o result (result) que é o objeto com todas as motos que colocarmos.
curr.cor.forEach(function(color) { // acessando as cores (cor) do current (curr) motos por vez. E o forEach recebe uma cor (cor) de cada vez.
if (result[color]) { //Para cada item do array do código acima, o if verifica se existe. Então o result (objeto ou (moto no nosso caso)) está verificando se existe a cor [cor] nele.
result[color]++; // se já existe ele soma. Então o result (objeto) soma a cor [cor] nele ++ .
} else { //Se não (else).
result[color] = 1; // Se não ele atribui um (1) para poder inicializar a variável. Então se não (else) o result (objeto) atribui a cor [cor] um (1).
}
result.total++; }); // No result (objeto) soma o total ++ .
}
}

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.group(group) /** método group() passando como parâmetro a variável (var) chamada (group), agrupando (group()) documentos e executa função de agregação, retornando valores setados na variável (var) chamada (group) acima. **/


```
[
  {
    "total": 36,
    "preta": 8,
    "azul e branca": 2,
    "vermelha": 1,
    "cinza": 2,
    "vermelha e branca": 2,
    "laranja": 4,
    "bege": 1,
    "vermelha e preta": 1,
    "azul": 4,
    "branca": 5,
    "roxa": 3,
    "abacate": 1,
    "pérola": 1,
    "null": 1
  }
]
```


/**
Método aggregate():
Calcula valores agregados (aggregate) para dados em uma coleção.

Operador $unwind:
Desmonta um campo de array dos documentos existentes (input) e monta (output) um documento para cada (for each) elemento.
Cada documento montado (output document) é um documento existente (input document) com o valor do campo de array substituído pelo elemento.

Operador $group:
Agrupa documentos por expressão especificada e monta (outputs) para a próxima etapa (stage) um documento para cada (for each) grupo distinto.
Os documentos de montagem (output) contém um campo (field) _id na qual contém o grupo distinto pela chave (key).

Operador $sum:
Calcula e retorna a soma (sum) de todos os valores que resulta por aplicar uma expressão específica para cada documento em um grupo de documentos, na qual,
compartilham o mesmo grupo pela chave (key).
$sum ignora valores não numéricos.
$sum é um operador de cumulação disponível somente na fase $group.

Comando de Agregação count:
Conta o número de documentos em uma coleção. Retorna um documento que possui essa contagem.
**/

// AGGREGATE
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var aggregate = [{$unwind: "$cor"}, {$group: {_id: "$cor", count: {$sum: 1}}}] /** variáver (var) chamada (aggregate) que recebe operador ($unwind:) recebendo como parâmetro ($cor) desmontando e montando o array ([cor]), o operador ($group:) recebe como parâmetro o campo (_id:) e dentro do campo o valor ($cor) agrupando documentos para um documento para cada (for each) grupo distindo, o comando de agregação (count:) recebe como parâmetro o operador ($sum:) recebendo um (1) como valor, na qual irá contar (count:) a quantidade de documentos na coleção (collection) e calcula a soma ($sum) de todos os valores para cada documento que compartilha o mesmo grupo pela chave (key). **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.aggregate(aggregate) /** método (aggregate()) recebendo como parâmetro a variável (var) chamada (aggregate) que foi setada acima, calculando valores agregados (aggregate) para dados em uma coleção, ou seja, irá calcular esses valores de acordo com o que foi setado na variável (var) (aggregate) acima. **/


```
{
  "result": [
    {
      "_id": null,
      "count": 1
    },
    {
      "_id": "pérola",
      "count": 1
    },
    {
      "_id": "abacate",
      "count": 1
    },
    {
      "_id": "roxa",
      "count": 3
    },
    {
      "_id": "branca",
      "count": 5
    },
    {
      "_id": "cinza",
      "count": 2
    },
    {
      "_id": "bege",
      "count": 1
    },
    {
      "_id": "laranja",
      "count": 4
    },
    {
      "_id": "azul",
      "count": 4
    },
    {
      "_id": "vermelha e branca",
      "count": 2
    },
    {
      "_id": "vermelha",
```

```
    "count": 1
   },
   {
    "_id": "vermelha e preta",
    "count": 1
   },
   {
    "_id": "azul e branca",
    "count": 2
   },
   {
    "_id": "preta",
    "count": 8
   }
  ],
  "ok": 1
}
```

5. Realizar 3 counts na motos.

```
/**
Método count():
        Retorna documentos contados (count()) que correspondem a uma consulta find(). O método (count()) não executa a operação do método find(), porém conta e retorna a quantidade de resultados correspondentes a consulta. **/
-> .count -- todos
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {} // consulta vazia ({}), visto no resumo 4.

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.count(query) /** método (count()) recebe como parâmetro a variável (var) chamada (query), na qual, conta (count()) o que foi setado na variável (var) (query) acima. **/
13 // contou (count()) e retornou 13 records


-> .count -- só modelo chopper
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {modelo: 'chopper'} /** consulta recebendo valor ('chopper') no campo (field) (modelo:),  visto nos resumos 2, 3 e 4. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.count(query) /** método (count()) recebe como parâmetro a variável (var) chamada (query), na qual, conta (count()) o que foi setado na variável (var) (query) acima. **/
3 // contou (count()) e retornou 3 records


-> .count -- só de quantas motos são importadas
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {importada: true} /** consulta recebendo valor (true) no campo (field) (importada:),  visto nos resumos 2, 3 e 4. **/


sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.count(query) /** método (count()) recebe como parâmetro a variável (var) chamada (query), na qual,  conta (count()) o que foi setado na variável (var) (query) acima. **/
6 // contou (count()) e retornou 6 records
```

-> .count -- motos que não são importadas E cilindradas acima de 750
sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> var query = {$and: [{importada: false}, {cilindradas: {$gt: 750}}]} /** consulta recebendo valor (false) no campo (field) (importada:), e ($and) valor (750) dentro do operador maior que ($gt:) no campo (field) (cilindradas:) visto nos resumos 3 e 4. **/

sistemaon-VirtualBox(mongod-3.0.7) be-mean-moto> db.motos.count(query) /** método (count()) recebe como parâmetro a variável (var) chamada (query), na qual, conta (count()) o que foi setado na variável (var) (query) acima. **/
3 // contou (count()) e retornou 3 records