📘 Query selectors MongoDB

Vamos conhecer um pouco sobre os querys selectors do **MongoDB**, para fazermos **buscas personalizadas** em nosso DB.

Aviso!

Os comandos dessa aula são exclusivos do **MongoDB**, ou seja, para rodá-los precisamos entrar no **MongoCLI**.

Copiar para àrea de transferência

```
$ mongosh
```

Praparativos:

- Entre no CLI do **MongoDB**;
- Crie/entre no seu banco de dados (DB);
- Crie uma collection chamada `usuarios`;
- Cadastre os seguintes usuários:
  - **nome**: 'Lucas', **idade**: 17 (int), **clt**: null;
  - **nome**: 'Tamires', **idade**: 15 (int);
  - **nome**: 'Maria', **idade**: 23 (int), **clt**: true (boolean);
  - **nome**: 'Roberto', **idade**: 17 (int).

Após ter cadastrado os usuários em sua collection `usuarios`, **consulte todos os dados** e vamos obter um resultado parecido com o exemplo abaixo:

Copiar para àrea de transferência

```
meu_db> db.usuarios.find()
[
    {
        _id: ObjectId("6115740fb912a64ab0167be0
```

```
        nome: 'Lucas',
        idade: 17,
        clt: null
    },
    {
        _id: ObjectId("61157421b912a64ab0167be1
        nome: 'Tamires',
        idade: 15
    },
    {
        _id: ObjectId("61157435b912a64ab0167be2
        nome: 'Maria',
        idade: 23,
        clt: true
    },
    {
        _id: ObjectId("6115743cb912a64ab0167be3
        nome: 'Roberto',
        idade: 17
    }
]
```

Obtendo um dado específico

Conhecemos o método `find()`, ele tem a função de retornar os dados. Vamos supor que precisamos adquirir os dados da **Tamires**, para isso utilizamos o seguinte comando:

> **Aviso!**
>
> **Sintaxe**: `db.nome_collection.find({chave: valor})`, utiliza-se **chave/valor** para especificar o determinado dado.

Copiar para àrea de transferência

```
meu_db> db.usuarios.find({nome: 'Tamires'})
[
  {
      _id: ObjectId("61157421b912a64ab0167be1")
      nome: 'Tamires',
      idade: 15
```

```
  }
]
```

Operadores lógicos

Obtendo dados com a exclusão de um dado específico

Vamos supor que precisamos adquirir todos os dados **menos** o do **Roberto**, para isso utiliza-se o seguinte comando:

> **Dica!**
>
> **Sintaxe**: **db.nome_collection.find({chave: {$ne: valor}})**, além de especificar **chave/valor**, utilizamos o operador **$ne** (equivalente a **!** ou **not**), para dar instrução ao DB de que nos retorne todos EXCETO o seguinte dado.

Copiar para área de transferência

```
meu_db> db.usuarios.find({nome: {$ne: 'Roberto'
[
    {
        _id: ObjectId("6115740fb912a64ab0167be0
        nome: 'Lucas',
        idade: 17,
        clt: null
    },
    {
        _id: ObjectId("61157421b912a64ab0167be1
        nome: 'Tamires',
        idade: 15
    },
    {
        _id: ObjectId("61157435b912a64ab0167be2
        nome: 'Maria',
        idade: 23,
        clt: true
    }
]
```

Obtendo dado maior que (>) e maior ou igual que (>=)

Para obter usuários com idade **maior que** 17 anos, usa-se o seguinte comando:

> **Dica!**
>
> **Sintaxe**: `db.nome_collection.find({chave: {$gt: valor}})`, além de especificar **chave/valor**, utilizamos o operador **$gt** (equivalente a `>`), para dar instrução ao DB de que nos retorne todos os dados maiores que o valor especificado.

Copiar para àrea de transferência

```
meu_db> db.usuarios.find({idade: {$gt: 17}})
[
    {
      _id: ObjectId("61157435b912a64ab0167be2")
      nome: 'Maria',
```

```
    idade: 23,
    clt: true
  }
]
```

Caso quiséssemos idade **maior ou igual** a 17 anos, usaríamos o seguinte comando:

Copiar para área de transferência

```
meu_db> db.usuarios.find({idade: {$gte: 17}})
[
  {
    _id: ObjectId("6115740fb912a64ab0167be0")
    nome: 'Lucas',
    idade: 17,
    clt: null
  },
  {
    _id: ObjectId("61157435b912a64ab0167be2")
    nome: 'Maria',
```

```
    idade: 23,
    clt: true
  },
  {
    _id: ObjectId("6115743cb912a64ab0167be3")
    nome: 'Roberto',
    idade: 17
  }
]
```

> **Dica!**
>
> **Sintaxe**: `db.nome_collection.find({chave: {$gte: valor}})`, além de especificar **chave/valor**, utilizamos o operador **$gte** (equivalente a `>=`), para dar instrução ao DB de que nos retorne todos os dados maiores ou iguais ao valor especificado.

Obtendo dado menor que (<) e menor ou igual que (<=)

Seguindo o exemplo, para obter usuários com idade **menor que** 17 anos, usa-se o seguinte comando:

Copiar para área de transferência

```
meu_db> db.usuarios.find({idade: {$lt: 17}})
[
    {
      _id: ObjectId("61157421b912a64ab0167be1")
      nome: 'Tamires',
      idade: 15
    }
]
```

Dica!

Sintaxe: `db.nome_collection.find({chave: {$lt: valor}})`, além de especificar **chave/valor**, utilizamos o operador **$lt** (equivalente a `<`), para dar instrução ao DB de que nos retorne todos os dados menores que o valor especificado.

Caso quiséssemos obter dados com idade **menor ou igual** a 17 anos, usaríamos o seguinte comando:

Copiar para área de transferência

```
meu_db> db.usuarios.find({idade: {$lte: 17}})
[
    {
      _id: ObjectId("6115740fb912a64ab0167be0")
      nome: 'Lucas',
      idade: 17,
      clt: null
    },
    {
      _id: ObjectId("61157421b912a64ab0167be1")
      nome: 'Tamires',
      idade: 15
    },
    {
      _id: ObjectId("6115743cb912a64ab0167be3")
      nome: 'Roberto',
      idade: 17
    }
]
```

Dica!

Sintaxe: `db.nome_collection.find({chave: {$lte: valor}})`, além de especificar **chave/valor**, utilizamos o operador **$lte** (equivalente a `<=`), para dar instrução ao DB de que nos retorne todos os dados menores ou iguais ao valor especificado.

Obtendo dados que NÃO correspondem o especificado

Vamos pensar em uma situação onde queremos retornar todos os dados que **NÃO** sejam maiores que 15 anos.

Copiar para àrea de transferência

```
db.usuarios.find({idade: {$not: {$gt: 15}}})
[
    {
      _id: ObjectId("611babc5eb34e426e79fd6fc")
      nome: 'Tamires',
```

```
        idade: 15
    }
]
```

> **Dica!**
>
> **Sintaxe**: `db.nome_collection.find({chave: {$not: {$gt: valor}}})`, utiliza-se **chave** para especificar o campo a ser verificado e o operador `$not` que recebe um outro **operador** ou uma **expressão** (regex), com seus determinados valores.

Obtendo dados que possuem uma chave específica

Agora vamos imaginar que precisamos fazer uma busca por dados que tenham a **chave** `clt`, observe o seguinte comando:

Copiar para área de transferência

```
meu_db> db.usuarios.find({clt: {$exists: true}}
[
    {
        _id: ObjectId("6115740fb912a64ab0167be0
        nome: 'Lucas',
        idade: 17,
        clt: null
    },
    {
        _id: ObjectId("61157435b912a64ab0167be2
        nome: 'Maria',
        idade: 23,
        clt: true
    }
]
```

Dica!

Sintaxe: `db.nome_collection.find({chave: {$exists: boolean}})`, utiliza-se **chave** para especificar o campo a ser verificado e o operador `$exists` que recebe um valor **boolean**, onde se for `true`, o comando nos trará todos os dados que possuem aquela chave, caso contrário trará todos os dados que não possuem aquela chave.

Aviso!

Reparem que mesmo o campo `clt` sendo `null` ele é considerado, já que existe dentro daquele documento.

Obtendo dados que possuem alguns dos valores especificados

Agora vamos aprender um operador que especificamos a chave que queremos, por exemplo `idade`, e queremos que seja retornado **alguns valores** determinados.

Copiar para área de transferência

```
meu_db> db.usuarios.find({idade: {$in: [29, 23,
            [
                {
                    _id: ObjectId("61157421b912
                    nome: 'Tamires',
                    idade: 15
                },
                {
                    _id: ObjectId("61157435b912
                    nome: 'Maria',
                    idade: 23,
                    clt: true
                }
            ]
```

Usando AND e OR

AND

Imagine uma situação onde queremos obter um dado com duas ou mais condições. Simplesmente podemos usar a `,` que fará o papel do **AND**, observe o seguinte exemplo:

Copiar para àrea de transferência

```
db.usuarios.find({idade: 17, nome: 'Roberto'})
[
    {
      _id: ObjectId("6115743cb912a64ab0167be3")
      nome: 'Roberto',
      idade: 17
    }
]
```

Dica!

Sintaxe: `db.nome_collection.find({chave: {$in: [valor1, valor2, ...]}})`, utiliza-se **chave** para especificar o

campo a ser verificado e o operador `$in` que é uma **lista** de valores que será determinada, para ser retornados.

Aviso!

Caso colocássemos no lugar de **Roberto** o nome **Maria**, **NÃO** haveria retorno, já que não temos um usuário que se chame **Maria** com **17 anos**. O **AND** representado pela `,` exige que as duas condições sejam verdadeiras para algum resultado ser retornado.

OR

Agora vamos supor que queremos obter um usuário com a idade de 23 anos **ou** 15 anos. Usaremos o comando `$or`, observe o seguinte comando:

Copiar para àrea de transferência

```
db.usuarios.find({$or:[{idade: 23}, {idade: 15}
[
```

```
  {
    _id: ObjectId("61157421b912a64ab0167be1"),
    nome: 'Tamires',
    idade: 15
  },
  {
    _id: ObjectId("61157435b912a64ab0167be2"),
    nome: 'Maria',
    idade: 23
  }
]
```

Importante!

Reparem que o operador `$or` tem seus valores em formato de **dicionários** `{}` dentro de uma **lista** `[]`, isso é importante para o funcionamento do **OR**, caso não seja inserido no comando ocorrerá um erro.

Aviso!

Caso fosse alternado o valor **23** para **24**, o retorno seria apenas os dados da **Tamires** de **15 anos**. O **OR** representado por `$or` funciona caso **algum** dos dados passados em **chave/valor** sejam verdadeiros, para que possa ser retornado algum resultado.

Referências!

[Operators - Docs | MongoDb]

[Query selector | Blog felipetoscano]