

## Aula 3 - Trabalhando com dados

### Docupedia Export

Author:Ferro Alisson (CtP/ETS)

Date:15-Aug-2023 13:11

## Table of Contents

|                            |           |
|----------------------------|-----------|
| <b>1 Inserindo dados</b>   | <b>6</b>  |
| <b>2 Buscando dados</b>    | <b>8</b>  |
| <b>3 Atualizando dados</b> | <b>16</b> |
| <b>4 Excluindo dados</b>   | <b>20</b> |

Para mostrarmos na tela as databases existentes usamos o código

```
show databases
```

e clicar em executar no canto direito superior, será exibido todas as databases

**{}** Playground Result **×**

```
1  [
2    {
3      "name": "Teste",
4      "sizeOnDisk": 131072,
5      "empty": false
6    },
7    {
8      "name": "admin",
9      "sizeOnDisk": 40960,
10     "empty": false
11   },
12   {
13     "name": "config",
14     "sizeOnDisk": 110592,
15     "empty": false
16   },
17   {
18     "name": "local",
19     "sizeOnDisk": 73728,
20     "empty": false
21   },
22   {
23     "name": "test",
24     "sizeOnDisk": 122880,
```

Para criar uma database, basta digitar o comando, onde só precisamos usar use e o nome da database

```
use "nome_do_database"
```

# 1 Inserindo dados

A database somente é criada ao inserir uma collection, para isso vamos utilizar, para isso vamos inserir uma collection com o comando

```
db.people.insertOne({
  name: "Alisson",
  lastname: "Ferro",
  salary: 1234
})
```

onde db se refere a database, people é o nome da collection, e insertOne é o comando para inserir um documento,

Para inserir vários dados de uma só vez utilizamos o **insertMany**.

```
db.people.insertMany([
  {
    name: 'Queila',
    lastname: 'Lima',
    salary: 1234
  },
  {
    name: 'Donathan',
    lastname: 'Goncalves',
    salary: 1234
  },
])
```

Ou também podemos passar como um código Javascript, criando uma constante ou variável com um array de objetos e após isso dentro do insertMany colocamos a nossa variável criada

```
const arrpeople = [
  {
    name: 'Luís',
    lastname: 'Balem',
    salary: 1234
  },
]
```

```
{
  name: 'Leonardo',
  lastname: 'Trevisan',
  salary: 1234
},
]
db.people.insertMany(arrpeople)
```

## 2 Buscando dados

O MongoDB utiliza JSON-like documents para fazer consultas. As consultas consistem em pares chave-valor e operadores. Por exemplo, para encontrar documentos com um campo específico igual a um valor, você usaria a notação { campo: valor }. Para aplicar operadores de comparação, você pode usar { campo: { operador: valor } }.

Agora para buscar todos documentos, vamos utilizar o comando find();

```
db.people.find()
```



```
{ } Playground Result X
1  [
2    {
3      Edit Document
4      "_id": {
5        "$oid": "64b575535e5947a5e8dd26bd"
6      },
7      "name": "Alisson",
8      "lastname": "Ferro",
9      "salary": 1234
10   },
11  {
12    Edit Document
13    "_id": {
14      "$oid": "64b575bce32c7de8af581982"
15    },
16    "name": "Queila",
17    "lastname": "Lima",
18    "salary": 1234
19  },
20  {
21    Edit Document
22    "_id": {
23      "$oid": "64b575bce32c7de8af581983"
24    },
25    "name": "Donathan",
26    "lastname": "Goncalves",
27    "salary": 1234
28  },
29  {
30    Edit Document
31    "_id": {
32      "$oid": "64b5762774efa9a68b59a8b3"
```

semelhante ao **where** do SQL, podemos passar parâmetros para filtrar

```
db.people.find({ name: 'Alisson' })
```

E nosso resultado será, isso mostra todas as collections que combinam com o parâmetro passado e o resultado será semelhante ao abaixo

```
{ Playground Result X
1  [
2  |  {
3  |      Edit Document
4  |      "_id": {
5  |          "$oid": "64b575535e5947a5e8dd26bd"
6  |      },
7  |      "name": "Alisson",
8  |      "lastname": "Ferro",
9  |      "salary": 1234
10 |  }
```

Também é possível passar valores diferentes como parâmetro, como um regex

```
db.people.find({ name: /n/ });
```

E assim será exibido todos os dados que contém "n" no campo name

```
{ Playground Result X
1  [
2  {
3    "_id": {
4      "$oid": "64b575535e5947a5e8dd26bd"
5    },
6    "name": "Alisson",
7    "lastname": "Ferro",
8    "salary": 1234
9  },
10 {
11   "_id": {
12     "$oid": "64b575bce32c7de8af581983"
13   },
14   "name": "Donathan",
15   "lastname": "Goncalves",
16   "salary": 1234
17 },
18 {
19   "_id": {
20     "$oid": "64b5762774efe0a68b50e8b4"
21   },
22   "name": "Leonardo",
23   "lastname": "Trevisan",
24   "salary": 1234
25 }
26 ]
```

Podemos também consultar documentos com mais de uma condição usando o operador lógico "\$and":

```
db.people.find({ $and: [{ name: 'Alisson' }, { lastname: 'Balem' }] })
```

Como não temos nenhum Alisson Balem, nosso retorno será um array vazio

 Playground Result 

1 

Consultar documentos com um campo maior que um valor usando o operador "\$gt":

```
db.people.find({ salary: { $gt: 123 } })
```

E será exibido todos os valores que possuir salário maior que 123

**{}** Playground Result ×

```
1  [
2    {
3      "_id": {
4        "$oid": "64b575535e5947a5e8dd26bd"
5      },
6      "name": "Alisson",
7      "lastname": "Ferro",
8      "salary": 1234
9    },
10   {
11     "_id": {
12       "$oid": "64b575bce32c7de8af581982"
13     },
14     "name": "Queila",
15     "lastname": "Lima",
16     "salary": 1234
17   },
18   {
19     "_id": {
20       "$oid": "64b575bce32c7de8af581983"
21     },
22     "name": "Donathan",
23     "lastname": "Goncalves",
24     "salary": 1234
25   },
26   {
27     "_id": {
28       "$oid": "64b5762774efe0a68b50e8b3"
29     },
30     "name": "Luis",
31     "lastname": "Balem",
```

Como todos possuem o mesmo salário, então foi exibido todas as pessoas do banco de dados

Outros operadores de comparação de consulta

| Name  | Description  |
|-------|--|
| \$eq  | Corresponde a valores que são iguais a um valor especificado.              |
| \$gt  | Corresponde a valores maiores que um valor especificado.                   |
| \$gte | Corresponde a valores maiores ou iguais a um valor especificado.           |
| \$in  | Corresponde a qualquer um dos valores especificados em uma matriz.         |
| \$lt  | Corresponde a valores que são menores que um valor especificado.           |
| \$lte | Corresponde a valores menores ou iguais a um valor especificado.           |
| \$ne  | Corresponde a todos os valores que não são iguais a um valor especificado. |
| \$nin | Não corresponde a nenhum dos valores especificados em uma matriz           |

Para mais informações [clique aqui](#)

Podemos também consultar documentos e retornar apenas determinados campos usando a projeção, ou seja, fazer uma consulta e retornar somente os campos escolhidos

```
db.people.find({ salary: { $gte: 123 } }, { name: 1, lastname:1 })
```

```
{ } Playground Result X
1  [
2    {
3      "_id": {
4        "$oid": "64b575535e5947a5e8dd26bd"
5      },
6      "name": "Alisson",
7      "lastname": "Ferro"
8    },
9    {
10     "_id": {
11       "$oid": "64b575bce32c7de8af581982"
12     },
13     "name": "Queila",
14     "lastname": "Lima"
15   },
16   {
17     "_id": {
18       "$oid": "64b575bce32c7de8af581983"
19     },
20     "name": "Donathan",
21     "lastname": "Goncalves"
22   },
23   {
24     "_id": {
25       "$oid": "64b5762774efe0a68b50e8b3"
26     },
27     "name": "Luis",
28     "lastname": "Balem"
29   },
30   {
31     "_id": {
```

### 3 Atualizando dados

Para atualizar documentos no banco de dados MongoDB, você pode usar o método `update()` ou os métodos mais recentes `updateOne()` ou `updateMany()`. A escolha do método depende se você deseja atualizar apenas um documento ou vários documentos que correspondam a um critério específico.

```
db.people.updateOne(  
  { _id: ObjectId('64b575535e5947a5e8dd26bd') },  
  { $set:{ name:"Alisson Alterado" }}  
);
```

Agora iremos buscar esse dado com o `find`

```
db.people.find(  
  { _id: ObjectId('64b575535e5947a5e8dd26bd') }  
);
```

E como resultado recebemos





```
{} Playground Result X
1  [
2  {
3    "_id": {
4      "$oid": "64b575535e5947a5e8dd26bd"
5    },
6    "name": "Alisson Alterado",
7    "lastname": "Ferro",
8    "salary": 1234
9  }
10 ]
```

Podemos também editar mais de um documento ao mesmo tempo com o método `updateMany`

```
db.people.updateMany(
  { salary: 1234 },
  { $set: { salary: 12345 } }
);
```

E fazendo um `find`, recebemos o resultado a seguir

 Playground Result 

```
1  [
2    {
3      "_id": {
4        "$oid": "64b575535e5947a5e8dd26bd"
5      },
6      "name": "Alisson Alterado",
7      "lastname": "Ferro",
8      "salary": 12345
9    },
10   {
11     "_id": {
12       "$oid": "64b575bce32c7de8af581982"
13     },
14     "name": "Queila",
15     "lastname": "Lima",
16     "salary": 12345
17   },
18   {
19     "_id": {
20       "$oid": "64b575bce32c7de8af581983"
21     },
22     "name": "Donathan",
23     "lastname": "Goncalves",
24     "salary": 12345
25   },
26   {
27     "_id": {
28       "$oid": "64b5762774efe0a68b50e8b3"
29     },
30     "name": "Luis",
31     "lastname": "Balem",
```

Podemos também utilizar os operadores de comparação de consulta

## 4 Excluindo dados

Para excluir dados do banco de dados MongoDB, você pode usar o método `deleteOne()` para remover um único documento ou o método `deleteMany()` para remover vários documentos que correspondam a um critério específico.

Excluir um único documento que corresponda a um critério específico:

```
db.people.deleteOne({  
  name: /Alisson/  
})
```

E como resultado vamos visualizar

```
{ } Playground Result X
1  [
2    {
3      "_id": {
4        "$oid": "64b575bce32c7de8af581982"
5      },
6      "name": "Queila",
7      "lastname": "Lima",
8      "salary": 12345
9    },
10   {
11     "_id": {
12       "$oid": "64b575bce32c7de8af581983"
13     },
14     "name": "Donathan",
15     "lastname": "Goncalves",
16     "salary": 12345
17   },
18   {
19     "_id": {
20       "$oid": "64b5762774efe0a68b50e8b3"
21     },
22     "name": "Luis",
23     "lastname": "Balem",
24     "salary": 12345
25   },
26   {
27     "_id": {
28       "$oid": "64b5762774efe0a68b50e8b4"
29     },
30     "name": "Leonardo",
31     "lastname": "Trevisan",
```

Como é possível ver, o Alisson foi excluído

Agora iremos ver o deleteMany()

```
db.people.deleteMany({  
  name: /n/  
})
```

Agora deletamos todos os nomes que contenham a letra "n"

```
{ } Playground Result X  
1  [  
2    {  
3      "_id": {  
4        "$oid": "64b575bce32c7de8af581982"  
5      },  
6      "name": "Queila",  
7      "lastname": "Lima",  
8      "salary": 12345  
9    },  
10   {  
11     "_id": {  
12       "$oid": "64b5762774efe0a68b50e8b3"  
13     },  
14     "name": "Luis",  
15     "lastname": "Balem",  
16     "salary": 12345  
17   }  
18 ]
```

Desafio: Crie um catálogo de produtos onde você pode adicionar, atualizar e excluir informações sobre os produtos. Cada produto pode ter um nome, descrição, preço, categoria e quantidade em estoque.