# 📘 Manipulando dados com MongoDB

Nessa teoria aprenderemos as quatro operações no **MongoDB** reponsáveis pelo **CRUD** (**C**reate, **R**ead, **U**pdate e **D**elete).

---

## Objetivo

Criar um **DB** chamado `crud` e uma **coleção** chamada `users` para armazenar o `nome` e `idade` dos usuários.

---

## Preparativos

> **Aviso!**
>
> Os comandos dessa aula são exclusivos do **MongoDB**, ou seja, para rodá-los precisamos

entrar no **MongoCLI**.

Copiar para área de transferência

```
$ mongosh
```

Crie o DB `crud`:

Copiar para área de transferência

```
test> use crud

switched to db crud
```

---

Crie a coleção `users`:

Copiar para área de transferência

```
crud> db.users.insertOne({nome: "John Doe", ida

{
  acknowledged: true,
  insertedId: ObjectId("611523a7f91912c50a06400
}
```

**Dica!**

Perceba que a coleção `users` foi criada com o comando de inserção de documentos `insertOne`, caso não tenha entendido não se preocupe, vamos ver sobre **inserts** logo abaixo.

---

# CREATE

O **MongoDB** fornece os seguintes métodos para **inserir documentos** em uma coleção:

- `db.collection.insertOne()` - Escreve apenas 1 documento.
- `db.collection.insertMany()` - Escreve 1 ou mais documentos.

**Importante!**

Os métodos de inserção tem dois comportamentos bem interessantes, são eles:

> Caso a coleção não exista, ela será criada para logo em seguida inserir o documento;
>
> É gerado um identificador (`id`) de forma **automática** para o documento inserido.

Uma outra coisa que devemos prestar atenção é na hora de criar a **key** do **JSON**, por exemplo: Se a **key** `nome` fosse `nome completo`, precisaríamos envolver ela entre **aspas**:

Copiar para área de transferência

```
test> db.collection.insertOne({"nome comp
```

Vamos **inserir** mais documentos na **coleção** `users`, dessa vez com o `insertMany()`:

Copiar para área de transferência

```
crud> db.users.insertMany([
  {nome: "Maria Rosa", idade: 33},
  {nome: "Jose Silva", idade: 45}
])

{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6115242ef91912c50a064003"),
    '1': ObjectId("6115242ef91912c50a064004")
```

```
  }
}
```

---

# READ

As operações de **leitura** recuperam documentos de uma coleção. O **MongoDB** fornece os seguintes métodos para ler documentos de uma coleção:

- `db.collection.find()` - Retorna documento(s).

> **Dica!**
>
> O `find()` possui alguns cursores para tratar o retorno da leitura. Um **cursor** é uma coleção **MongoDB** do **documento que é retornado** na execução do método `find()`. Veja os mais utilizados:

Copiar para área de transferência

```
test> db.collection.insertOne({"nome compl
```

`cursor.pretty()` - O pretty organiza seu retorno em um formato fácil de ler. Veja um exemplo de aplicação:

Copiar para área de transferência

```
db.collection.find().pretty()
```

`cursor.limit(number)` - O limit recebe um número por parâmetro para especificar o número máximo de documentos que podem ser retornados. Veja um exemplo de aplicação:

Copiar para área de transferência

```
db.collection.find().limit(5)
```

`cursor.sort()` - O sort recebe um campo por parâmetro para que o resultado seja classificado em ordem alfabética a partir daquele campo. Veja um exemplo de aplicação:

Copiar para àrea de transferência

```
db.collection.find().sort({nome: 1})
```

Especifique no parâmetro de classificação o campo ou campos pelos quais classificar e um valor de `1` ou `-1` para especificar uma classificação **crescente ou decrescente**, respectivamente.

Lendo a coleção `users`:

Copiar para àrea de transferência

```
crud> db.users.find()

[
```

```
  {
    _id: ObjectId("611523a7f91912c50a064002"),
    nome: 'John Doe',
    idade: 25
  },
  {
    _id: ObjectId("6115242ef91912c50a064003"),
    nome: 'Maria Rosa',
    idade: 33
  },
  {
    _id: ObjectId("6115242ef91912c50a064004"),
    nome: 'Jose Silva',
    idade: 45
  }
]
```

---

# UPDATE

As operações de **atualização** modificam os documentos existentes em uma coleção . O **MongoDB**

fornece os seguintes métodos para atualizar documentos de uma coleção:

- `db.collection.updateOne()` - Atualiza o 1º documento achado.
- `db.collection.updateMany()` - Atualiza 1 ou mais documentos, de acordo com a seleção.

Vamos supor que o usuário com o nome de Maria Rosa queira **atualizar sua idade**, como faríamos isso?

Copiar para àrea de transferência

```
crud> db.users.updateOne({nome: "Maria Rosa"},


{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Ao olhar os registros percebemos que a idade da Maria Rosa foi **atualizada**:

Copiar para área de transferência

```
crud> db.users.find().pretty()

[
  {
    _id: ObjectId("611523a7f91912c50a064002"),
    nome: 'John Doe',
    idade: 25
  },
  {
    _id: ObjectId("6115242ef91912c50a064003"),
    nome: 'Maria Rosa',
    idade: 25
  },
  {
    _id: ObjectId("6115242ef91912c50a064004"),
```

```
    nome: 'Jose Silva',
    idade: 45
  }
]
```

Como a intenção era atualizar somente um documento, foi utilizado o `upadeteOne()`. Para alterar mais de um documento, utilize o `updateMany()`, por exemplo:

**Alterar o nome** de todas a pessoas que tem 25 anos para João da Silva:

Copiar para àrea de transferência

```
crud> db.users.updateMany({idade: 25}, {$set: {

{
  acknowledged: true,
  insertedId: null,
```

```
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

Ao olhar os registros percebemos que os usuários de idade 25 tiveram o nome **atualizado**:

Copiar para àrea de transferência

```
crud> db.users.find().pretty()

[
  {
    _id: ObjectId("611523a7f91912c50a064002"),
    nome: 'João da Silva',
    idade: 25
  },
  {
    _id: ObjectId("6115242ef91912c50a064003"),
```

```
    nome: 'João da Silva',
    idade: 25
  },
  {
    _id: ObjectId("6115242ef91912c50a064004"),
    nome: 'Jose Silva',
    idade: 45
  }
]
```

**Importante!**

O `$set` é um operador de atualização, ele adiciona novos campos aos documentos. Além do `$set` existem outros operadores de atualização, para saber mais acesse este link.

Lembre que ao utilizar o método `Many`, que trabalha com 1 ou mais documentos, precisamos obrigatoriamente passar os dados no formato de **lista**.

# DELETE

As operações de **exclusão** removem documentos de uma coleção. O **MongoDB** fornece os seguintes métodos para excluir documentos de uma coleção:

- `db.collection.deleteOne()` - Deleta o 1º documento achado.
- `db.collection.deleteMany()` - Deleta 1 ou mais documentos.

Em nosso DB `crud` na coleção `users` temos dois usuários com o nome de João da Silva. Deletaremos um deles utilizando o `deleteOne()`, mas se quiséssemos apagar os dois usuários, teríamos que utilizar o `deleteMany()`:

Copiar para área de transferência

```
crud> db.users.deleteOne({nome: "João da Silva"

{ acknowledged: true, deletedCount: 1 }
```

Ao fazer isso, foi deletado o primeiro documento encontrado com o campo `nome` igual a `"João da Silva"`.

**Aviso!**

Ao utilizar o comando de **delete** da seguinte maneira, todos os dados da coleção serão deletados. Use com sabedoria e somente quando for necessário.

Copiar para àrea de transferência

```
db.collection.deleteMany({})
```

**Referências!**

[CRUD - Docs | Mongo]