### **Tipos de instalação**

O MongoDB disponibiliza três tipos de instalação:

1. Standalone
   * Apenas indicado para ambientes de desenvolvimento
   * Não exige nenhum tipo de configuração relativa à segurança;
2. Replica Set
   * É o mínimo indicado para ambientes de produção.
   * Neste tipo, os dados são replicados em cada um dos servidores do cluster e temos apenas um ponto de escrita.
   * Em alguns casos, podemos utilizar os demais servidores para escalar a leitura.
3. Shard
   * Esse é um tipo de instalação no qual podemos escalar a escrita de informações no banco.
   * Os dados são divididos no cluster através de chaves de partição que chamamos de *shard keys* .
   * A *shard key* pode ser composta por um ou mais atributos do documento, e sua escolha pode afetar a performance, eficiência e escalabilidade do banco.
   * Escalar a escrita significa dar mais capacidade para que o banco de dados processe mais operações, aumentando a performance.

### **Escolhendo a distribuição**

O MongoDB está disponível para os seguintes sistemas operacionais:

* Linux
* MacOS
* Windows

Você utilizará o *MongoDB Community Edition* , escolha a instalação especifica para a sua distribuição Linux [deste link](https://docs.mongodb.com/manual/administration/install-on-linux/) .

Caso você esteja utilizando MacOS, siga as instruções [deste link](https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/) .

### **Instalando o MongoDB Community Edition**

Os próximos passos para instalação do MongoDB serão realizados com base na distribuição [Ubuntu](https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/) utilizando o apt *package manager* .

#### 1. Importando a chave pública utilizada pelo gerenciamento de pacotes

Abra o terminal e utilize o comando abaixo para importar chave pública GPG do MongoDB .

Copiar

wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

Este comando deve retornar um OK .

Porém, se você receber um erro indicando que gnupg não está instalado, faça como abaixo:

1. Instalar o gnupg e as bibliotecas necessárias através do comando:

Copiar

sudo apt-get install gnupg

1. Após a instalação, tente importar a chave outra vez:

Copiar

wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

#### 2. Crie o arquivo de lista ( *list file* ) para o MongoDB

Crie o arquivo /etc/apt/sources.list.d/mongodb-org-4.4.list para o Ubuntu 20.04 (Focal):

Copiar

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list

#### 3. Atualize o banco de dados local de pacotes

Copiar

sudo apt-get update

#### 4. Instale os pacotes do MongoDB

Você pode instalar a última versão estável do \*\*MongoDB , ou uma versão específica.

Para instalar a última versão estável, utilize o comando abaixo:

Copiar

sudo apt-get install -y mongodb-org

Pronto! Agora você já tem a última versão estável do MongoDB instalada e pronta para ser executada na sua máquina. Mas antes, vamos dar uma olhada nos pacotes que foram instalados.

### **Os pacotes instalados**

Você deve ter notado que, durante a instalação, alguns pacotes adicionais foram instalados:

1. mongodb-org-server : esse pacote contém o que podemos chamar de "servidor" do MongoDB . Contém todos os recursos necessários para que uma instância do banco seja executada;
2. mongodb-org-shell : o *shell* é onde você se conecta com o MongoDB através do terminal. É uma interface que suporta JavaScript e é super completa para administração de instâncias e clusters;
3. mongodb-org-mongos : pronuncia-se "Mongo S" e só se faz necessário em ambientes *Shard* . Não entraremos em detalhes sobre ele agora;
4. mongodb-org-tools : esse pacote contém algumas ferramentas nativas do MongoDB. Como por exemplo:
   * mongodump : ferramenta para extrair dados no formato BSON (falaremos dele mais adiante). Em alguns ambientes, pode fazer parte da estratégia de backup;
   * mongorestore : ferramenta para restaurar backups gerados pelo *mongodump* ;
   * mongoimport : ferramenta para importar arquivos JSON , CSV ou TSV para uma instância do MongoDB ;
   * mongoexport : exporta dados de uma instância do MongoDB para arquivos JSON ou CSV .

### **Executando o MongoDB Community Edition**

#### Considerações sobre o *ulimit*

Alguns sistemas operacionais baseados em *UNIX* limitam os recursos de sistema que uma sessão pode utilizar. Esses limites têm grande impacto negativo para a operação do MongoDB , e em ambientes de produção devem ser observados com muita atenção. Veja a seção [UNIX ulimit Settings](https://docs.mongodb.com/manual/reference/ulimit/) da documentação do MongoDB para maiores informações.

#### Diretórios de trabalho

Se você instalou o MongoDB via apt (gerenciador de pacotes do Linux), então algumas configurações são executadas e mantidas em diretórios do sistema operacional. Por padrão, no Linux, os dados ficarão armazenados em /var/lib/mongodb , e o log de funcionamento em /var/log/mongodb .

No MacOS, os dados e os logs ficam em /usr/local/var/mongodb e /usr/local/var/log/mongodb , respectivamente.

Por padrão, o MongoDB roda utilizando a conta do usuário mongodb , que também foi criada durante a instalação. Se você quiser rodar uma instância com outro usuário, deverá dar as permissões para ele nos diretórios de dados e log.

#### Arquivo de configuração

O pacote oficial inclui um [arquivo de configuração](https://docs.mongodb.com/manual/reference/configuration-options/#conf-file) ( /etc/mongod.conf ). Essas configurações (como especificação dos caminhos dos diretórios de dados e log) têm efeito após o *startup* da instância (ou seja, quando ela for iniciada). Logo, se você fizer qualquer modificação nesse arquivo com a instância do MongoDB rodando, deverá reiniciá-la para que tenha efeito.

### **Mãos à obra, vamos executar!**

##### 1. Iniciando o MongoDB

No Linux:

Copiar

sudo service mongod start

No MacOS:

Copiar

brew services start mongodb-community

##### 2. Verifique se o MongoDB foi iniciado com sucesso

No Linux:

Copiar

sudo service mongod status

No MacOS:

Copiar

brew services list | grep mongodb-community

Você também pode checar o arquivo de log que, por padrão, é localizado em /var/log/mongodb/mongod.log , no Linux. Ou em /usr/local/var/log/mongodb , no Mac. Você pode verificar se a instância está rodando e pronta para conexões através da linha abaixo:

Copiar

[initanlisten] waiting for connections on port 27017\*\*

##### Parando a instância

No Linux:

Copiar

sudo service mongod stop

No MacOS:

Copiar

brew services stop mongodb-community

##### Reiniciando a instância

No Linux:

Copiar

sudo service mongod restart

No MacOS:

Copiar

brew services restart mongodb-community

#### Configurando a inicialização do servidor do MongoDB

Por padrão, após a instalação, seu servidor vai estar configurado para não iniciar junto ao sistema. Caso queira ativar o início automático quando ligar o computador, utilize o comando:

Copiar

sudo systemctl enable mongod.service

Caso não queira mais que isso aconteça (para poupar memória RAM, por exemplo), você pode desativar o início automático utilizando o comando:

Copiar

sudo systemctl disable mongod.service

Na primeira vez que for utilizar o MongoDB após ligar o computador, será necessário iniciar o servidor com o comando:

Copiar

sudo service mongod start

### **Desinstalando o MongoDB**

Caso sua instalação tenha retornado algum problema, siga os passos abaixo para desinstalar e tente realizar a instalação novamente.

### Preste muita atenção aos comandos ###

Pare sua instância do mongodb:

Copiar

sudo service mongod stop

Primeiro, remova todos os pacotes instalados:

Copiar

sudo apt-get purge mongodb-org\*

Agora, remova os arquivos de dependências que não são mais necessários. Em seguida, remova as versões antigas dos arquivos de pacotes.

Copiar

sudo apt-get autoremove

Copiar

sudo apt-get autoclean

Após isso, remova os arquivos do mongodb que podem ter ficado para trás.

Copiar

sudo rm -rf /var/log/mongodb

Copiar

sudo rm -rf /var/lib/mongodb

Se a desinstalação for concluída com sucesso, o comando mongod --version não deve retornar a versão do seu mongodb.

#### Informação importante

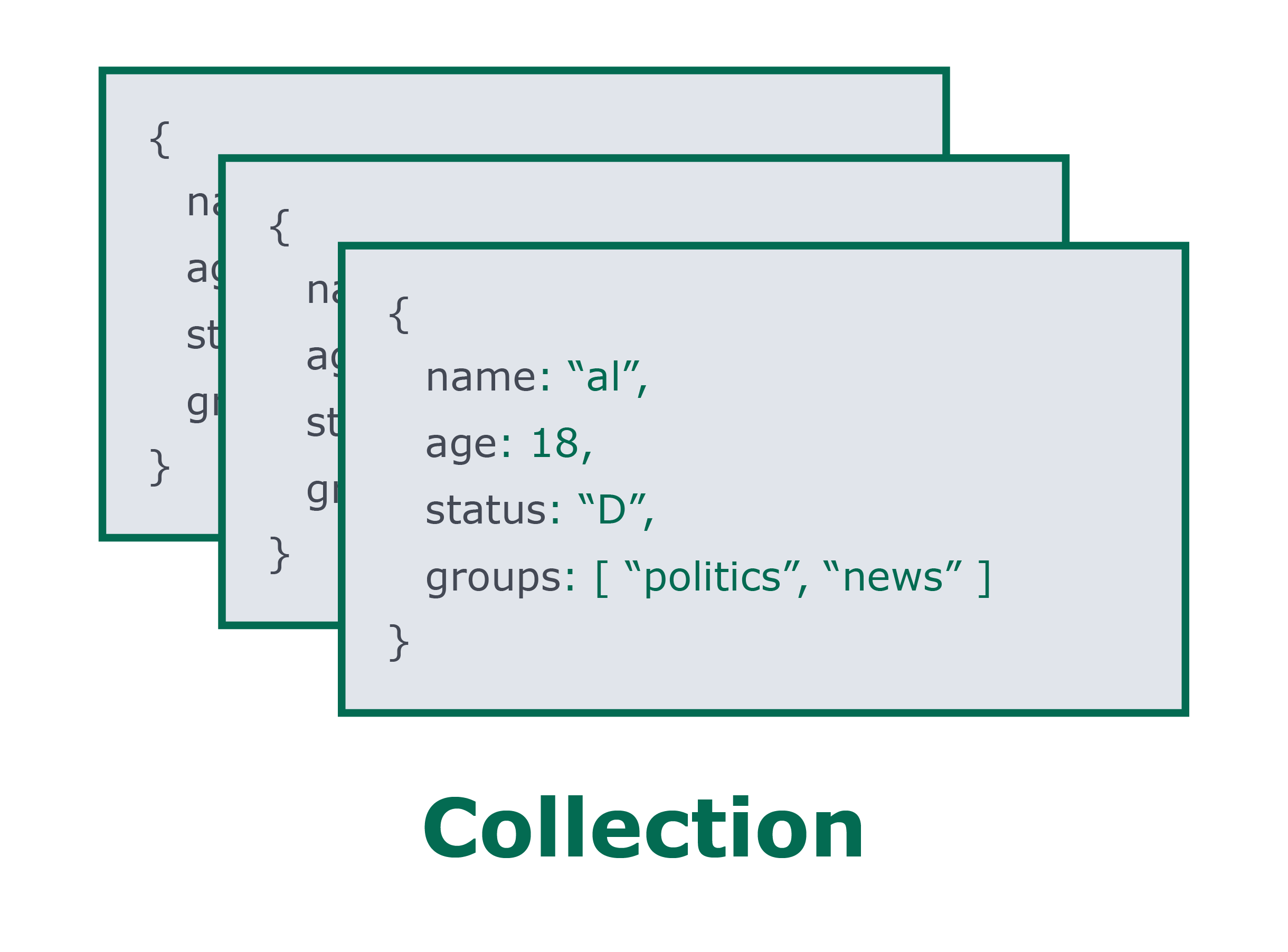
Por padrão, o MongoDB só permite conexões locais. Ou seja, apenas de *clients* que estejam rodando na mesma máquina onde a instância estiver sendo executada. Para alterar essa configuração e permitir conexões remotas, veja sobre [IP Binding](https://docs.mongodb.com/manual/core/security-mongodb-configuration/) na documentação.

### **Bancos de Dados, Coleções e Documentos**

A estrutura de armazenamento do MongoDB consiste em:

* ter diversos bancos de dados;
* dentro destes bancos temos as coleções (que seriam equivalentes às tabelas dos bancos de dados relacionais);
* dentro destas coleções temos os documentos (que seriam equivalentes aos registros dos bancos de dados relacionais).

O MongoDB armazena os documentos no formato BSON (Binary JSON). Entenda mais sobre esse formato [aqui](https://docs.mongodb.com/manual/core/document/#bson-document-format) .



#### Bancos de Dados

Assim como nos sistemas gerenciadores de bancos de dados relacionais, dentro de uma mesma instância do MongoDB você pode ter um ou vários bancos de dados. Uma grande diferença, é que não temos a formalidade de criar um banco de dados antes de fazer uma operação nele.

Por exemplo, quando vamos fazer um insert , o MongoDB cuida disso para você: criando o banco e a coleção (caso não existam previamente) juntos com o documento inserido. Tudo isso em uma mesma operação.

Uma vez conectado à uma instância do MongoDB através do MongoDB Shell , você só precisa especificar o contexto em que essa escrita acontecerá. Nesse caso, o contexto é o nome do banco de dados que você quer criar:

Copiar

use nomeDoBanco

db.nomeDaColecao.insertOne( { x: 1 })

Feito! A função insertOne() cria tanto o banco de dados nomeDoBanco , como a coleção nomeDaColecao , caso eles não existam. Se existirem, apenas mapeia o documento a ser inserido dentro deles e, por fim, executa a operação.

Uma dica para nomear bancos e coleções é seguir [este guia](https://docs.mongodb.com/manual/reference/limits/#restrictions-on-db-names) .

#### Coleções

Como citado anteriormente, os documentos no MongoDB são armazenados dentro das coleções . Lembrando que uma coleção é equivalente à uma tabela dos bancos de dados relacionais.

##### Criando uma coleção

Como você viu, se uma coleção não existe, o MongoDB cria essa coleção no momento do primeiro insert .

Copiar

db.nomeDaColecao1.insertOne({ x: 1 })

db.nomeDaColecao2.createIndex({ y: 1 })

Veja que tanto as operações insertOne() e createIndex() criam uma nova coleção (caso ela não exista).

##### Criação explícita

Você também pode utilizar o método db.createCollection() para criar uma coleção e especificar uma série de parâmetros, como o tamanho máximo do documento ou as regras de validação para os documentos .

Se você não quiser especificar algum desses parâmetros, o uso do método para criação não é necessário. O exemplo abaixo cria uma coleção, especificando sua [collation](https://docs.mongodb.com/manual/reference/collation/#collation-document-fields) .

Copiar

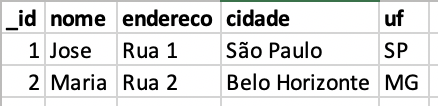
db.createCollection( "nomeDaColecao", { collation: { locale: "pt" } } );

Você pode fazer modificações nos parâmetros de uma coleção através do [collMod](https://docs.mongodb.com/manual/reference/command/collMod/#dbcmd.collMod) .

Você pode ver mais sobre o método db.createCollection() na [própria documentação](https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection) .

#### Documentos

Como dito, os documentos são equivalentes aos registros ou linhas de uma tabela nos bancos de dados relacionais. Além disso, cada atributo (campo) é equivalente a uma coluna de uma linha da tabela. Sua diferença é que documentos podem conter estruturas mais ricas, diferentes entre documentos, e armazenar muito mais informações do que você consegue em uma "linha simples" de uma tabela relacional. Abaixo, temos uma representação de um registro numa tabela relacional, e também o seu correspondente em um documento :



Copiar

{

"\_id": 1,

"nome": "Jose",

"endereco": {

"logradouro": "Rua 1",

"regiao": "Zona Norte",

"cidade": "São Paulo",

"uf": "SP"

}

},

{

"\_id": 2,

"nome": "Maria",

"endereco": {

"logradouro": "Rua 2",

"cidade": "Belo Horizonte",

"uf": "MG"

}

}

Como você viu acima, um insert recebe como parâmetro um JSON . Esse parâmetro define os dados e a estrutura do documento. É importante ressaltar que, por ser *schemaless* , ou seja, sem esquema por padrão, a estrutura não faz parte da coleção, e sim do documento . Com isso, você pode ter várias "estruturas" por coleção. No exemplo acima, podemos observar essa diferença entre os documentos. No primeiro, temos o atributo regiao , que não existe no segundo documento.

Quando você fizer uma alteração, faça-a em nível de documento. Pois caso você a faça em nível de coleção, muitos documentos com estruturas diferentes poderão ser impactados com a criação, alteração ou remoção de um atributo que não faz parte da estrutura de todos (veremos isso mais à frente).

##### Validação de documentos

Você pode aplicar uma validação para que cada operação de escrita em sua coleção respeite uma estrutura. Utilize o [Schema Validation](https://docs.mongodb.com/manual/core/schema-validation/) para isso.

##### BSON Types

Por mais que o insert ocorra recebendo um documento JSON , internamente, o MongoDB armazena os dados em um formato chamado BSON (ou Binary JSON ). Esse formato é uma extensão do JSON e permite que você tenha mais tipos de dados armazenados no MongoDB , não somente os tipos permitidos pelo JSON .

### **Insert**

Você vai fazer algumas inserções no MongoDB utilizando duas funções específicas e também com *schemas* diferentes e mais "ricos"!

Os métodos insertOne() e insertMany() têm suas particularidades e limitações. Enquanto um faz a inserção de um único documento por vez, o outro pode inserir milhares de documentos em uma única operação. Portanto, saber quando e onde aplicar fará toda a diferença quando você estiver codificando.

Os documentos mais "ricos" são aqueles de estruturas bem complexas que você pode armazenar em suas coleções, deixando os seus dados aninhados e com muitas informações, e sem a necessidade de joins para acessá-los em outras coleções.

Você já viu como os dados são armazenados no MongoDB , viu o conceito de bancos de dados, coleções e documentos. Agora você vai aprender como, efetivamente, criar todos esses elementos.

#### insertOne()

Veja este vídeo e entenda um pouco mais sobre o insertOne() .

Agora, crie um documento na coleção products , no banco de dados sample , com os seguintes atributos e valores:

Copiar

{

"productName": "Caixa",

"price": 20

}

Agora, assuma o controle do campo \_id , passando um valor qualquer para ele e crie um novo documento com os mesmos atributos e valores do documento anterior.

#### insertMany()

Veja este vídeo sobre o insertMany() .

Insira mais três documentos na coleção products em uma única operação:

Copiar

[

{ "productName": "Lapis", "stock": 10, "price": 20,"status":"A"},

{ "productName": "Tesoura", "price": 5, "status": "B" },

{ "productName": "Borracha", "price": 15, "status": "A" }

]

### **find()**

Após inserir documentos em seu banco de dados, você vai querer recuperá-los. Certo?

Assim como nos bancos de dados relacionais, no MongoDB temos um método específico para essa operação: o find() .

### **Parâmetros do find()**

O método find() serve para selecionar os documentos de uma coleção e retorna um cursor com esses documentos.

Esse método recebe dois parâmetros:

db.collection.find(query, projection)

* query (opcional):
  + Tipo: documento;
  + Descrição: especifica os filtros da seleção usando os *query operators* . Para retornar todos os documentos da coleção, é só omitir esse parâmetro ou passar um documento vazio ({}).
* projection (opcional):
  + Tipo: documento;
  + Descrição: especifica quais atributos serão retornados nos documentos selecionados pelo parâmetro *query* . Para retornar todos os atributos desses documentos, é só omitir esse parâmetro.

Esse método retorna um cursor para os documentos que correspondem aos critérios de consulta.

### **Projeção (projection)**

Como dito, o parâmetro projection determina quais atributos serão retornados dos documentos que atendam aos critérios de filtro. O formato recebido por ele é algo como:

Copiar

{ "atributo1": <valor>, "atributo2": <valor> ... }

O <valor> pode ser uma das seguintes opções:

* 1 ou true para incluir um campo nos documentos retornados;
* 0 ou false para excluir um campo;
* Uma expressão usando [Projection Operators](https://docs.mongodb.com/manual/reference/operator/projection/) .

Você pode escolher exibir no resultado da consulta apenas certos atributos.

A projeção é sempre o segundo parâmetro do método find() .

Veja só este exemplo:

Copiar

db.movies.insertOne(

{

"title" : "Forrest Gump",

"category" : [ "drama", "romance" ],

"imdb\_rating" : 8.8,

"filming\_locations" : [

{ "city" : "Savannah", "state" : "GA", "country" : "USA" },

{ "city" : "Monument Valley", "state" : "UT", "country" : "USA" },

{ "city" : "Los Anegeles", "state" : "CA", "country" : "USA" }

],

"box\_office" : {

"gross" : 557, "opening\_weekend" : 24, "budget" : 55

}

}

)

A operação acima insere um documento na coleção movies com vários atributos. Com a operação abaixo, selecionamos esse documento na coleção movies , passando como parâmetro de projeção apenas os atributos title e imdb\_rating :

Copiar

db.movies.findOne(

{ "title" : "Forrest Gump" },

{ "title" : 1, "imdb\_rating" : 1 }

)

Como resultado, teremos o seguinte:

Copiar

{

"\_id" : ObjectId("5515942d31117f52a5122353"),

"title" : "Forrest Gump",

"imdb\_rating" : 8.8

}

Note que o atributo \_id também foi retornado. Isso acontece porque ele é o único atributo que você não precisa especificar para que seja retornado. O movimento aqui é ao contrário, se você não quiser vê-lo no retorno, é só suprimí-lo da seguinte forma:

Copiar

db.movies.findOne(

{ "title" : "Forrest Gump" },

{ "title" : 1, "imdb\_rating" : 1, "\_id": 0 }

)

Agora sim, nosso resultado será apenas com os atributos devidos:

Copiar

{

"title" : "Forrest Gump",

"imdb\_rating" : 8.8

}

### **Gerenciamento do cursor**

Ao executar o método find() , o MongoDB Shell itera automaticamente o cursor para exibir os 20 primeiros documentos. Digite it para continuar a iteração. Assim, mais 20 documentos serão exibidos até o final do cursor.

Um método bastante interessante que é utilizado num cursor é o count() . O método count() retorna o número de documentos de uma coleção, e também pode receber um critério de seleção para retornar apenas o número de documentos que atendam a esse critério.

Você pode retornar o número de documentos de uma coleção com a seguinte operação:

Copiar

db.movies.count()

Veremos adiante mais utilizações para o método count() .

### **Tipos e comparações**

O MongoDB trata alguns tipos de dados como equivalentes para fins de comparação. Por exemplo, tipos numéricos sofrem conversão antes da comparação. No entanto, para a maioria dos tipos de dados, os [operadores de comparação](https://docs.mongodb.com/manual/reference/operator/query-comparison/) realizam comparações apenas em documentos em que o [tipo BSON](https://docs.mongodb.com/manual/reference/bson-type-comparison-order/#bson-types-comparison-order) do atributo destino do documento corresponde ao tipo do operando da query.

Para compreender melhor esse conceito, veja o exemplo abaixo, considerando a seguinte coleção:

Copiar

{ "\_id": "apples", "qty": 5 }

{ "\_id": "bananas", "qty": 7 }

{ "\_id": "oranges", "qty": { "in stock": 8, "ordered": 12 } }

{ "\_id": "avocados", "qty": "fourteen" }

A operação abaixo usa o operador de comparação $gt( *greater than* , maior que, >) para retornar os documentos em que o valor do atributo qty seja maior do que 4 :

Copiar

db.collection.find( { qty: { $gt: 4 } } )

A operação trará como retorno os seguintes documentos:

Copiar

{ "\_id": "apples", "qty": 5 }

{ "\_id": "bananas", "qty": 7 }

O documento com o \_id igual a "avocados" não foi retornado porque o valor do campo qty é do tipo string , enquanto o operador $gt é do tipo integer .

O documento com o \_id igual a "oranges" também não foi retornado porque qty é do tipo object .

Nesses casos, vemos o *schemaless* funcionando na prática!

### **Utilizando o find()**

#### Exemplos

Para esses exemplos você vai utilizar os documentos de uma coleção chamada [bios](https://docs.mongodb.com/manual/reference/bios-example-collection/) . Caso você queira executar os exemplos localmente, copie o trecho de código que representa a coleção, e execute no seu cliente do MongoDB para inserí-la em sua instância local.

Os documentos dessa coleção, de maneira geral, têm esse formato:

Copiar

{

"\_id" : <value>,

"name" : { "first" : <string>, "last" : <string> }, // documento embedado ou subdocumento

"birth" : <ISODate>,

"death" : <ISODate>,

"contribs" : [ <string>, ... ], // Array de Strings

"awards" : [

{ "award" : <string>, year: <number>, by: <string> } // Array de subdocumentos

...

]

}

### **Selecionando todos os documentos da coleção**

O método find() , quando utilizado sem parâmetros, retorna todos os documentos da coleção juntamente com todos os seus campos. Por exemplo, a operação abaixo retorna todos os documentos da coleção [bios](https://docs.mongodb.com/manual/reference/bios-example-collection/) :

Copiar

db.bios.find()

Essa operação corresponde à seguinte consulta no SQL :

Copiar

SELECT \* FROM bios;

### **Selecionando documentos com critérios de busca**

#### Query por igualdade

A operação abaixo retorna os documentos da coleção bios em que o atributo \_id é igual a 5 :

Copiar

db.bios.find( { \_id: 5 } )

Essa operação corresponde à seguinte consulta no SQL :

Copiar

SELECT \* FROM bios WHERE \_id = 5;

Agora, a operação abaixo retorna todos os documentos da coleção bios em que o campo last do subdocumento name é igual a "Hopper" :

Copiar

db.bios.find( { "name.last": "Hopper" } )

Note que, para acessar campos em subdocumentos, utilizamos [*dot notation*](https://docs.mongodb.com/manual/core/document/#document-dot-notation-embedded-fields) (por exemplo, "subdocumento.atributo" ).

### **Projetando somente os atributos requeridos:**

Através do segundo parâmetro do método find() , podemos especificar quais atributos serão retornados. O exemplo abaixo retorna todos os documentos da coleção bios , trazendo apenas o atributo name de cada documento:

Copiar

db.bios.find({}, { name: 1 })

Lembrando que o atributo name é um subdocumento , pois armazena um objeto com outros atributos.

Essa operação corresponde à seguinte consulta no SQL :

Copiar

SELECT name FROM bios;

Procure utilizar a projeção para diminuir a quantidade de campos retornados pelo cursor. Isso ajuda muito no que se refere ao tráfego desses dados na rede!

### **Limitando o número de documentos retornados**

#### limit()

Você pode limitar o número de documentos retornados por uma consulta utilizando o método limit() . Esse método é semelhante à declaração LIMIT em um banco de dados que utiliza SQL .

Uma utilização comum do limit() é para maximizar a performance e evitar que o MongoDB retorne mais resultados do que o necessário para o processamento.

O método limit() é utilizado da seguinte forma:

Copiar

db.collection.find(<query>).limit(<número>)

Note que você deve especificar um valor numérico no limit() .

Um exemplo utilizando a coleção bios :

Copiar

db.bios.find().limit(5)

A operação acima retornará os cinco primeiros documentos da coleção bios .

Essa operação corresponde à seguinte consulta no SQL :

Copiar

SELECT \* FROM bios LIMIT 5;

#### pretty()

Com o método pretty() você pode deixar os resultados exibidos no MongoDB Shell um pouco mais legíveis. Esse método aplica uma indentação na exibição dos resultados no console, de forma que fica bem melhor de ler.

Exemplo de utilização do método pretty() , usando a coleção bios :

Copiar

db.bios.find().limit(5).pretty()

Utilize o método pretty() à vontade!

### **"Pulando" documentos**

#### skip(<número>)

Acione o método skip() para controlar a partir de que ponto o MongoDB começará a retornar os resultados. Essa abordagem pode ser bastante útil para realizar paginação dos resultados.

O método skip() precisa de um parâmetro numérico que determinará quantos documentos serão "pulados" antes de começar a retornar.

O exemplo abaixo na coleção bios pulará os dois primeiros documentos e retornará o cursor a partir daí:

Copiar

db.bios.find().skip(2)

Você pode combinar os métodos limit() e skip() criando, assim, uma paginação:

Copiar

db.bios.find().limit(10).skip(5)

Essa operação corresponde à seguinte consulta no SQL :

Copiar

SELECT \* FROM bios LIMIT 10 OFFSET 5;

### 

### **Agora, a prática!**

O MongoDB possui diversas ferramentas como, por exemplo, mongo , mongosh , Compass e outras ferramentas de terceiros. Você pode utilizar o que achar melhor para executar as *queries* , o importante é realizá-las.

Utilizando a coleção [bios](https://docs.mongodb.com/manual/reference/bios-example-collection/) , construa queries para retornar os seguintes itens:

Exercício 1 : Retorne o documento com o \_id igual a 8.

Exercício 2 : Retorne o documento com o \_id igual a 8, mas só exiba os atributos: \_id e name .

Exercício 3 : Retorne apenas os atributos name e birth do documento com o \_id igual a 8.

Exercício 4 : Retorne todos os documentos em que o atributo name.first seja igual a John , utilizando o método pretty() .

Exercício 5 : Retorne os 3 primeiros documentos da coleção bios utilizando o método pretty() .

Exercício 6 : Retorne 2 documentos da coleção bios pulando os 5 primeiros documentos.

Utilizando o [mongoimport](https://docs.mongodb.com/database-tools/mongoimport/#examples) , importe o arquivo [books.json](https://s3.us-east-2.amazonaws.com/assets.app.betrybe.com/back-end/mongodb/books-48d15e4d8924badc2308cc4a62eb3ea4.json) para a sua instância local do MongoDB e utilize a coleção books para construir as seguintes consultas:

Exercício 7 : Retorne a quantidade de documentos da coleção books .

Exercício 8 : Conte quantos livros existem com o status = "PUBLISH" .

Exercício 9 : Exiba os atributos title , isbn e pageCount dos 3 primeiros livros. NÃO retorne o atributo \_id .

Exercício 10: Pule 5 documentos e exiba os atributos \_id , title , authors e status dos livros com o status = "MEAP" , limitando-se a 10 documentos.