Sumário

[Introdução 2](#_Toc22407141)

[O que é um banco de dados? 2](#_Toc22407142)

[Tipos de Banco de Dados 2](#_Toc22407143)

[Banco de dados Relacional (SQL) 2](#_Toc22407144)

[O que é? 2](#_Toc22407145)

[CRUD 3](#_Toc22407146)

[Quando usar? 3](#_Toc22407147)

[Banco de dados Não Relacional (NoSQL) 3](#_Toc22407148)

[O que é? 3](#_Toc22407149)

[Quando usar? 4](#_Toc22407150)

[Comparação SQL x NoSQL 4](#_Toc22407151)

[MongoDB 6](#_Toc22407152)

[O que é? 6](#_Toc22407153)

[Estruturas 6](#_Toc22407154)

[Database 6](#_Toc22407155)

[Collection 6](#_Toc22407156)

[Document 6](#_Toc22407157)

[Instalação 8](#_Toc22407158)

[Manipulando o MongoDB pela linha de comando 8](#_Toc22407159)

[Criando databases 8](#_Toc22407160)

[Importando dados 9](#_Toc22407161)

[Comandos Gerais 10](#_Toc22407162)

[Read 11](#_Toc22407163)

[Operadores 11](#_Toc22407164)

[Create 16](#_Toc22407165)

[Update 18](#_Toc22407166)

[Delete 19](#_Toc22407167)

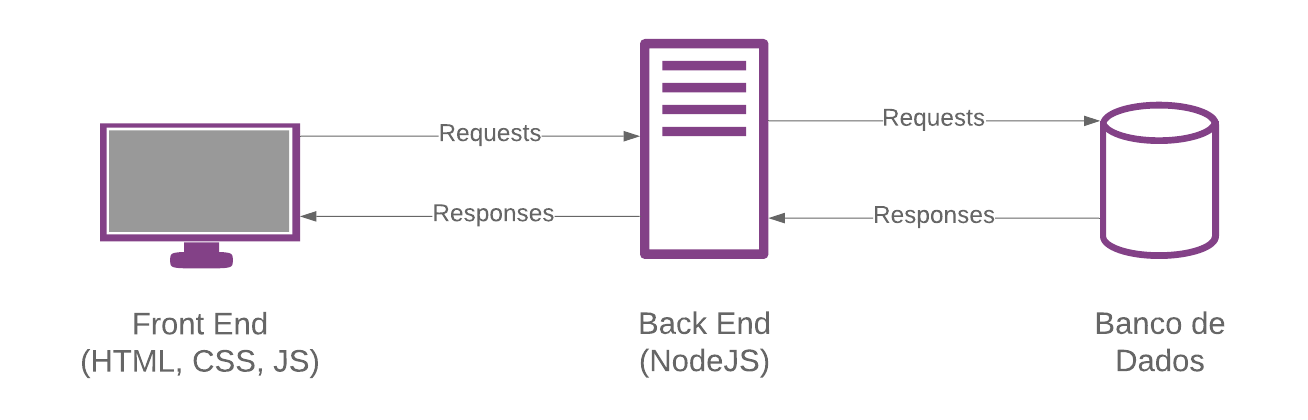
[Desafio 21](#_Toc22407168)

# Introdução

## O que é um banco de dados?

    É onde armazenamos informações sobre nosso mundo de forma organizada e fácil de se pesquisar. Podemos guardar informações sobre os usuários de nosso site ou aplicativo, como email, nome, telefone, etc. Para o caso de um aplicativo de banco de investimento, pode-se guardar informações como saldo da conta, transações realizadas (extrato), etc.

    Uma aplicação de banco de dados é um programa que vai rodar no seu servidor e vai acessar os dados que ele armazena na memória da máquina.

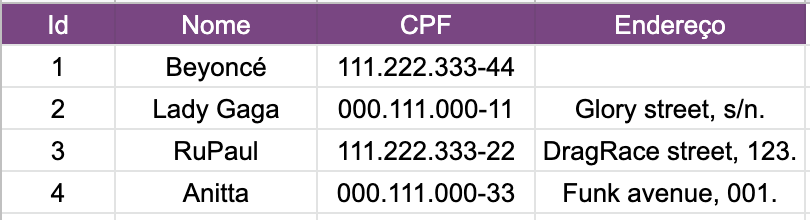


## Tipos de Banco de Dados

Existem formas diferentes de estruturar seu banco de dados, ele pode ser um banco de dados Relacional ou Não Relacional.

## Banco de dados Relacional (SQL)

### O que é?



É um banco de dados que modela os dados de forma que eles sejam percebidos como tabelas (relações). Uma tabela é uma estrutura simples de linhas (rows) e colunas (columns).

Ao criar o banco, você define um **esquema relacional**. Nesse esquema você diz quais tabelas vão ter e qual a relação entre elas. Cada tabela tem definida qual o tipo de cada coluna(number, string, datetime), e se ela é obrigatória ou opcional (nullable). Você pode adicionar mais tabelas depois.

Ver tabela em Excel para entendermos os conceitos.

Exemplos de bancos de dados SQL: [MySQL](https://dev.mysql.com/doc/), [PostgreSQL](https://www.postgresql.org/docs/), [Microsoft SQL Server](https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017).

### CRUD

    A forma de se manipular os dados de um banco de dados relacional é usando uma linguagem padrão chamada **SQL** (Structured Query Language). Com ela você pode inserir, remover, ou atualizar dados nas tabelas, e também realizar consultas. Uma consulta é uma pergunta ao banco de dados, também chamada de **query**. Essas operações são chamadas de **CRUD** (isso é genérico para todos os tipos de bancos de dados):

**C**reate: Criar um novo registro na tabela

**R**etrieve: Consultar, recuperar registros de uma ou mais tabelas (query)

**U**pdate: Atualizar o valor de um registro existente em uma tabela

**D**elete: Remover um registro da tabela

### Quando usar?

Para trabalhar com um banco relacional a primeira coisa que você precisa fazer é projetar a estrutura do banco, isto é, você não consegue inserir um dado se não tiver previamente definido os "esquemas" das tabelas. Isso é bom quando as questões de negócio bem definidas, e dá flexibilidade em fazer as consultas e relações entre os dados.

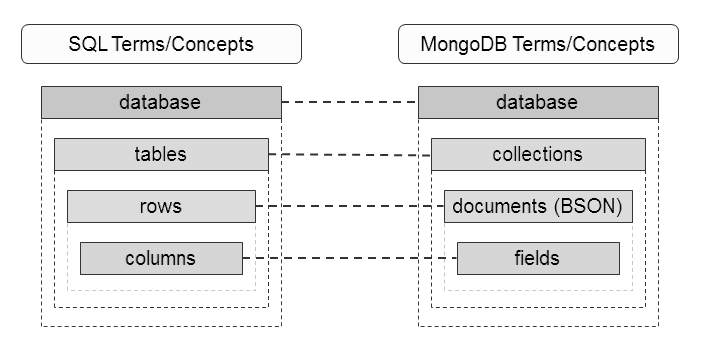
## Banco de dados Não Relacional (NoSQL)

### O que é?

    É um banco de dados que modela seus dados de forma que não sejam com tabelas, como os relacionais. Neles os dados podem ser armazenados de muitas formas diferentes, como Key-value (chave-valor), Objeto, Documento, e [outros](https://en.wikipedia.org/wiki/NoSQL). Vamos utilizar o **MongoDB** que é do tipo **Documento**.



    Aqui, no lugar de tabelas temos **coleções** (collections),cada coleção tem vários **documentos** (documents) ao invés de linhas, e no lugar de colunas temos simplesmente **campos** (fields).



    Exemplos de bancos de dados NoSQL: [MongoDB](https://docs.mongodb.com/), [Redis](https://redis.io/documentation), [outros](https://bigdata-madesimple.com/18-free-and-widely-used-open-source-nosql-databases/).

### Quando usar?

    São muito usados para aplicações de tempo real e também para big data. Como não precisa de uma estrutura pré-definida, dão flexibilidade e velocidade no desenvolvimento de aplicações simples. É também mais fácil de escalar.

## Comparação SQL x NoSQL

|  |  |  |
| --- | --- | --- |
|  | **Bancos de Dados SQL** | **Bancos de Dados NoSQL** |
| Tipos | Um só tipo (SQL) com pequenas variações | Vários tipos diferentes incluindo Chave-Valor, Documento, etc |
| Development History | Desenvolvido na década de 1970 para lidar com a primeira onda de aplicativos de armazenamento de dados | Desenvolvido no final dos anos 2000 para lidar com as limitações dos bancos de dados SQL, especialmente escalabilidade, dados multiestruturados, geodistribuição e sprints de desenvolvimento ágil |
| Exemplos | MySQL, Postgres, Microsoft SQL Server, Oracle Database | MongoDB, Cassandra, HBase, Redis |
| Data Storage Model | Registros individuais (por exemplo, "funcionários") são armazenados como linhas em tabelas, com cada coluna armazenando um dado específico sobre esse registro (por exemplo, "gerente", "data contratada", etc.), muito parecido com uma planilha. Os dados relacionados são armazenados em tabelas separadas e, em seguida, unidos quando consultas mais complexas são executadas. Por exemplo, 'escritórios' podem ser armazenados em uma tabela e 'funcionários' em outra. Quando um usuário deseja encontrar o endereço de trabalho de um funcionário, o mecanismo de banco de dados une as tabelas 'funcionário' e 'escritório' para obter todas as informações necessárias. | Varia de acordo com o tipo de banco de dados. Por exemplo, os bancos de dados de documentos acabam com o modelo de tabela e linha, armazenando todos os dados relevantes juntos em um único 'documento' em JSON, XML ou outro formato, que pode aninhar valores hierarquicamente. |
| Esquemas | Estrutura e tipos de dados são fixados antecipadamente. Para armazenar informações sobre um novo item de dados, o banco de dados inteiro deve ser alterado, durante o qual o banco de dados deve ser colocado offline. | Normalmente dinâmico, com algumas regras obrigatórias de validação de dados. Os aplicativos podem adicionar novos campos rapidamente e, ao contrário das linhas da tabela SQL, dados diferentes podem ser armazenados juntos, conforme necessário. |
| Escalando | Verticalmente, significa que um único servidor deve se tornar cada vez mais poderoso para lidar com o aumento da demanda. É possível distribuir bancos de dados SQL em muitos servidores, mas é necessária uma engenharia adicional significativa, e os recursos relacionais principais, como JOINs, integridade referencial e transações, são geralmente perdidos. | Horizontalmente, o que significa que, para adicionar capacidade, um administrador de banco de dados pode simplesmente adicionar mais servidores comuns ou instâncias de nuvem. O banco de dados automaticamente espalha os dados pelos servidores conforme necessário. |

    Referências:

<https://www.treinaweb.com.br/blog/sql-vs-nosql-qual-usar/>

<https://www.mongodb.com/nosql-explained?jmp=footer>

# MongoDB

## O que é?

    MongoDB é um banco de dados não relacional (NoSQL) do tipo Documento (document-base). Nesse tipo de banco de dados, cada registro é pensado como um documento, e todos os dados desse documento ficam armazenadas junto dele. Ele pode conter dados sem um esquema prévio, em que cada documento possui seu conjunto de chaves e valores independente dos outros documentos.

## Estruturas

### Database

    O database é seu banco de dados, que vai possuir um conjunto de collections.

### Collection

    Uma coleção (collection) é um conjunto de documentos, seria o equivalente a uma tabela em bancos SQL.

### Document

Um documento é como se fosse um objeto em JSON (javascript object notation) que você usa no seu código. Em outros bancos de dados, o documento pode estar em outros formatos também, como XML, mas no MongoDB utiliza somente JSON.

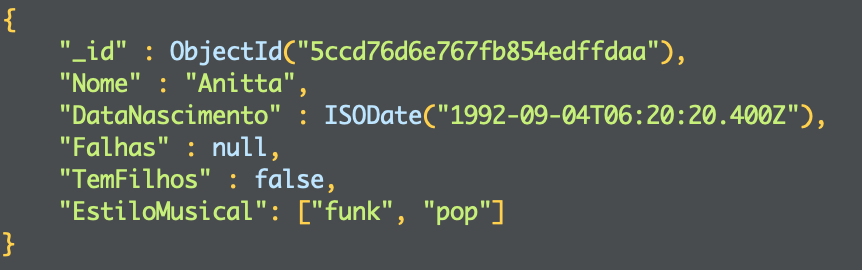
O JSON é uma estrutura de chaves (nome, dataNascimento, etc) e valores, que podem ser dos tipos: **string, number, boolean, null, object, array**.

O MongoDB armazena esses documentos JSON no formato **BSON** (Binary JSON), para ser mais eficiente na codificação e decodificação dos dados. O BSON também fornece alguns alguns tipos a mais, como **Date** e **ObjectId**.

Exemplo de um documento no MongoDB da coleção Diva:



Lembrando que dois documentos dentro da mesma coleção não precisam ter os mesmos campos. Exemplo de outro documento da mesma coleção Diva:



#### ObjectId

Cada documento no MongoDB possui um identificador único, que é gerado automaticamente e armazenado no campo **\_id**. Ele é um **ObjectId**, que contém dentro dele informações do timestamp da criação do documento, assim dá para ordenar seus documentos por esse campo e você não precisa ter um segundo campo como "DataCriação". Mais detalhes: [MongoDB ObjectId](https://www.tutorialspoint.com/mongodb/mongodb_objectid.htm)

# Instalação

Acesse o guia de instalação e siga o passo a passo.

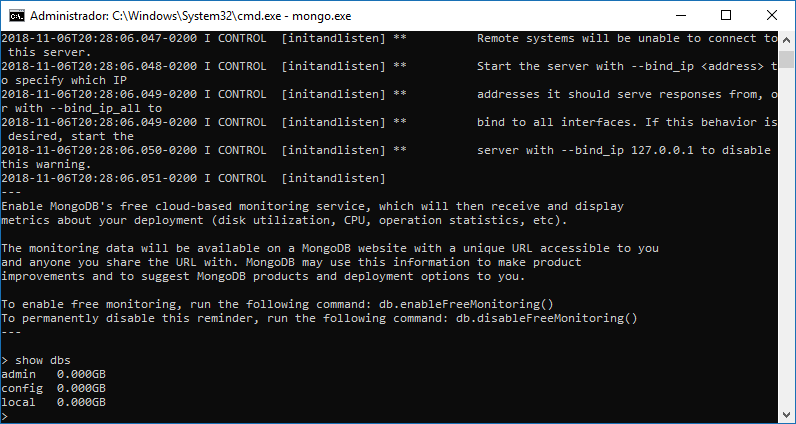
Ir até a pasta de instalação – *‘C:\Program Files\MongoDB\Server\4.0\bin’* e executar **mongod.exe**

O comando **mongod** faz com o que processo do banco de dados inicie.

O comando **mongo** faz com que você se conecte com uma instância específica do mongo.

Abra outro CMD/Prompt de Comando, entre na pasta: ‘C:\Program Files\MongoDB\Server\4.0\bin’, e execute **mongo.exe**

Por padrão, estes são os bancos criados:



## Manipulando o MongoDB pela linha de comando

<https://docs.mongodb.com/manual/crud/>

## Criando databases

**use** nomeDB

**show dbs** – não irá mostrar nenhum pois não tem nenhuma coleção inserida nele.

**show collections**

## Importando dados

**mongoimport** serve para importar dados de um determinado documento (Excel, JSON, CSV) para o banco de dados do mongodb. Abra um novo CMD e execute o comando abaixo:

**mongoimport --db todo -c tasks --type csv --file tasks.csv --headerline**

**mongoimport --db reprograma -c alunas --jsonArray --file alunas.json**

## Comandos Gerais

**show dbs** – mostrar todos os bancos de dados

**show collections** – mostrar todas as coleções

**use reprograma** – colocar o banco de dados em uso

**db.dropDatabase()** – deletar todo o banco de dados

**db.alunas.count()** – contar a quantidade de registros inserida

## Read

**db.alunas.findOne()** – seleciona o primeiro registro

**db.alunas.find()** – seleciona todos os registros da coleção selecionada

**db.alunas.find().pretty()** – busca todos os registros da coleção com uma melhor apresentação

**db.alunas.find().limit(2)** – busca os dois primeiros registros da coleção

#### Filtro

**db.alunas.find({ “nome” : “Helena” }).pretty() –** Realizando filtro. Busca apenas a aluna aonde o nome será igual a Helena. Ele é case sensitive. Isso significa que, caso você digite **Nome** no valor da chave, ele não irá ser encontrado.

Quando quer fazer uma busca encadeada por título, devemos colocar o ponto para identificar qual chave interna estamos buscando.

**db.alunas.find({ "livros.titulo" : "Clean Code A" }).pretty()**

**db.alunas.find({ "livros.leu" : true }).pretty()**

### Operadores

**Cuidado:** Quando são colocadas duas condições pro mesmo campo simultaneamente, ele irá considerar apenas o primeiro filtro da lista - **db.alunas.find({ "nome" : "Helena" }, { "nome" : "Aya" }).pretty()**

#### AND

**db.alunas.find({ "nome" : "Helena", "nasceuEmSp" : true }).pretty()**

**db.alunas.find({ $and: [{ "nome" : "Helena"}, {"ativa" : true }] }).pretty()**

#### OR

**db.alunas.find({ $or: [{ "nome" : "Helena"}, {"nome" : "Aya" }] }).pretty()**

#### IN

**db.alunas.find({ "nome" : { $in : ["Helena", "Aya"] } }).pretty()**

#### Exercícios

1. Selecione todos os registros.

**db.alunas.find()**

1. Selecione todos os registros e deixe a com apresentação melhor visível.

**db.alunas.find().pretty()**

1. Selecione todos os registros em que o nome seja igual a ‘Aya’.

**db.alunas.find({ "nome" : "Aya" }).pretty()**

1. Selecione todos os registros em que a aluna nasceu em São Paulo.

**db.alunas.find({ "nasceuEmSp" : true }).pretty()**

1. Selecione todos os registros em que contenha a título do livro igual a “Java - Web Developers”.

**db.alunas.find({ "livros.titulo" : "Java - Web Developers"}).pretty()**

1. Selecione todos os registros em que a aluna tenha o livro “Clean Code A” e o nome “Helena”.

**db.alunas.find({"livros.titulo" : "Clean Code A" , "nome" : “Helena” }).pretty()**

1. Selecione todos os registros em que a aluna não tenha nascido em sp e tenha o livro Cangaceiro JavaScript.  
   **db.alunas.find({"nasceuEmSp" : false , "livros.titulo" : "Cangaceiro JavaScript" }).pretty()**
2. Selecione todos os registros em que contenham a aluna tenha o livro Java – Web Developers ou Clean Code B. Primeiro: realize o exercício com o operador ou e posteriormente realize o exercício com o operador in.  
   **db.alunas.find({ $or: [{ "livros.titulo" : "Clean Code B"} , { "livros.titulo" : "Java - Web Developers" }] }).pretty()**

**db.alunas.find({ "livros.titulo" :{ $in: ["Clean Code B", "Java - Web Developers"] }}).pretty()**

1. Pegue o exercício acima e inclua também aonde a aluna pode ter nascido em sp.  
   **db.alunas.find({ $or: [{ "livros.titulo" : "Clean Code B"} , { "livros.titulo" : "Java - Web Developers" }, {“nasceuEmSp” : true}] }).pretty()**

Utilizando os dois operadores juntos. $or e $in.

1. Procure todas as alunas que tenham nascido em são paulo ou os livros ‘MySQL’ e ‘MongoDB’.

**db.alunas.find({**

**$or : [**

**{ "nasceuEmSp" : true },**

**{ "livros.titulo" : { $in : [ "MySQL", "MongoDB" ] } }**

**]**

**}).pretty()**

#### LIKE

**db.alunas.find({ "nome" : /X/ }).pretty() –** um nome que contenha um *h* em qualquer parte.

**db.alunas.find({ "nome" : /h/i }).pretty()**

**db.alunas.find({ "nome" : /a$/ }).pretty() –** um nome que termine com a letra *a*

**db.alunas.find({ "nome" : /^H/ }).pretty()** – um nome que comece com a letra *H*

**db.alunas.find({ "nome" : /^h/i }).pretty()**

i – ignoreCase - <https://www.w3schools.com/jsref/jsref_regexp_i.asp>

#### Exercícios

1. Selecione todos os registros em que os livros contenham em qualquer parte da palavra, 'Clean'.

**db.alunas.find({ "livros.titulo" : /clean/i }).pretty()**

1. Selecione todos os registros em que os títulos dos livros terminem com a letra 'S'.

**db.alunas.find({ "livros.titulo" : /s$/ }).pretty() – sem o ignore case**

**db.alunas.find({ "livros.titulo" : /s$/i }).pretty() – com o ignore case**

1. Selecione todos os registros em que todos os livros terminem com a letra 't'.

**db.alunas.find({ "livros.titulo" : /t$/ }).pretty()**

1. Selecione todos os nomes das alunas terminem com a letra 'k' ou 'o'.

**db.alunas.find({ "nome" : {$in: [/k$/, /o$/] }}).pretty()**

**db.alunas.find({ $or : [ { "nome" : /k$/ }, { "nome" : /o$/ } ] }).pretty()**

1. Selecione todos os registros em que os nomes comecem com a letra 'a'.

**db.alunas.find( { "nome" : /^a/i } ).pretty()**

1. Selecione todos os registros em que nomes comecem com a letra 'a' ou 'h'.

**db.alunas.find({ $or : [ { "nome" : /^a/i}, {"nome" : /^h/i } ] }).pretty()**

**db.alunas.find( { "nome" : { $in : [ /^h/i, /^a/i ] } } ).pretty()**

#### Ordenação

**db.alunas.find().sort({ \_id: 1}).pretty()** - ordena pelo campo **\_id** de forma crescente

**db.alunas.find().sort({ nome: 1}).pretty()**

**db.alunas.find().sort({ \_id: -1}).pretty()** - ordena pelo campo **\_id** de forma decrescente

**db.alunas.find().sort({ nome: -1}).pretty()**

**db.alunas.find().sort({ nome: -1}).limit(1).pretty()** - ordena pelo campo **nome** de forma crescente e pega o primeiro registro

**db.alunas.find().limit(1).pretty() –** seleciona somente o primeiro registro

## Create

**-- inserir apenas um registro na instrução**

db.alunas.insert(

{

"nome": "Helena A",

"dateOfBirth": "1993-03-18T08:30:00.000Z",

"nasceuEmSp": false,

"livros": [

{"titulo":"MySQL" , "leu" : true},

{"titulo": "MongoDB", "leu" :true},

{"titulo": "Java", "leu":true},

{"titulo": "Clean Architecture", "leu" :true}

]

}

)

**-- inserir vários registros em uma única instrução**

db.alunas.insertMany([

{

"nome": "Helena B",

"dateOfBirth": "1993-03-18T08:30:00.000Z",

"nasceuEmSp": true,

"livros": [

{"titulo":"C#" , "leu" : true},

{"titulo": "JavaScript", "leu" :true},

{"titulo": "Java", "leu":true},

{"titulo": "Clean Architecture", "leu" :true}

]

}

,

{

"nome": "Helena C",

"dateOfBirth": "1993-03-18T08:30:00.000Z",

"nasceuEmSp": false,

"livros": [

{"titulo":"NodeJS" , "leu" : true},

{"titulo": "Java", "leu":true},

{"titulo": "Clean Architecture", "leu" :true}

]

}

])

-- verificar se os registros foram inseridos corretamente.

**db.alunas.find({}).sort({ \_id : -1 }).limit(3).pretty()**

#### Exercícios

1. Crie 5 novos registros de alunas.

Deve conter os critérios abaixo. 3 em sp e 2 que não nasceram em sp.

Deve conter pelo menos alguns dos livros colocados: HTML, CSS, Design, JavaScript, Java, C#, MySQL, MongoDB.

A data de nascimento deve ser inferior a data atual.

## Update

Caso eu faça isso, sem settar os valores que eu quero no campo, o documento inteiro será atualizado com apenas o campo que foi passado.

db.alunas.update(

{ "nome" : "Helena C" },

{ "nasceuEmSp" : false }

)

**db.alunas.find({}).sort({ \_id : -1 }).limit(1).pretty()**

db.alunas.update(

{ "nome" : "Helena B" },

{

$set : { "nasceuEmSp" : true }

}

)

#### Exercícios

1. Atualize os registros criados no exercício anterior, com os filtros abaixo (não se esqueça de buscar pelo \_id):

Deixe todos os registros criados acima que estiverem como true para false, e vice-versa.

Atualize a data de nascimento de 2 alunas.

Inclua pelo menos 3 livros que elas tenham lido.

db.alunas.update(

{ "\_id" : ObjectId("5dab90e10d61146c420264aa") },

{ $push: { “livros”: { “titulo” : “Design”, “leu” : true } } }

## Delete

**db.alunas.remove({ "\_id" : ObjectId("5dab90e10d61146c420264aa") })**

**db.alunas.remove({ "\_id" : ObjectId("5dab90e10d61146c420264ab") })**

#### Exercícios

1. Exclua 5 itens dada uma determinada condição.

Exemplo:

Por ordenação de inserção, os dois primeiros que foram inseridos.

Por ordem alfabética, os dois primeiros.

Por ordem de signo decrescente, o primeiro signo.

**Desafio Extra**

Por ordem de habilidades, a que possui o maior registro de habilidades.

db.alunas.aggregate([

{

$project: {

item: 1,

numeroDeHabilidades: { $cond: { if: { $isArray: "$habilidades" }, then: { $size: "$habilidades" }, else: "NA"} }

}

}

] )

Verificar o registro que possui a maior quantidade de habilidades.

db.alunas.find({ \_id : ObjectId("5d811ade31c542921b14417b") }).sort({ \_id : -1 }).limit(3).pretty()

Com ordenação

db.alunas.aggregate([

{

$project: {

item: 1,

numeroDeHabilidades: { $cond: { if: { $isArray: "$habilidades" }, then: { $size: "$habilidades" }, else: "NA"} }

}

},

{ $sort: { numeroDeHabilidades: 1 } }

] )

## Desafio

Você receberá um arquivo com o formato JSON contendo alguns personagens de quadrinhos e/ou desenhos animados e você deverá realizar as alterações que foram solicitadas abaixo pelo cliente.

1. Selecione todos os registros.
   1. **db.personagens.find().pretty()**
2. Selecione apenas o primeiro registro.
   1. **db.personagens.findOne()**
   2. **db.personagens.find().sort({\_id:1}).limit(1).pretty()**
3. Selecione todos os registros em que o nome seja igual a ‘Carla’ ou a quantidade de gatos seja igual a 2.
   1. **db.personagens.find( {$or : [ {"Nome" : "Carla" }, {"Qtd Gatos" : 2} ]}).pretty()**
4. Selecione todos os registros em que o local de nascimento seja igual a Roma e necessariamente que a quantidade de cachorros seja igual a 3.
   1. **db.personagens.find( {"Local" : "Roma", "Qtd Cachorros" : 3} ).pretty()**
5. Selecione todos os registros em que a quantidade de gatos seja igual a 3 ou a quantidade de cachorros seja igual a 6.
   1. **db.personagens.find({ $or: [ {"Qtd Gatos" : 3}, { "Qtd Cachorros" : 6 } ] }).pretty()**
6. Selecione todos os registros em que o nome comece com C.
   1. **db.personagens.find({ "Nome" : /^C/ }).pretty()**
7. Selecione todos os registros em que o nome comece com H.
   1. **db.personagens.find({ "Nome" : /^H/ }).pretty()**
8. Selecione todos os registros em que o nome termine com a.
   1. **db.personagens.find({ "Nome" : /a$/ }).pretty()**
9. Selecione todos os registros em que o nome contenha s.
   1. **db.personagens.find({ "Nome" : /s/ }).pretty()**
10. Selecione todos os registros em que o nome contenha ‘e’ ou ‘o’.
    1. **db.personagens.find( { $or : [{ "Nome" : /e/}, { "Nome": /o/ } ]}).pretty()**
11. Insira 3 novos super-heróis.
12. Delete 1 super-herói em que o nome seja igual a Thor.
13. Atualize a quantidade de gatos em que o usuário com o registro de nome ‘Thor’ possui. Atualize a quantidade de gatos de ‘0’ para ‘7’.
14. Selecione todos os registros que foram trabalhados e exporte para um arquivo JSON.