

# Introdução a Banco de Dados

<b>Introdução</b>	<b>3</b>
O que é um banco de dados?	3
<b>Tipos de Banco de Dados</b>	<b>3</b>
Banco de dados Relacional (SQL)	3
O que é?	3
CRUD	4
Quando usar?	4
Exercício	4
Banco de dados Não Relacional (NoSQL)	5
O que é?	5
Quando usar?	6
Comparação SQL x NoSQL	6
<b>MongoDB</b>	<b>8</b>
O que é?	8
Estruturas	8
Database	8
Collection	8
Document	9
ObjectId	9
Instalação	11
Manipulando o MongoDB com Robo 3T	12
Manipulando o MongoDB pela linha de comando	12
Importando dados	12
Comandos Gerais	12
Create	13
Exercícios	13
Read	13
Filtro	13
Operadores	14
AND	14
OR	14
IN	14
Exercícios	14
LIKE	14
Exercícios	15

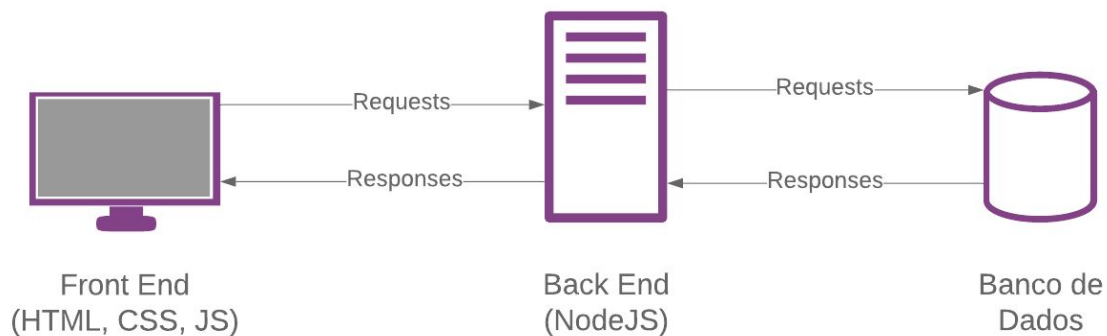
\$gt e \$lt	15
Exercícios	15
Ordenação	15
Update	15
Exercícios	16
Delete	16
Exercícios	16
<b>Desafio</b>	16

# 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?

Id	Nome	CPF	Endereço
1	Beyoncé	111.222.333-44	
2	Lady Gaga	000.111.000-11	Glory street, s/n.
3	RuPaul	111.222.333-22	DragRace street, 123.
4	Anitta	000.111.000-33	Funk avenue, 001.

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: [Tabela Excel](#)

Exemplos de bancos de dados SQL: [MySQL](#), [PostgreSQL](#), [Microsoft SQL Server](#).

## 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):

**Create:** Criar um novo registro na tabela

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

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

**Delete:** 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.

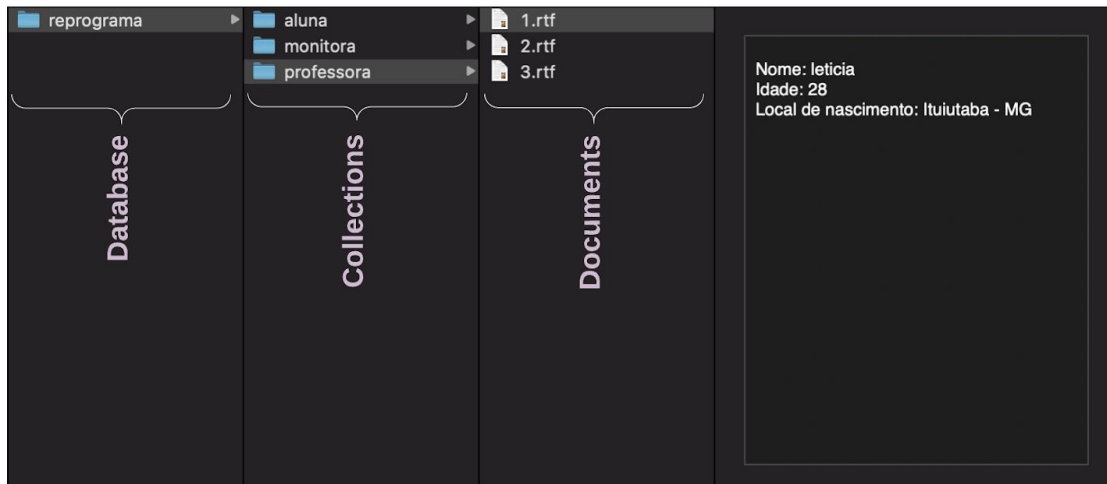
## Exercício

1. Insira uma nova usuária na tabela com seu nome.
2. Qual o relacionamento existente entre as duas tabelas?
3. Qual o saldo atual na conta da Beyoncé?
4. Insira uma nova transação na conta da Beyoncé.
5. Filtros - exibir somente usuárias que comecem com a letra L
6. Filtros - exibir somente usuárias que não possuem endereço preenchido

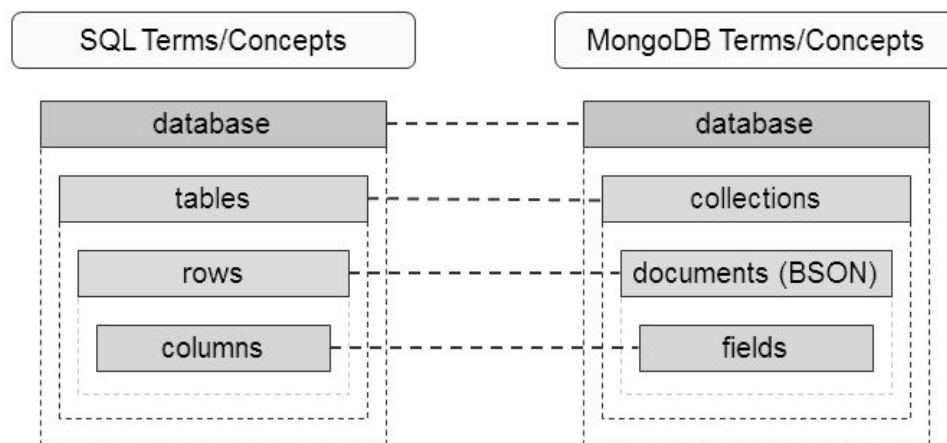
## Banco de dados Não Relacional (NoSQL)

### 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](#). 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](#), [Redis](#), [outros](#).

## 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	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

	<p>(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.</p>	<p>relevantes juntos em um único 'documento' em JSON, XML ou outro formato, que pode aninhar valores hierarquicamente.</p>
Esquemas	<p>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.</p>	<p>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.</p>
Escalando	<p>Verticalmente, significa que um único servidor deve se</p>	<p>Horizontalmente, o que significa que, para adicionar</p>

	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.	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-oriented database). 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 tipos a mais, como **Date** e **ObjectId**.

Exemplo de um documento no MongoDB da coleção Diva:

```
{
  "_id" : ObjectId("5ccd7305e767fb854edffd51"),
  "Nome" : "Beyoncé",
  "DataNascimento" : ISODate("1981-09-04T06:20:20.400Z"),
  "Falhas" : null,
  "TemFilhos" : true,
  "QuantidadeFilhos" : 3,
  "Albuns" : [
    "Homecoming",
    "Lemonade",
    "Beyoncé",
    "4"
  ],
  "Endereço" : {
    "Principal" : "Rua das divas",
    "Complemento" : "sem número",
    "CEP" : "00102-03"
  }
}
```

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

```
{
  "_id" : ObjectId("5ccd76d6e767fb854edffdaa"),
  "Nome" : "Anitta",
  "DataNascimento" : ISODate("1992-09-04T06:20:20.400Z"),
  "Falhas" : null,
  "TemFilhos" : false,
  "EstiloMusical": ["funk", "pop"]
}
```

## 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](#)

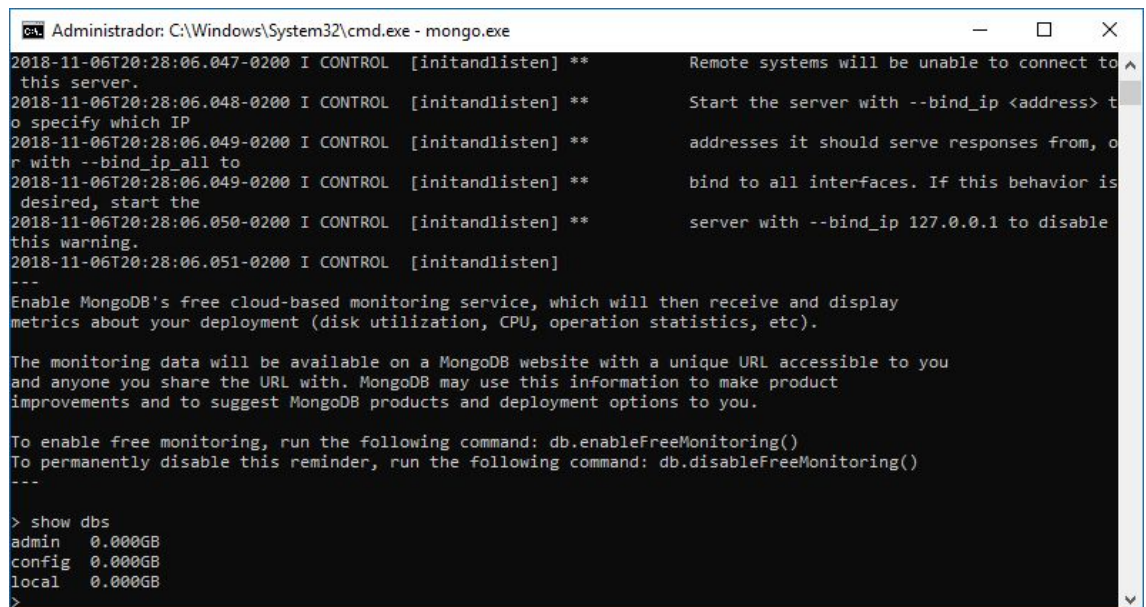
## Instalação

1. Acesse o [guia de instalação](#) e siga o passo a passo.
2. 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.

3. Abra outro CMD/Prompt de Comando, entre na pasta: ‘C:\Program Files\MongoDB\Server\4.0\bin’, e execute **mongo.exe**
4. Por padrão, estes são os bancos criados:



```
Administrador: C:\Windows\System32\cmd.exe - mongo.exe
2018-11-06T20:28:06.047-0200 I CONTROL [initandlisten] ** Remote systems will be unable to connect to
this server.
2018-11-06T20:28:06.048-0200 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> t
o specify which IP
2018-11-06T20:28:06.049-0200 I CONTROL [initandlisten] ** addresses it should serve responses from, o
r with --bind_ip_all to
2018-11-06T20:28:06.049-0200 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is
desired, start the
2018-11-06T20:28:06.050-0200 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable
this warning.
2018-11-06T20:28:06.051-0200 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>
```

## Manipulando o MongoDB com Robo 3T

Vamos começar a mexer com o MongoDB na ferramenta gratuita Robo 3T. Para instalar, acesse o [guia de instalação](#) e siga as instruções.

1. Criar uma conexão com seu MongoDB.
2. Criar um **Database** chamado **reprograma**.
3. Criar uma **Collection** chamada **aluna**.
4. CRUD
  - 4.1. **C** - Inserir um documento com informações sobre você (nome, localNascimento, profissao)
  - 4.2. **C** - Inserir 3 documentos com informações sobre outras alunas.
  - 4.3. **R** - Consultar todos os documentos
  - 4.4. **U** - Atualizar seu documento, inserindo um outro dado pessoal.
  - 4.5. **R** - Contar quantos documentos tem pela linha de comando
  - 4.6. **R** - Buscar somente o seu documento pelo campo **nome** — (preencher o **.find()** na linha de comando)
  - 4.7. **R** - Contar quantos documentos tem pela linha de comando com seu Nome
  - 4.8. **D** - Deletar todos os documentos

## Manipulando o MongoDB pela linha de comando

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

### 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 reprograma -c alunas --type csv --file alunas.csv --headerline
```

### 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.aluna.count()** – contar a quantidade de registros inserida

## Create

```
db.aluna.insert({ "nome" : "Nome MongoDB", "localNascimento" : "São Paulo" })
```

```
db.aluna.insertMany([ { "nome" : "Nome 1", "localNascimento" : "São Paulo" }, { "nome" :  
"Nome 2", "localNascimento" : "Minas Gerais" } ])
```

## Exercícios

1. Insira 1 novo registro conforme descrição abaixo:

nome = Ada Lovelace, localNascimento = Londres, profissao = Matemática, anoNascimento = 1815

2. Insira 4 novos registros **de uma só vez** conforme descrições abaixo:

nome = Grace Hopper, localNascimento = Nova York, profissao = Matemática, anoNascimento = 1906

nome = Jean E. Sammet, localNascimento = Nova York, profissao = Cientista da Computação, anoNascimento = 1928

nome = Karen Sparck Jones, localNascimento = Willingham, profissao = Cientista da Computação, anoNascimento = 1935

nome = Carol Shaw, localNascimento = Palo Alto, profissao = Engenheira de Software, anoNascimento = 1955

## Read

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

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

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

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

## Filtro

**db.aluna.find({ "nome" : "Ada Lovelace" }).pretty()** – Realizando filtro. Busca apenas a aluna cujo **nome** seja igual a **Ada Lovelace**. Ele é case sensitive. Isso significa que, caso você digite **Nome** no valor da chave, o mesmo não irá ser encontrado.

## Operadores

**Cuidado:** Quando são colocadas duas condições pro mesmo campo simultaneamente, ele irá considerar apenas o primeiro filtro da lista - `db.aluna.find({ "Nome" : "Ada Lovelace" }, { "Nome" : "Grace Hopper" }).pretty()`

### AND

`db.aluna.find({ "nome" : "Nome A", "profissão" : "Profissão X" }).pretty()`

### OR

`db.aluna.find({ $or: [{ "nome" : "Nome A"}, {"nome" : "Nome B" } ] }).pretty()`

### IN

`db.aluna.find({ "nome" : { $in : ["Nome A", "Nome B" ] } }).pretty()`

## Exercícios

1. Selecione todos os registros.
2. Selecione todos os registros e deixe a com apresentação melhor.
3. Selecione todos os registros em que o nome seja igual a 'Ada Lovelace'.
4. Selecione todos os registros em que o nome seja igual a 'Grace Hopper'.
5. Selecione todos os registros em que a profissão seja 'Matemática'.
6. Selecione todos os registros onde o nome seja igual a 'Ada Lovelace' e a profissão seja 'Matemática'. Depois, a profissão seja 'Engenheira de Software' e o nome 'Carol Shaw'.
7. Selecione todos os registros em que o nome seja igual a 'Ada Lovelace' OU 'Grace Hopper'.
8. Pegue o exercício acima e inclua: onde o localNascimento seja igual 'Londres'.
9. Trabalhe com um operador **diferente de 'OU'** para trazer todos os registros onde as profissões sejam iguais a 'Matemática' ou 'Cientista da Computação'.
10. **Utilizando os dois operadores juntos. \$or e \$in.**  
Procure todas as alunas cujo nome seja igual a 'Ada Lovelace' ou a profissão seja igual a 'Cientista da Computação' ou 'Matemática'.

### LIKE

`db.aluna.find({ "nome" : /e/ }).pretty()` – um nome que contenha **e** em qualquer parte.

`db.aluna.find({ "nome" : /e$/ }).pretty()` – um nome que termine com **e**.

`db.aluna.find({ "nome" : /^e/ }).pretty()` – um nome que comece com **e**.

## Exercícios

1. Selecione todos os registros em que os nomes contenham em qualquer parte da palavra a letra 'p'.
2. Selecione todos os registros em que os nomes terminem com a letra 's'.
3. Selecione todos os registros em que os nomes terminem com a letra 'w'.
4. Selecione todos os registros em que os nomes terminem com a letra 's' ou 'w'.
5. Selecione todos os registros em que os nomes comecem com a letra 'A'.
6. Selecione todos os registros em que os nomes comecem com a letra 'K' ou 'J'.

## \$gt e \$lt

**db.aluna.find({ "anoNascimento" : { \$gt : 2018 } }).pretty()** - **gt** significa greater than - maior que

**db.aluna.find({ "anoNascimento" : { \$lt : 2019 } }).pretty()** - **lt** significa less than - menor que

## Exercícios

Traga somente alunas em que o ano de nascimento seja superior a 1900 e inferior a 1930.

## Ordenação

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

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

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

## Update

Para atualizar campos do registro:

```
db.aluna.update(  
  { "nome" : "Nome A" }, // filtro para escolher o registro  
  {  
    $set : { "idade" : 120, "profissao": "desenvolvedora" } // campos a serem  
    atualizados  
  }  
)
```

**Cuidado:** caso eu faça o update, sem dizer no **\$set** os campos que eu quero atualizar, o documento inteiro será atualizado com apenas o campo que foi passado.

```
db.aluna.update(  
  { "nome" : "Nome A" },  
  { "idade" : 120 }  
)
```

## Exercícios

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

nome = Ada Lovelace Maravilhosa, localNascimento = Londres - UK

nome = Grace Hopper Inspiração, localNascimento = Nova York - EUA

nome = Jean E. Sammet Fantástica, localNascimento = Nova York - EUA

nome = Karen Sparck Jones Fodona, localNascimento = Willingham - UK

nome = Carol Shaw Gamer Mara, localNascimento = Palo Alto - EUA

## Delete

```
db.aluna.remove({ "_id" : ObjectId("5bed5ca7f3bed37a5d90e63f") })
```

## Exercícios

1. Exclua os últimos 5 itens criados e atualizados nos itens anteriores com as seguintes condições:
  - a. Faça o filtro pelo `_id` no item abaixo:
    - i. Ada Lovelace
    - ii. Grace Hopper
  - b. Faça o filtro pelo `localNascimento` no item abaixo:
    - i. Karen Sparck Jones
  - c. Faça o filtro pela profissão, nos itens abaixo:
    - i. Jean E. Sammet
    - ii. Carol Shaw

## Desafio

Você receberá um arquivo com o formato CSV 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.
2. Selecione apenas o primeiro registro.
3. Selecione todos os registros em que o nome seja igual a 'Carla' ou a quantidade de gatos seja igual a 2.
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.



5. Selecione todos os registros em que a quantidade de gatos seja igual a 3 ou a quantidade de cachorros seja igual a 6.
6. Selecione todos os registros em que o nome comece com C.
7. Selecione todos os registros em que o nome comece com H.
8. Selecione todos os registros em que o nome termine com a.
9. Selecione todos os registros em que o nome contenha s.
10. Selecione todos os registros em que o nome contenha 'e' ou 'o'.
11. Selecione todos os registros em que a ordem seja maior do que 10.
12. Selecione todos os registros em que a quantidade de gatos seja maior do que 2 ou menor do que
13. Insira 3 novos super-heróis.
14. Delete 1 super-herói em que o nome seja igual a Thor.
15. Atualize a quantidade de gatos em que o usuário com o registro de nome 'Carla' possui. Atualize a quantidade de gatos de '0' para '7'.
16. Selecione todos os registros que foram trabalhados e exporte para um arquivo JSON.