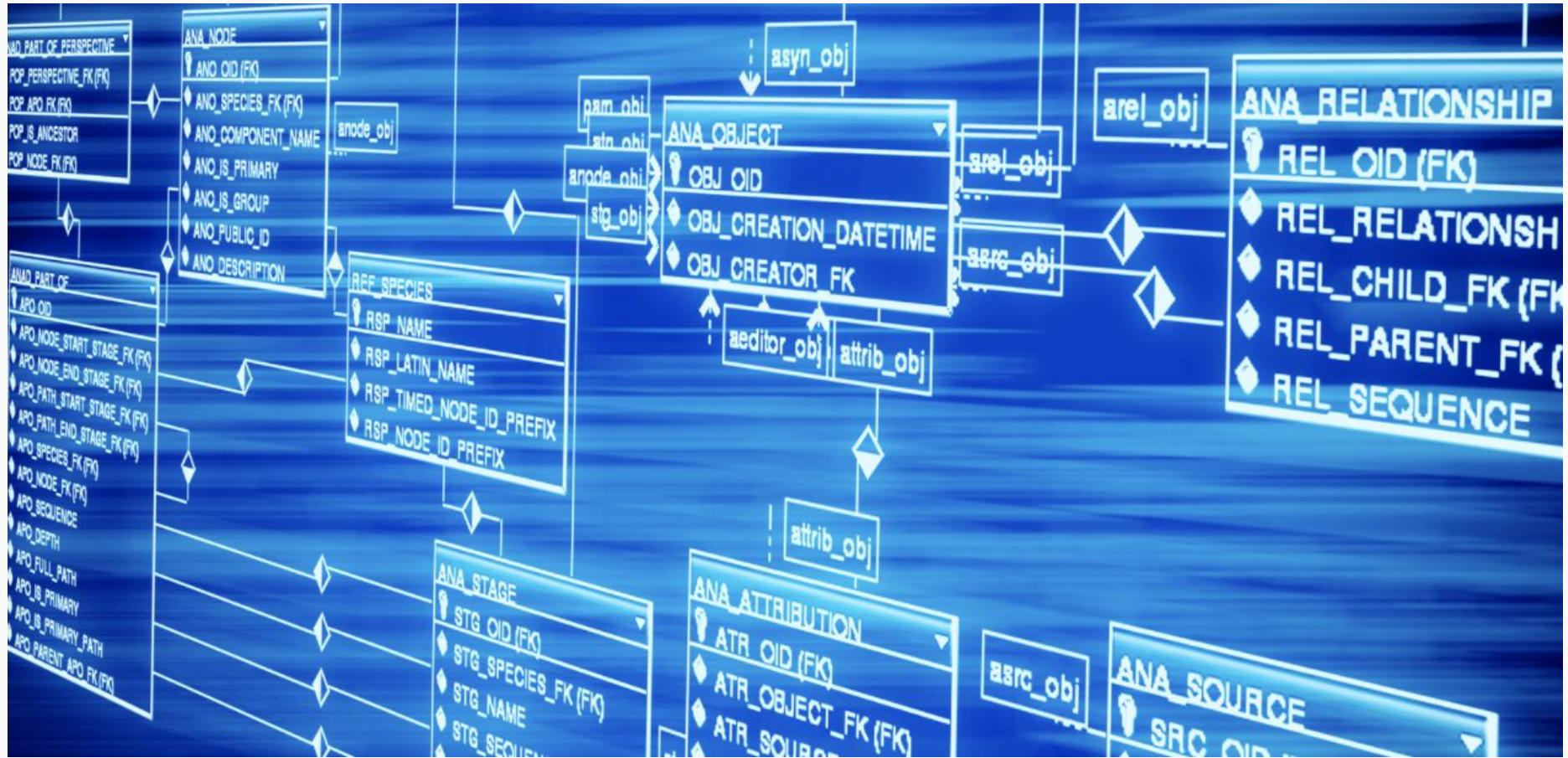


Introdução ao Banco de dados

Reprograma - Backend

Prof^a: Kelly Joany de Oliveira Santos

O que é um banco de dados?



O que é um banco de dados?

Bancos de dados são programas que nos permitem salvar e recuperar dados para nossos aplicativos.

Os bancos de dados são tipicamente persistentes, o que significa que mesmo se nosso programa parar/reiniciar, ainda podemos acessar os dados.

Além da persistência, os sistemas de banco de dados fornecem uma série de outras propriedades que os tornam excepcionalmente úteis e convenientes:

- Confiabilidade: os dados sempre podem ser acessados.
- Eficiência: Poderíamos apenas usar arquivos para armazenar os dados, mas essa solução será lenta para qualquer programa sério.
- Escalabilidade: conforme nossas demandas para obter mais e mais dados aumentam, os bancos de dados também tornam mais fácil aumentar nossa capacidade de infraestrutura.

Além da persistência, os sistemas de banco de dados fornecem uma série de outras propriedades que os tornam excepcionalmente úteis e convenientes:

- Simultaneidade: Podemos ter muitos clientes conectados ao nosso banco de dados (programa) simultaneamente.
- Abstrações de dados: podemos armazenar dados usando tipos de dados complexos que tornam mais fácil salvar os dados sem nos preocupar com os detalhes subjacentes da implementação.
- Linguagem de consulta de alto nível: os bancos de dados têm uma linguagem na qual podemos fazer perguntas para obter os dados de qualquer maneira que possamos precisar.

Banco de dados relacionais

Um banco de dados relacional organiza os dados em tabelas ou relações. Uma tabela é composta de linhas e colunas. As linhas também são chamadas de registros ou tuplas. As colunas também são chamadas de campo ou atributo. Uma tabela de banco de dados é semelhante a uma folha de cálculo.

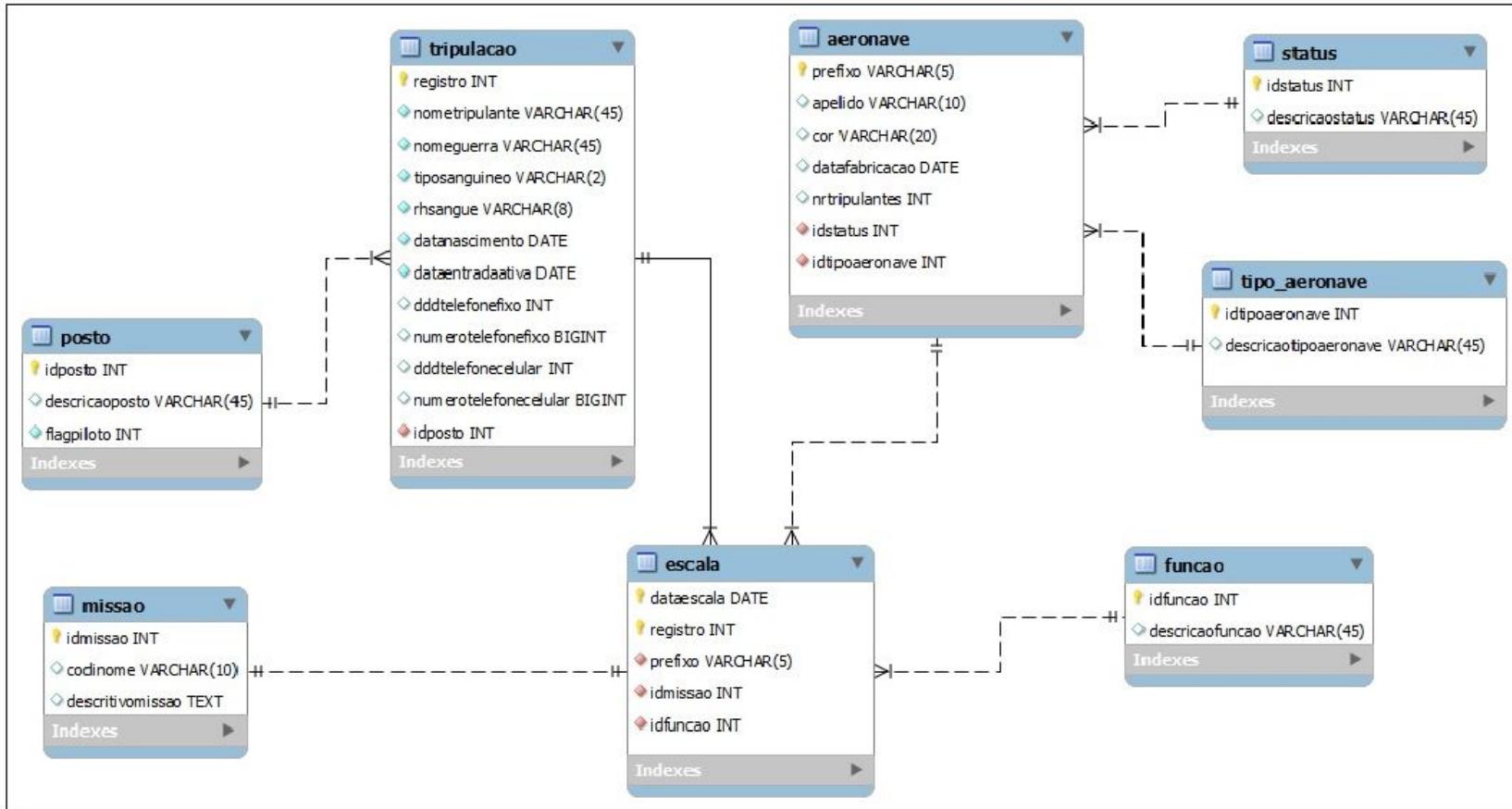
Outra característica importante deste modelo é o uso de elementos para garantir a integridade dos dados. As restrições mais comuns são as chaves primárias e as estrangeiras.

Banco de dados relacionais

O termo chave primária é utilizado para identificar o atributo que foi escolhido pelo projetista do banco para identificar unicamente os registros que são armazenados em uma determinada tabela.

Nenhum registro pode ter o mesmo valor no campo chave primária de uma determinada tabela do banco de dados, por isso os atributos chaves devem ser selecionados com muito cuidado. Já a chave estrangeira transforma o valor de um atributo dependente do valor existente em outro atributo, normalmente de outra tabela, criando uma relação de dependência entre atributos de tabelas distintas.

Banco de dados relacionais



Banco de dados relacionais

MySQL Workbench

Local instance MySQL57

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

gonodemodulo2

Tables

- appointments
- sequelizemeta
- users

Query 1

users appointments users

SELECT * FROM gonodemodulo2.users;

Result Grid

| | id | name | email | avatar | password_hash | provider | created_at | updated_at |
|---|------|-------|----------------------|--------------------------------------|---|----------|---------------------|---------------------|
| 1 | 1 | teste | teste@mail.com | teste.ioo | \$2a\$08\$okVmD8G5OMeELIRseuhdE.E2v8JvIdL... | 1 | 2019-03-04 13:15:24 | 2019-03-04 13:15:24 |
| 2 | 2 | kelly | kelly.ioanv@mail.com | teste.ioo | \$2a\$08\$druijh9kPQGvOz8ovvoW.skfoYnb4/h... | 1 | 2019-03-04 13:21:04 | 2019-03-04 13:21:04 |
| 3 | 3 | maria | maria@mail.com | 0edd57fc6215c28db79b80385e0a703c.ioo | \$2a\$08\$sa4VsAL/dBb/zNRsO02jOkFJhfSVJ/fy... | 0 | 2019-03-05 12:23:22 | 2019-03-05 12:23:22 |
| 4 | 4 | allan | allan@mail.com | a23a3c56b65a49fc3af7f0fe7b43b4ea.ioo | \$2a\$08\$W9H1RoalDVId1KeJvF9IOvkDXxki3IY... | 1 | 2019-03-07 01:23:33 | 2019-03-07 01:23:33 |
| | HULL | HULL | HULL | NULL | NULL | NULL | NULL | NULL |

Banco de dados não relacionais

Não seguem uma estrutura de tabelas e cada banco pode apresentar uma maneira diferenciada para navegar em seus dados. Existem diversos modos para organizar os dados.

- **Chave valor:** é precisamente o que o nome descreve você tem uma chave e um valor, o uso geralmente é para os casos onde você faz buscas somente pela chave, não sendo possível buscar pelo valor, casos de uso desse modelo geralmente são armazenamento de sessões de usuários e carrinhos de compras, exemplificando, você tem o id do usuário que é a chave e os itens do carrinho ou as informações da sessão do usuário. Bancos: Dynamo, Riak.
- **Documento:** é um tipo de chave valor, a grande diferença é que você tem várias chaves e valores e consegue buscar por ambos. Os casos de uso deste modelo são logs de eventos, gerenciamento de conteúdo, blogs e afins. Bancos: MongoDB, Apache CouchDB.

Banco de dados não relacionais

The screenshot shows the AWS DynamoDB console interface. On the left, the navigation menu is visible with options like Painel, Tabelas, Backups, Capacidade reservada, Preferências, DAX, Painel, Clusters, Grupos de sub-redes, Parameter groups, and Eventos. The 'Tabelas' option is selected. In the center, the 'Heroes' table is displayed under the 'Itens' tab. A search bar at the top allows filtering by table name. Below it, there are buttons for 'Criar item' and 'Ações'. A verification message 'Verificar: [Tabela] Heroes: id, nome' is present. The main area shows a table with columns: id, nome, createdAt, and poder. The data is as follows:

| | id | nome | createdAt | poder |
|--|--------------------------------------|-------------|--------------------------|--------------|
| | 09952dd0-0e31-11eb-9694-db6db9d839d2 | Flash | 2020-10-14T15:22:14.189Z | Velocidade |
| | 0a7b3a50-0e2c-11eb-94a8-f7f6ae7ef8dd | Flash | 2020-10-14T14:46:28.213Z | Velocidade |
| | 0b276c40-0a70-11eb-8813-2d0feb566189 | Batman | 2020-10-09T20:43:10.469Z | Rico |
| | 0dc51be0-0e31-11eb-80ca-a9ec48602865 | Flash | 2020-10-14T15:22:21.215Z | Velocidade |
| | 13e0f930-0ee6-11eb-9427-b3ac9ce38e9c | Flash | 2020-10-15T12:58:10.371Z | Velocidade |
| | 1c9a47e0-0a70-11eb-8813-2d0feb566189 | Flash | 2020-10-09T20:43:39.742Z | Velocidade |
| | 2b601120-0e2f-11eb-b3ac-adfd3d565308 | Flash | 2020-10-14T15:08:51.890Z | Velocidade |
| | 37ba1030-0e2d-11eb-a1ce-3f370ead616e | Flash | 2020-10-14T14:54:53.619Z | Velocidade |
| | 450d5a50-0a70-11eb-9e71-8bf9e7ae51e0 | Flash | 2020-10-09T20:44:47.605Z | Velocidade |
| | 470243a0-0e2d-11eb-8400-7f6050423569 | Flash | 2020-10-14T14:55:19.258Z | Velocidade |

Banco de dados não relacionais

```
{  
    _id : <ObjectId1>,  
    name : "abc",  
    contact : {  
        phone : "9012398755",  
        email : "abc@test.com"  
    },  
    address : {  
        address : "plot 21,virat nagar",  
        city : "xy"  
    }  
}
```

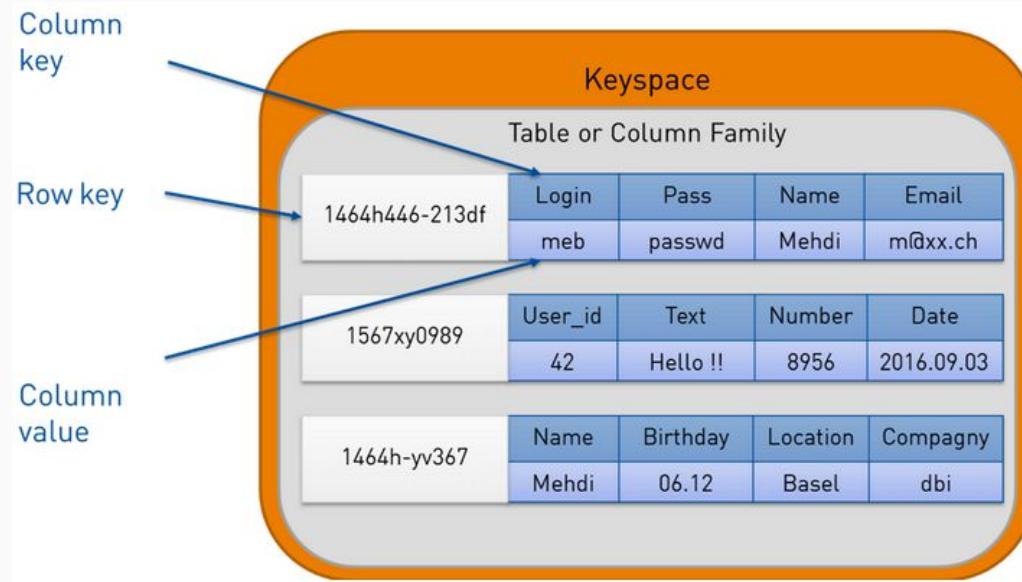
Embedded
Document

Banco de dados não relacionais

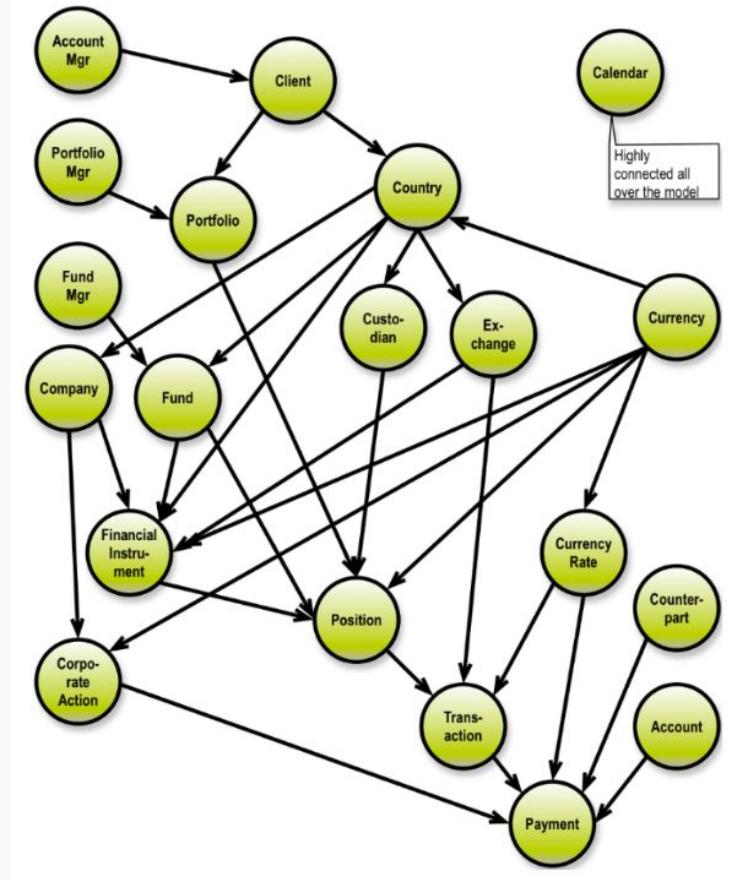
- **Coluna:** A ideia aqui é parecida com um banco relacional, somente a ideia de ter colunas, então você vai ter colunas e linhas quase parecido com um banco relacional, casos de uso são praticamente os mesmos de banco de dados via documento, o que diferencia um do outro é que aqui você tem que criar a "tabela" com as colunas enquanto no banco de dados de documento não. Bancos Cassandra, HBase.
- **Grafo:** este é válido para modelos de dados onde é importante o relacionamento entre eles e o relacionamento é complexo, por exemplo você quer saber se o João e a Maria foram no restaurante Y. Os casos de uso mais comuns para este modelo são redes sociais e serviços baseados em localização. Bancos: Apache Giraph, Neo4J.

Banco de dados não relacionais

| Orientado a Linhas | Orientado a Colunas |
|--|---|
| Joao 2432.00 1988 Rio de Janeiro | Joao Maria Pedro Jorge |
| Maria 2511.00 1986 São Paulo | 2432.00 2511.00 3500.00 4200.00 |
| Pedro 3500.00 1976 Mato Grosso | 1988 1986 1976 1930 |
| Jorge 4200.00 1930 Paraná | Rio de Janeiro São Paulo Mato Grosso Paraná |



Banco de dados não relacionais

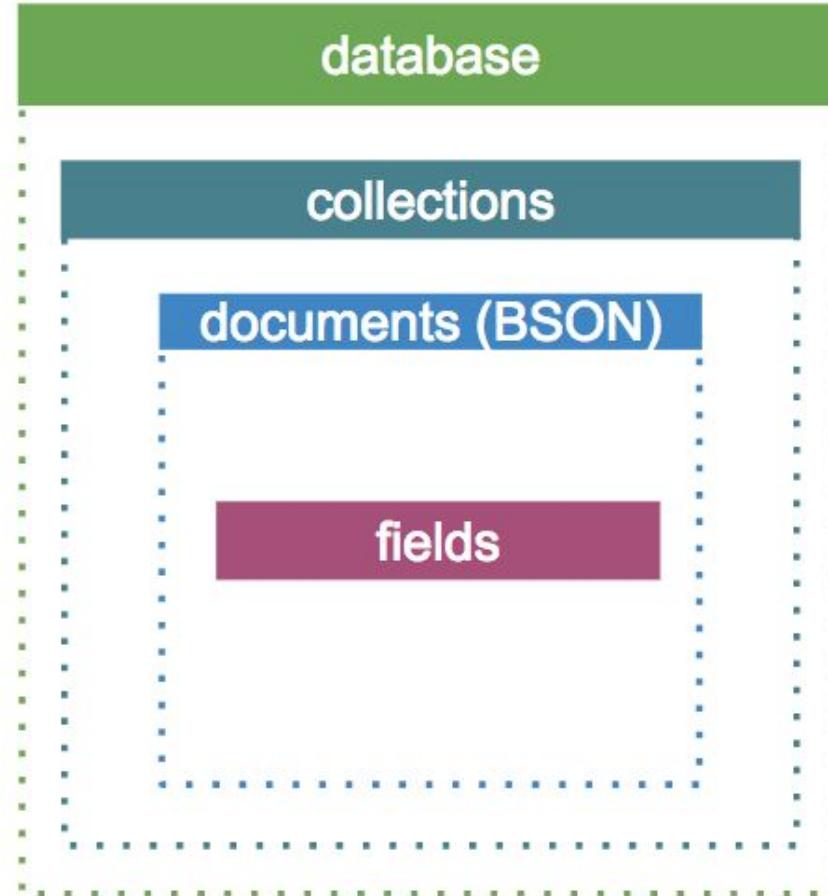


Introdução ao MongoDB

MongoDB é um programa de banco de dados orientado a documentos de plataforma cruzada gratuito e de código aberto desenvolvido pela MongoDB Inc.

MongoDB é um banco de dados não relacional que armazena dados em coleções. Ele armazena dados em campos flexíveis do tipo JSON, o que significa que podem variar de documento para documento e, com o tempo, a estrutura de dados pode mudar.

Introdução ao MongoDB



Instalação do MongoDB

Acessar o link: <https://www.mongodb.com/download-center/community/releases>

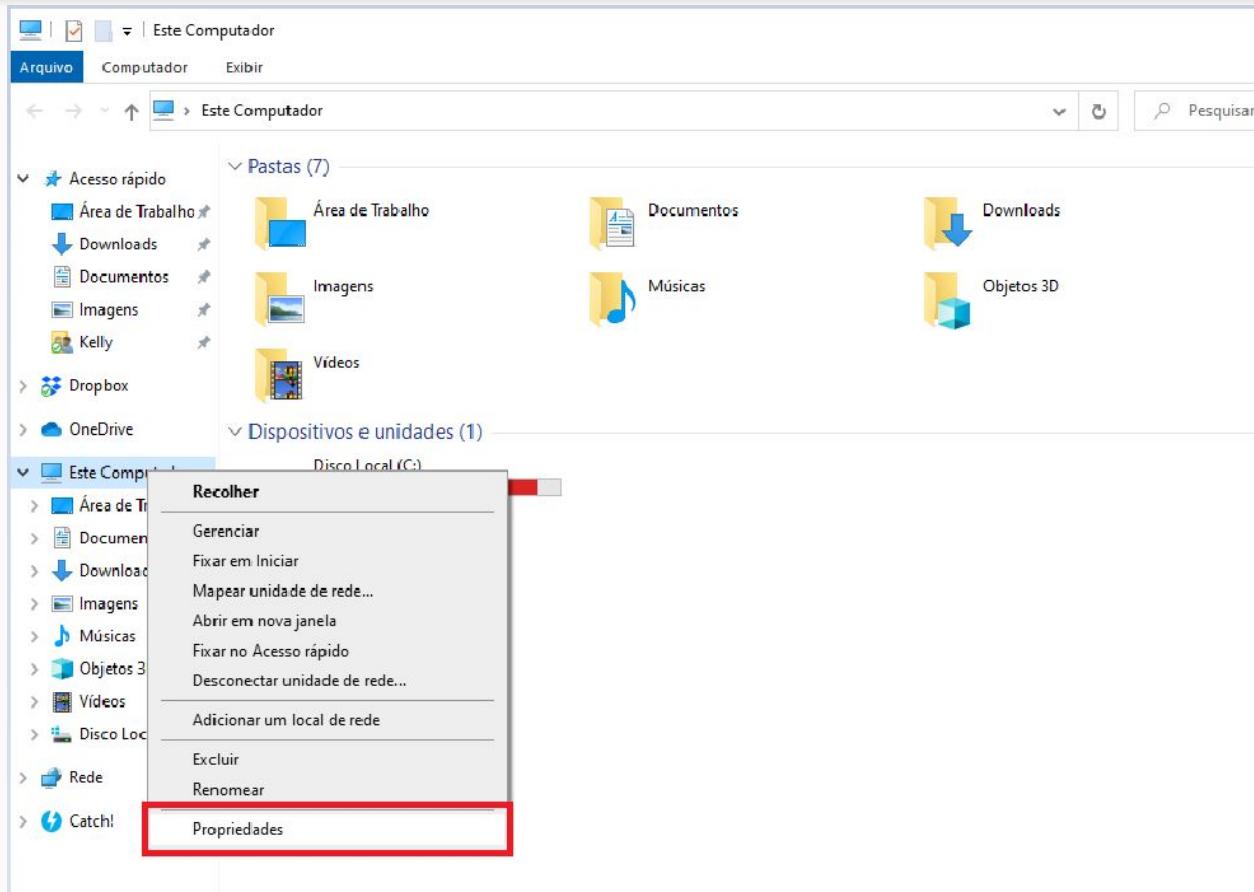
- Windows x64
 - Archive: [mongodb-windows-x86_64-4.4.1.zip](#)

Criar uma pasta no C:\MongoDB

Descompactar os arquivos e mover os arquivos para a pasta recém criada.

Criar uma pasta no C:\data\db

Configuração do MongoDB



Configuração do MongoDB

Sistema

Início do Painel de Controle Exibir informações básicas sobre o computador

Gerenciador de Dispositivos

Configurações remotas

Proteção do sistema

Configurações avançadas do sistema

Edição do Windows

Windows 10 Pro

© 2019 Microsoft Corporation. Todos os direitos reservados.

Windows 10

Sistema

Processador: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz

Memória instalada (RAM): 16,0 GB (utilizável: 15,9 GB)

Tipo de sistema: Sistema Operacional de 64 bits, processador com base em x64

Caneta e Toque: Nenhuma Entrada à Caneta ou por Toque está disponível para este vídeo

Nome do computador, domínio e configurações de grupo de trabalho

Nome do computador: Avell

Nome completo do computador: Avell

Descrição do computador: Avell

Grupo de trabalho: WORKGROUP

Alterar configurações

Ativação do Windows

Windows ativado Ler os Termos de Licença para Software Microsoft

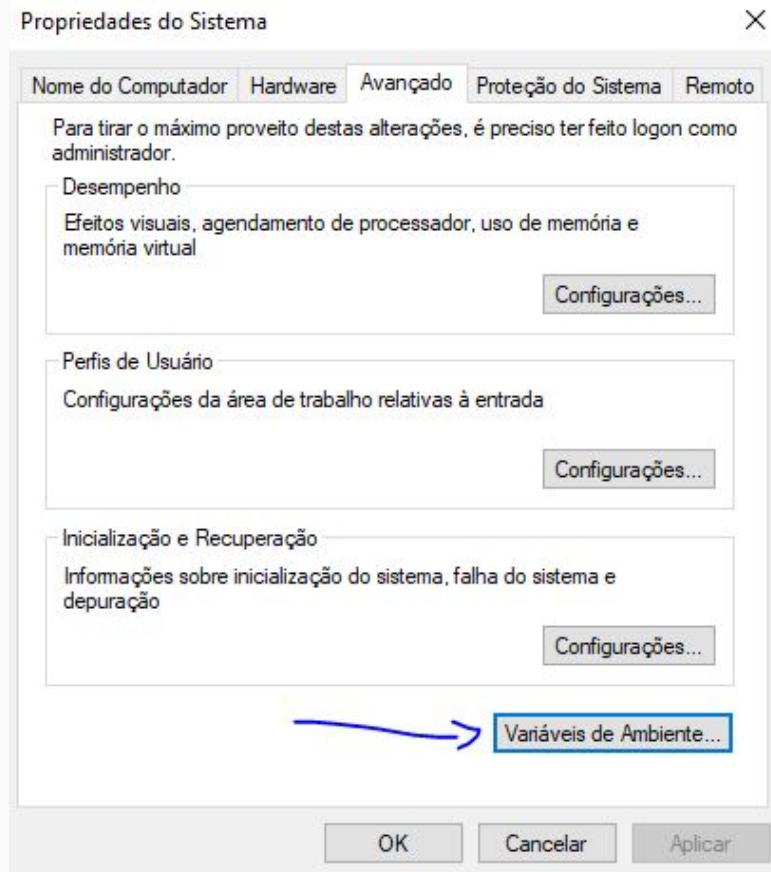
ID do Produto (Product ID): 00331-10000-00001-AA229

Alterar chave do produto (Product Key)

Consulte também

Segurança e Manutenção

Configuração do MongoDB



Variáveis de Ambiente

Variáveis de usuário para Avell

| Variável | Valor |
|--------------------------|--|
| ChocolateyLastPathUpdate | 132464781386433449 |
| OneDrive | C:\Users\Avell\OneDrive |
| Path | C:\Users\Avell\AppData\Local\Microsoft\WindowsApps;C:\Users\A... |
| TEMP | C:\Users\Avell\AppData\Local\Temp |
| TMP | C:\Users\Avell\AppData\Local\Temp |

[Novo...](#) [Editar...](#) [Excluir](#)

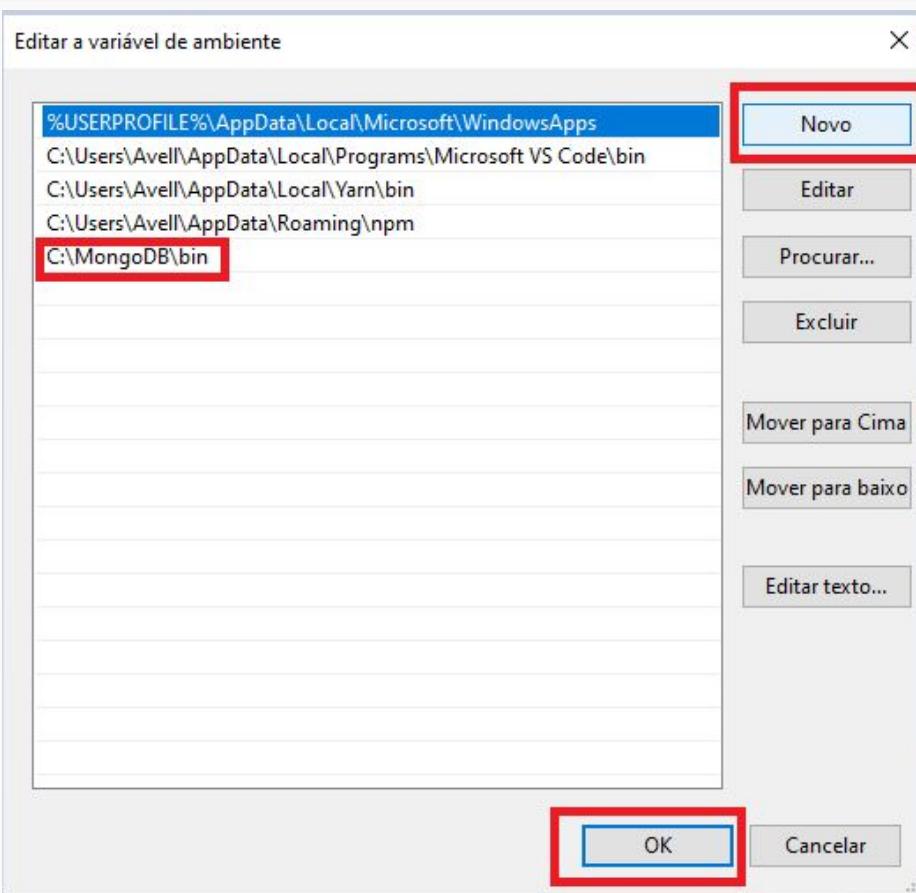
Variáveis do sistema

| Variável | Valor |
|-------------------|--|
| ANDROID_HOME | C:\Users\Avell\AppData\Local\Android\Sdk |
| ChocolateyInstall | C:\ProgramData\chocolatey |
| CLASSPATH | : |
| ComSpec | C:\WINDOWS\system32\cmd.exe |
| DriverData | C:\Windows\System32\Drivers\DriverData |
| JAVA_HOME | C:\Program Files\Java\jdk1.8.0_201 |
| MSMPI_BIN | C:\Program Files\Microsoft MPI\Bin\ |

[Novo...](#) [Editar...](#) [Excluir](#)

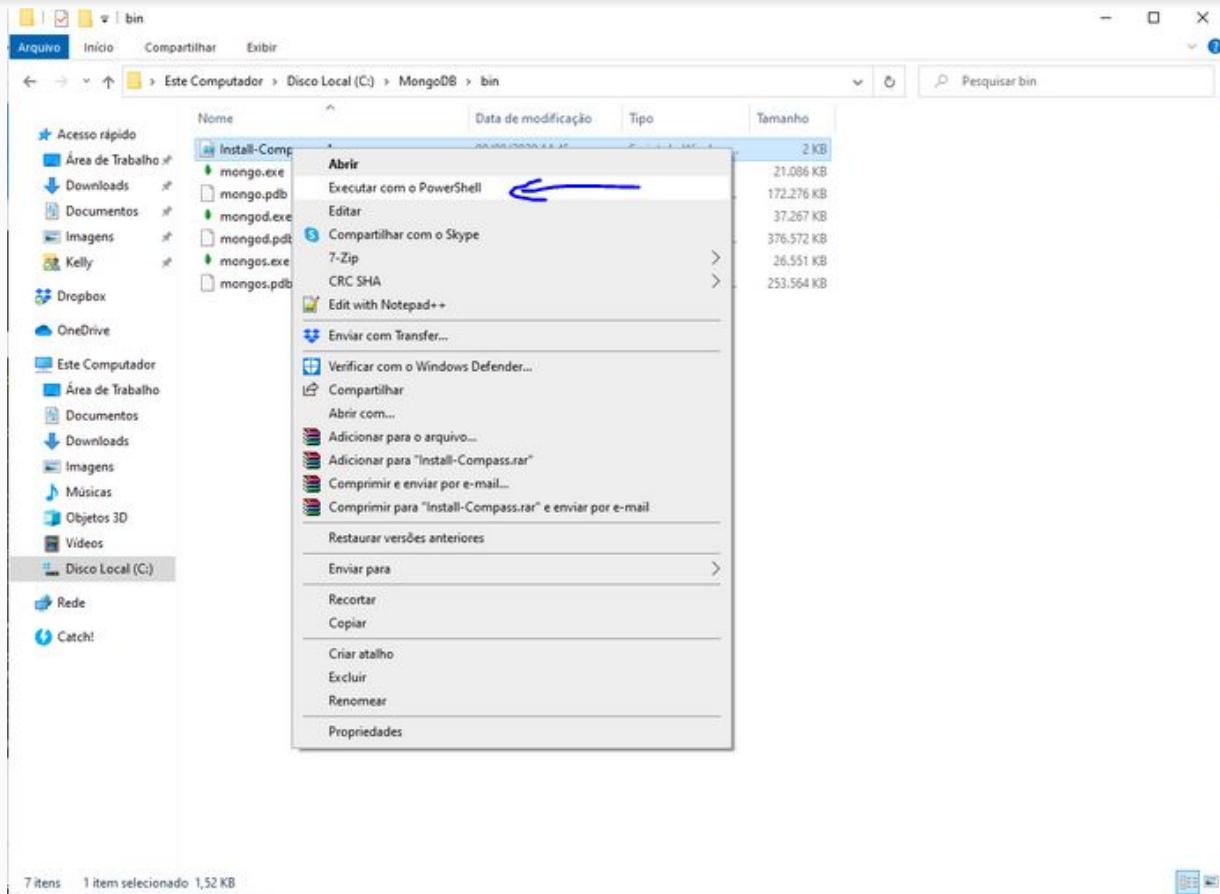
OK Cancelar

Configuração do MongoDB

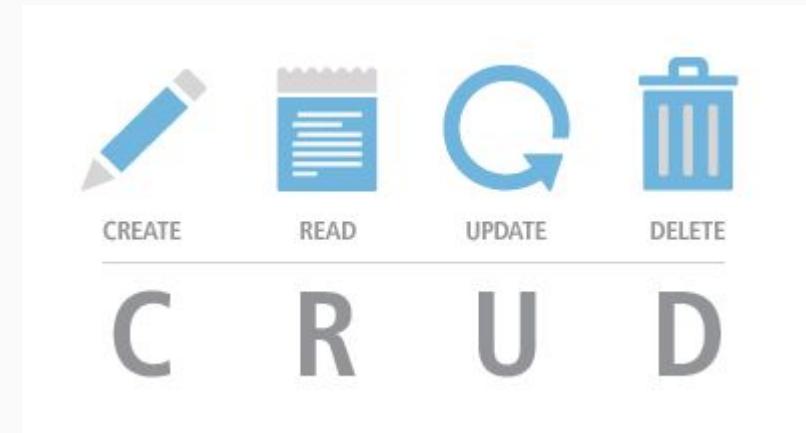


Fechar e abrir o CMD :)

Instalação do MongoDB Compass



CRUD no MongoDB



Instalação do MongoDB

```
C:\Users\Avell
λ cd C:\MongoDB\bin
```

```
C:\Users\Avell
λ cd C:\MongoDB\bin

C:\MongoDB\bin
λ mongod
{"t":{"$date":"2020-10-21T14:07:42.754-03:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2020-10-21T14:07:43.167-03:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
 {"t":{"$date":"2020-10-21T14:07:43.167-03:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in use."}
 {"t":{"$date":"2020-10-21T14:07:43.168-03:00"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":7640,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bit","host":"Avell"}}
 {"t":{"$date":"2020-10-21T14:07:43.168-03:00"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Target operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
 {"t":{"$date":"2020-10-21T14:07:43.169-03:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"4.4.1","gitVersion":"ad91a93a5a31e175f5cbf8c69561e788bbc55ce1","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
 {"t":{"$date":"2020-10-21T14:07:43.169-03:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Microsoft Windows 10","version":"10.0 (build 18363)"}}, {"t":{"$date":"2020-10-21T14:07:43.169-03:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{}}}}
```

Testar o MongoDB

cmd - mongo

```
C:\MongoDB\bin
λ mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("aeb6301c-80f6-4e18-ab4e-5e767a6206df") }
MongoDB server version: 4.4.1
---
The server generated these startup warnings when booting:
    2020-10-21T14:07:43.556-03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
    2020-10-21T14:07:43.556-03:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces . If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
---
---
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()
---
```

> |

mongod.exe mongo.exe

Search

CRUD no MongoDB

Create

| Type | Command | Description |
|-----------------|---|--|
| Create One | <code>.insertOne({<doc>})</code> | Inserts a document into the collection. IE: <code>db.cars.insertOne({make:"ford"})</code> Doc |
| Create One/Many | <code>.insert([{<doc>}, {<doc>}, {<doc>}])</code> | Inserts One or more documents into the collection. If an array is passed in it will make a record of each document in the array. Otherwise it will accept a single document. Doc |

CRUD no MongoDB

cmd - mongo

```
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>
```

cmd - mongo

```
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
> use detran
switched to db detran
> db.automoveis
detran.automoveis
```

cmd - mongo

```
> db.automoveis.insertOne({ tipo: 1, modelo: "esportivo", placa: 222222})
    "acknowledged" : true,
    "insertedId" : ObjectId("5f9095db369eaa57a2438f7a")
}
> db.automoveis.insertOne({ tipo: 1, modelo: "passeio", placa: 33333})
    "acknowledged" : true,
    "insertedId" : ObjectId("5f9095f5369eaa57a2438f7b")
}
> db.automoveis.insertOne({ tipo: 1, modelo: "jipe", placa: 55555})
    "acknowledged" : true,
    "insertedId" : ObjectId("5f909613369eaa57a2438f7c")
}
> db.automoveis.insertOne({ tipo: 2, modelo: "honda", placa: 111111})
    "acknowledged" : true,
    "insertedId" : ObjectId("5f909626369eaa57a2438f7d")
}
> db.automoveis.insertOne({ tipo: 2, modelo: "yamaha", placa: 9999999})
    "acknowledged" : true,
    "insertedId" : ObjectId("5f909635369eaa57a2438f7e")
}
> |
```

CRUD no MongoDB

Read

Most of these commands allow methods to be chained onto them.

| Type | Command | Description |
|---------------------|--------------------------------|--|
| Count | <code>.count()</code> | Returns the number of items in the collection. Doc |
| Return All | <code>.find({})</code> | Returns an array of all items in a collection. <code>.find()</code> Must be passed <code>{}</code> in order to return all documents. Doc |
| Return All Filtered | <code>.find({query})</code> | Returns an array of items in a collection that match the query passed into <code>.find()</code> . See Query and Project Operators for extra info on queries. Doc |
| Return One Filtered | <code>.findOne({query})</code> | Returns a document (not an array) of the first item found, filtering based off what was passed into <code>.findOne()</code> . Useful when searching by unique fields like <code>_id</code> , IE <code>db.cars.findOne({_id: 1})</code> . Doc |

CRUD no MongoDB

cmd - mongo

```
> db.automoveis.count()  
5  
> |
```

```
> db.automoveis.find()  
{ "_id" : ObjectId("5f9095db369eaa57a2438f7a"), "tipo" : 1, "modelo" : "esportivo", "placa" : 222222 }  
{ "_id" : ObjectId("5f9095f5369eaa57a2438f7b"), "tipo" : 1, "modelo" : "passeio", "placa" : 33333 }  
{ "_id" : ObjectId("5f909613369eaa57a2438f7c"), "tipo" : 1, "modelo" : "jipe", "placa" : 55555 }  
{ "_id" : ObjectId("5f909626369eaa57a2438f7d"), "tipo" : 2, "modelo" : "honda", "placa" : 111111 }  
{ "_id" : ObjectId("5f909635369eaa57a2438f7e"), "tipo" : 2, "modelo" : "yamaha", "placa" : 9999999 }  
> |
```

CRUD no MongoDB

```
cmd - mongo
> db.automoveis.find().pretty()
{
    "_id" : ObjectId("5f9095db369eaa57a2438f7a"),
        "tipo" : 1,
        "modelo" : "esportivo",
        "placa" : 222222
}

{
    "_id" : ObjectId("5f9095f5369eaa57a2438f7b"),
        "tipo" : 1,
        "modelo" : "passeio",
        "placa" : 0
}

{
    "_id" : ObjectId("5f909613369eaa57a2438f7c"),
        "tipo" : 1,
        "modelo" : "jipe",
        "placa" : 555555
}

{
    "_id" : ObjectId("5f909626369eaa57a2438f7d"),
        "tipo" : 2,
        "modelo" : "honda",
        "placa" : 111111
}

{
    "_id" : ObjectId("5f909635369eaa57a2438f7e"),
        "tipo" : 2,
        "modelo" : "yamaha",
        "placa" : 9999999
}
> |
```

CRUD no MongoDB

Update

| Type | Command | Description |
|--------------------|--|--|
| Update/Replace | <code>.update({query}, { \$set: {query} }, options)</code> | The first argument is used as the query to find the document. The second argument specifies which field which to update. Exclude <code>\$set:</code> and the entire document will be Replaced. Common options: <code>upsert: <boolean></code> to keep it unique, and <code>multi: <boolean></code> will update multiple documents if set to true. Docs and Field Operator Docs |
| Update One/Many | <code>.updateOne()</code> and <code>.updateMany()</code> | Basically the same as the above function, except the <code>multi: <boolean></code> option is basically defaulted to <code>false</code> and <code>true</code> , and isn't an allowed option that can be passed in. One Doc, Many Doc |

CRUD no MongoDB

```
> db.automoveis.insertOne({ tipo: 1, modelo: "passeio", placa: 33333})  
    "acknowledged" : true,  
    "insertedId" : ObjectId("5f9095f5369eaa57a2438f7b")  
}
```

cmd - mongo

```
> db.automoveis.updateOne({modelo: "passeio"}, {$set: {placa: 0000000}})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
> |
```

```
> db.automoveis.findOne({modelo: "passeio"})  
    "_id" : ObjectId("5f9095f5369eaa57a2438f7b"),  
    "tipo" : 1,  
    "modelo" : "passeio",  
    "placa" : 0  
}|
```

CRUD no MongoDB

Delete

| Type | Command | Description |
|-----------------|-----------------------------------|---|
| Delete One | <code>.deleteOne({query})</code> | Deletes the first document that matches the query. Recommend to search by <code>_id</code> or another unique field. Doc |
| Delete Many/All | <code>.deleteMany({query})</code> | Deletes all records that match the query. Leave the query blank to delete all documents. Doc |

cmd - mongo

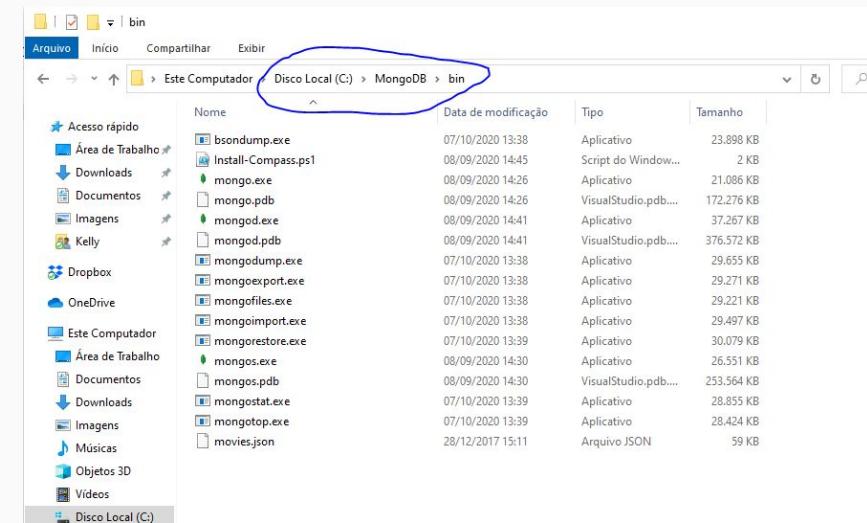
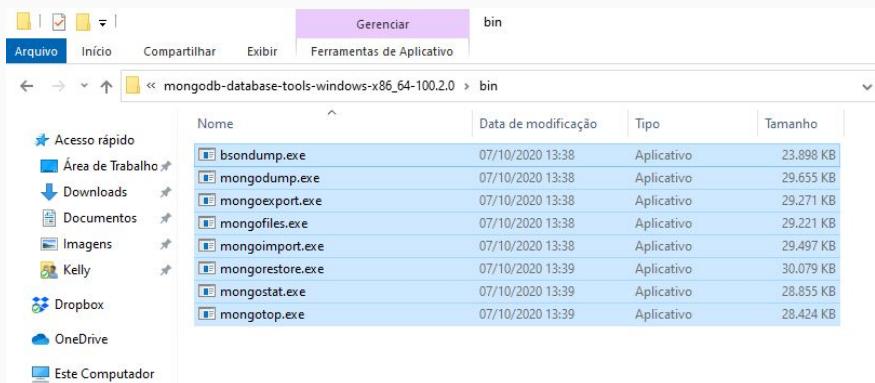
```
> db.automoveis.deleteOne({modelo: "passeio"})
{ "acknowledged" : true, "deletedCount" : 1 }
> |
```

<https://gist.github.com/michaeltreat/d3bdc989b54cff969df86484e091fd0c>

Importar Dados para o MongoDB

Baixar: https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools

Copiar



Importar Dados para o MongoDB

cmd

```
C:\MongoDB\bin
λ ls
bsondump.exe*    mongo.exe*   mongod.exe*   mongodump.exe*   mongofiles.exe*   mongorestore.exe*   mongos.pdb      mongotop.exe*
Install-Compass.ps1  mongo.pdb   mongod.pdb   mongoexport.exe*  mongoimport.exe*  mongos.exe*       mongostat.exe*   movies.json
```

```
C:\MongoDB\bin
λ |
```

```
C:\MongoDB\bin
λ mongoimport --db video --collection movies --file movies.json --jsonArray|
```

```
C:\MongoDB\bin
λ mongoimport --db video --collection movies --file movies.json --jsonArray
2020-10-22T10:27:13.903-0300      connected to: mongodb://localhost/
2020-10-22T10:27:13.950-0300      250 document(s) imported successfully. 0 document(s) failed to import.
```

Importar Dados para o MongoDB

MongoDB Compass - localhost:27017/video

Connect View Help

Local

5 DBS 3 COLLECTIONS C

☆ FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 4.4.1 Community

Filter your data

> admin

> config

> detran

> local

> video + -

movies

Collections

CREATE COLLECTION

| Collection Name | Documents | Avg. Document Size | Total Document Size | Num. Indexes | Total Index Size | Properties |
|-----------------|-----------|--------------------|---------------------|--------------|------------------|------------|
| movies | 250 | 184.0 B | 44.9 KB | 1 | 20.0 KB | |

Importar Dados para o MongoDB

The screenshot shows the MongoDB Compass interface connected to the 'video.movies' collection. The left sidebar displays the database structure with 'Local' selected, showing 5 DBs and 3 Collections. The 'video.movies' collection is highlighted. The main pane shows the document list with 250 entries. The first few documents are displayed as follows:

- `_id: ObjectId("5f9188b148b695d8d8bddbc5")`
title: "The Shawshank Redemption"
year: "1994"
director: "Frank Darabont"
duration: "2h 22min"
genre: Array
rate: "9.3"
- `_id: ObjectId("5f9188b148b695d8d8bddbc6")`
title: "The Lord of the Rings: The Return of the King"
year: "2002"
director: "Peter Jackson"
duration: "3h 21min"
genre: Array
rate: "8.9"
- `_id: ObjectId("5f9188b148b695d8d8bddbc7")`
title: "The Godfather"
year: "1972"
director: "Francis Ford Coppola"
duration: "2h 55min"
genre: Array
rate: "9.2"
- `_id: ObjectId("5f9188b148b695d8d8bddbc8")`
title: "Fight Club"
year: "1999"
director: "David Fincher"
duration: "2h 19min"

Tipo de dados

MongoDB Compass - localhost:27017/video.movies

Connect View Collection Help

Local

5 DBS 3 COLLECTIONS C

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.4.1 Community

Filter your data

admin config detran local video movies ...

video.movies Documents

video.movies DOCUMENTS 250 TOTAL SIZE 44.9KB AVG. SIZE 184B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER OPTIONS FIND RESET ...

ADD DATA VIEW { }  Displaying documents 1 - 20 of 250 REFRESH

| # | _id ObjectId | title String | year String | director String | duration String | ... |
|---|--------------------------|----------------------------------|-------------|------------------------|-----------------|---|
| 1 | 5f9188b148b695d8d8bddbc5 | "The Shawshank Redemption" | "1994" | "Frank Darabont" | "2h 22m" |    |
| 2 | 5f9188b148b695d8d8bddbc6 | "The Lord of the Rings: The Ret" | "2003" | "Peter Jackson" | "3h 21m" |    |
| 3 | 5f9188b148b695d8d8bddbc7 | "The Godfather" | "1972" | "Francis Ford Coppola" | "2h 55m" |    |
| 4 | 5f9188b148b695d8d8bddbc8 | "Fight Club" | "1999" | "David Fincher" | "2h 19m" |    |
| 5 | 5f9188b148b695d8d8bddbc9 | "Il buono, il brutto, il cattiv" | "1966" | "Sergio Leone" | "3h 2m" |    |
| 6 | 5f9188b148b695d8d8bddbcA | "Star Wars: Episode V - The Emp" | "1980" | "Irvin Kershner" | "2h 4m" |    |
| 7 | 5f9188b148b695d8d8bddbcB | "The Lord of the Rings: The Fel" | "2001" | "Peter Jackson" | "2h 58m" |    |
| 8 | 5f9188b148b695d8d8bddbcC | "Inception" | "2010" | "Christopher Nolan" | "2h 28m" |    |

Editando Dados

```
> _id: ObjectId("5a05b9f93c7b6ec72137fed6")
  title: "Fight Club"
  year: "1999"
  director: "David Fincher"
  duration: "2h 19min"
> genre: Array
  rate: "8.8"
```



```
1  _id: ObjectId("5a05b9f93c7b6ec72137fed6")
2  title : "Fight Club "
3  year : "1999 "
4  director : "David Fincher"
5  duration : "2h 19min "
6  > genre : Array
7  rate : "8.8 "
```

ObjectId
String
String
String
String
Array
String

Document Modified.

CANCEL UPDATE

Deletando Campos

```
1 _id: ObjectId("5a05b9f93c7b6ec72137fed6") ObjectId
2 title :"Fight Club " String
3 year :"1999 " String▼
4 director :"David Fincher " String
5 duration :"2h 19min " String
6 > genre :Array Array
7 rate :"8.8 " String
```

```
1 _id: ObjectId("5a05b9f93c7b6ec72137fed6") ObjectId
2 title :"Fight Club " String
3 year :"1999 " String
4 director :"David Fincher " String
5 duration :"2h 19min " String▼
6 > genre :Array Array
7 rate :"8.8 " String
```

Document Modified.

CANCEL UPDATE

Adicionando Campos

```
1 _id: ObjectId("5a05b9f93c7b6ec72137fed6") ObjectId
2 title : "Fight Club " String
3 year : "1999 " String
4 director : "David Fincher " String
5 duration : "2h 19min " String
6 > genre : Array Array
7 + rate : "8.8 " String▼
8 [ ] : " " String
```

Document Modified.

CANCEL UPDATE

Deletando documentos

```
> _id: ObjectId("5a05b9f93c7b6ec72137fed7")
  title: "The Lord of the Rings: The Fellowship of the Ring"
  year: "2001"
  director: "Peter Jackson"
  duration: "2h 58min"
> genre: Array
  rate: "8.8"
```



```
_id: ObjectId("5a05b9f93c7b6ec72137fed7")
title: "The Lord of the Rings: The Fellowship of the Ring"
year: "2001"
director: "Peter Jackson"
duration: "2h 58min"
> genre: Array
rate: "8.8"
```

Document Flagged For Deletion.

CANCEL DELETE

Filtros básicos

video.movies

DOCUMENTS 501

TOTAL SIZE
404.8KB

INDEXES 1

TOTAL SIZE
36.0KB

Avg. Size
827B

Documents

Schema

Explain Plan

Indexes

Validation

FILTER { title: "The Godfather" }

OPTIONS

FIND

RESET



INSERT DOCUMENT

VIEW

LIST

TABLE

Displaying documents 1 - 1 of 1



```
_id: ObjectId("5a05b9f93c7b6ec72137fece")
title: "The Godfather"
year: 1972
director: "Francis Ford Coppola"
duration: "2h 55min"
> genre: Array
  rate: "9.2"
```



Documents

Schema

Explain Plan

Indexes

Validation

FILTER

{ "_id": ObjectId('5a05b9f93c7b6ec72137fece') }

OPTIONS

FIND

RESET



Query com condicionais: \$and, \$or, \$ne, \$nor

```
{ $and: [ { <expression1> }, { <expression2> }, ... { <expressionN> } ] }
```

Documents Schema Explain Plan Indexes Validation

FILTER { \$and : [{ year : "2000" }, { rate: "8.5" }] } **OPTIONS** **FIND** **RESET**

INSERT DOCUMENT **VIEW** **LIST** **TABLE** Displaying documents 1 - 2 of 2

_id: ObjectId("5a05b9f93c7b6ec72137fefafa")
title: "Gladiator"
year: "2000"
director: "Ridley Scott"
duration: "2h 35min"
 > genre: Array
 rate: "8.5"

_id: ObjectId("5a05b9f93c7b6ec72137fefdf")
title: "Memento"
year: "2000"
director: "Christopher Nolan"
duration: "1h 53min"
 > genre: Array
 rate: "8.5"

Query com condicionais: \$and, \$or, \$ne, \$nor

```
{ $or: [ { <expression1> }, { <expression2> }, ... { <expressionN> } ] }
```

video.movies

DOCUMENTS 501 TOTAL SIZE 404.8KB AVG. SIZE 827B | INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Documents Schema Explain Plan Indexes Validation

FILTER { \$or : [{ year : "2000" }, { rate: "8.5" }] }

FIND RESET

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 20 of 38

`_id: ObjectId("5a05b9f93c7b6ec72137feea")
title: "American History X"
year: "1990"
director: "Tony Kaye"
duration: "1h 59min"
> genre: Array
rate: "8.5"`

`_id: ObjectId("5a05b9f93c7b6ec72137feed")
title: "Psycho"
year: "1960"
director: "Alfred Hitchcock"
duration: "1h 49min"
> genre: Array
rate: "8.5"`

`_id: ObjectId("5a05b9f93c7b6ec72137feee")
title: "The Green Mile"
year: "1999"
director: "Frank Darabont"
duration: "3h 9min"
> genre: Array
rate: "8.5"`

Query com condicionais: \$and, \$or, \$ne, \$nor

Not equal -> inclui documentos que não contém o valor informado.

```
{ <field> : { $ne : <value> } }
```

Documents Schema Explain Plan Indexes Validation

FILTER { rate: { \$ne: "9.0" } } **OPTIONS** **FIND** **RESET**

INSERT DOCUMENT **VIEW** LIST TABLE Displaying documents 1 - 20 of 499

Query com condicionais: \$and, \$or, \$ne, \$nor

```
{ $nor: [ { <expression1> }, { <expression2> }, ... { <expressionN> } ] }
```

video.movies

DOCUMENTS 501

TOTAL SIZE
404.8KB

AVG. SIZE
827B

INDEXES 1

TOTAL SIZE
36.0KB

AVG. SIZE
36.0KB

Documents

Schema

Explain Plan

Indexes

Validation



{ \$nor: [{ year: "2000" }, { director: "Sidney Lumet" }] }

▶ OPTIONS

FIND

RESET



INSERT DOCUMENT

VIEW

LIST

TABLE

Displaying documents 1 - 20 of 486



Operadores de comparação

```
C:\MongoDB\bin
λ mongoimport --db restaurants --collection restaurants --drop --file primer-dataset.json
2020-10-22T11:55:24.466-0300      connected to: mongodb://localhost/
2020-10-22T11:55:24.499-0300      dropping: restaurants.restaurants
2020-10-22T11:55:25.285-0300      25359 document(s) imported successfully. 0 document(s) failed to import.
```

```
{
  "address": {
    "building": "1007",
    "coord": [ -73.856077, 40.848447 ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
    { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
    { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
    { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
    { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
    { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445",
  "stars": 4,
  "reviews": 5,
  "capacity": 52,
  "tags": ['awesome place', 'great food', 'good music', 'incredible chef']
}
```

Operadores de comparação: \$eq

```
{ field: { $eq: value } }
```

```
{ borough : { $eq: "Brooklyn" } }
```

restaurants.restaurants

DOCUMENTS 25.4k TOTAL SIZE 10.1MB AVG. SIZE 419B INDEXES 1 TOTAL SIZE 232.0KB AVG. SIZE 232.0KB

Documents Schema Explain Plan Indexes Validation

FILTER { borough : { \$eq: "Brooklyn" } } **OPTIONS** **FIND** **RESET**

INSERT DOCUMENT VIEW **LIST** **TABLE** Displaying documents 1 - 20 of 6086

_id: ObjectId("5a47593aad8ad3cec0494213")

> address: Object
borough: "Brooklyn"
cuisine: "Hamburgers"
> grades: Array
name: "Wendy'S"
restaurant_id: "30112340"

_id: ObjectId("5a47593aad8ad3cec0494215")

> address: Object
borough: "Brooklyn"
cuisine: "American"
> grades: Array
name: "Riviera Caterer"
restaurant_id: "40356018"

Operadores de comparação: \$ne - Not Equal

```
{ field: { $ne: value } }
```

```
{ cuisine : { $ne: "Hamburgers" } }
```

restaurants.restaurants

DOCUMENTS 25.4k TOTAL SIZE 10.1MB AVG. SIZE 419B INDEXES 1 TOTAL SIZE 232.0KB AVG. SIZE 232.0KB

Documents Schema Explain Plan Indexes Validation

FILTER { cuisine : { \$ne: "Hamburgers" } }

OPTIONS **FIND** **RESET**

INSERT DOCUMENT **VIEW** **LIST** **TABLE** Displaying documents 1 - 20 of 24926

_id: ObjectId("5a47593aad8ad3cec0494212")

> address: Object
borough: "Bronx"
cuisine: "Bakery"
> grades: Array
name: "Morris Park Bake Shop"
restaurant_id: "30075445"

_id: ObjectId("5a47593aad8ad3cec0494214")

> address: Object
borough: "Manhattan"
cuisine: "Irish"
> grades: Array
name: "Dj Reynolds Pub And Restaurant"
restaurant_id: "30191841"

Operadores de comparação: \$gt - greater than / \$gte - greater than equal

The screenshot shows the MongoDB Compass interface with the database 'video.movies' selected. The 'Documents' tab is active. A filter is applied: { year : { \$gte: "2003" } }. The results show three movie documents:

- `_id: ObjectId("5f9188b148b695d8d8bddbc6")`
title: "The Lord of the Rings: The Return of the King"
year: "2003"
director: "Peter Jackson"
duration: "3h 21min"
> genre: Array
rate: "8.9"
- `_id: ObjectId("5f9188b148b695d8d8bddbcc")`
title: "Inception"
year: "2010"
director: "Christopher Nolan"
duration: "2h 28min"
> genre: Array
rate: "8.8"
- `_id: ObjectId("5f9188b148b695d8d8bddbd3")`
title: "The Dark Knight"
year: "2008"
director: "Christopher Nolan"
duration: "2h 32min"
> genre: Array
rate: "9.0"

Operadores de comparação: \$lt - less than / \$gt - less than equal

{ field: { \$lt: value } }

video.movies

DOCUMENTS 250 TOTAL SIZE 44.9KB AVG. SIZE 184B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { year : { \$lt: "2003" } } OPTIONS FIND RESET ...

ADD DATA VIEW { } ■■■■■ ADD DATA VIEW { } ■■■■■

Displaying documents 1 - 20 of 173 < > REFRESH

```
_id:ObjectId("5f9188b148b695d8d8bddbc5")
title:"The Shawshank Redemption"
year:"1994"
director:"Frank Darabont"
duration:"2h 22min"
> genre:Array
```

```
_id:ObjectId("5f9188b148b695d8d8bddbc7")
title:"The Godfather"
year:"1972"
director:"Francis Ford Coppola"
duration:"2h 55min"
> genre:Array
rate:"9.2"
```

```
_id:ObjectId("5f9188b148b695d8d8bddbc8")
title:"Fight Club"
year:"1999"
director:"David Fincher"
duration:"2h 19min"
> genre:Array
rate:"8.8"
```

Operadores de comparação

| Name | Description | Syntax |
|-------|---|------------------------------|
| \$eq | Matches values that are equal to a specified value | { field : { \$eq: value } } |
| \$ne | Matches all values that are not equal to a specified value | { field : { \$ne: value } } |
| \$gt | Matches values that are greater than a specified value | { field : { \$gt: value } } |
| \$gte | Matches values that are greater than or equal to a specified value | { field : { \$gte: value } } |
| \$lt | Matches values that are less than a specified value | { field : { \$lt: value } } |
| \$lte | Matches values that are less than or equal to a specified value | { field : { \$lte: value } } |

Operações em Array: \$in

```
{ field: { $in: [ value1, value2, ..... , valueN ] } }
```

video.movies

| DOCUMENTS | 250 | TOTAL SIZE | 44.9KB | AVG. SIZE | 184B |
|---|--|-------------------------------------|--------------------------------------|--|------------|
| INDEXES | 1 | TOTAL SIZE | 20.0KB | AVG. SIZE | 20.0KB |
| Documents | Aggregations | Schema | Explain Plan | Indexes | Validation |
| <input checked="" type="checkbox"/> FILTER {genre: { \$in: ["Drama", "Crime"] } } | <input type="button" value="OPTIONS"/> | <input type="button" value="FIND"/> | <input type="button" value="RESET"/> | ... | |
| <input type="button" value="ADD DATA"/> | <input type="button" value="VIEW"/> | <input type="button" value="{}"/> | <input type="button" value="grid"/> | Displaying documents 1 - 20 of 190 < > C REFRESH | |
| <pre>_id: ObjectId("5f9188b148b695d8d8bddbc5") title: "The Shawshank Redemption" year: "1994" director: "Frank Darabont" duration: "2h 22min" > genre: Array</pre> | | | | | |
| <pre>_id: ObjectId("5f9188b148b695d8d8bddbc6") title: "The Lord of the Rings: The Return of the King" year: "2003" director: "Peter Jackson" duration: "3h 21min" > genre: Array rate: "8.9"</pre> | | | | | |
| <pre>_id: ObjectId("5f9188b148b695d8d8bddbc7") title: "The Godfather" year: "1972" director: "Francis Ford Coppola" duration: "2h 55min" > genre: Array rate: "9.2"</pre> | | | | | |

Operações em Array: \$nin

```
{ field: { $nin: [ value1, value2, ..... , valueN ] } }
```

video.movies

| DOCUMENTS | 250 | TOTAL SIZE | 44.9KB | AVG. SIZE | 184B | INDEXES | 1 | TOTAL SIZE | 20.0KB | AVG. SIZE | 20.0KB |
|-----------|-----|------------|--------|-----------|------|---------|---|------------|--------|-----------|--------|
|-----------|-----|------------|--------|-----------|------|---------|---|------------|--------|-----------|--------|

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER {genre: { \$nin: ["Drama", "Crime"] } } OPTIONS FIND RESET ...

ADD DATA VIEW { } ■■■■■

Displaying documents 1 - 20 of 60 < > C REFRESH

```
_id: ObjectId("5f9188b148b695d8d8bddbc9")
title: "Il buono, il brutto, il cattivo"
year: "1966"
director: "Sergio Leone"
duration: "3h 2min"
> genre: Array
rate: "8.9"
```

```
_id: ObjectId("5f9188b148b695d8d8bddbca")
title: "Star Wars: Episode V - The Empire Strikes Back"
year: "1980"
director: "Irvin Kershner"
duration: "2h 4min"
> genre: Array
rate: "8.8"
```

```
_id: ObjectId("5f9188b148b695d8d8bddbcc")
title: "Inception"
year: "2010"
director: "Christopher Nolan"
duration: "2h 28min"
> genre: Array
rate: "8.8"
```

Operações em Array: \$all

{ field: { \$all: [value1, value2, , valueN] } }

The screenshot shows the MongoDB Compass interface for a collection named "video.movies". The interface includes a header with document and index statistics, a toolbar with tabs for "Documents" (selected), "Aggregations", "Schema", "Explain Plan", "Indexes", and "Validation". Below the toolbar is a search bar with a filter condition: {genre: {\$all: ["Drama", "Crime"]}}. The main area displays three document results:

- ```
_id: ObjectId("5f9188b148b695d8d8bddbc5")
title: "The Shawshank Redemption"
year: "1994"
director: "Frank Darabont"
duration: "2h 22min"
> genre: Array
```
- ```
_id: ObjectId("5f9188b148b695d8d8bddbc7")
title: "The Godfather"
year: "1972"
director: "Francis Ford Coppola"
duration: "2h 55min"
> genre: Array
rate: "9.2"
```
- ```
_id: ObjectId("5f9188b148b695d8d8bddbd0")
title: "12 Angry Men"
year: "1957"
director: "Sidney Lumet"
duration: "1h 36min"
> genre: Array
rate: "8.9"
```

The interface also features a "DISPLAYING DOCUMENTS 1 - 20 OF 50" message and standard navigation buttons.

## Operações em Array: \$all

```
{ field: { $elemMatch: { <query1>, <query2>, ... } } }
```

**restaurants.restaurants**

DOCUMENTS **25.4k** TOTAL SIZE 10.1MB AVG. SIZE 419B | INDEXES **1** TOTAL SIZE 236.0KB AVG. SIZE 236.0KB

**Documents** Aggregations Schema Explain Plan Indexes Validation

**FILTER** { grades: { \$elemMatch: { grade: "A", score: { \$gte: 15 } } } }

**OPTIONS** **FIND** **RESET** ...

**ADD DATA** **VIEW** **{}** **[ ]**

Displaying documents 1 - 9 of 9 < > **REFRESH**

```
_id:ObjectId("5f919d5c68ef451d9db60226")
address: Object
 building: "107-23"
 coord: Array
 0: -73.834012
 1: 40.683833
 street: "Liberty Avenue"
 zipcode: "11417"
borough: "Queens"
cuisine: "Caribbean"
grades: Array
 0: Object
 date: 2014-03-29T00:00:00.000+00:00
 grade: "A"
 score: 27
 1: Object
 date: 2013-06-12T00:00:00.000+00:00
 grade: "A"
 score: 12
 2: Object
 date: 2012-05-10T00:00:00.000+00:00
 grade: "A"
 score: 13
 3: Object
 date: 2011-12-29T00:00:00.000+00:00
 grade: "A"
 score: 13
name: "Gemini's Lounge"
restaurant_id: "40511696"
```

## Operações em Array

| Name        | Description                                                                                       | Syntax                                                   |
|-------------|---------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| \$nin       | Matches none of the values specified in an array                                                  | { field: { \$in: [ value1, value2, ..... , valueN ] } }  |
| \$in        | Matches any of the values specified in an array.                                                  | { field: { \$nin: [ value1, value2, ..... , valueN ] } } |
| \$all       | Matches arrays that contain all elements specified in the query.                                  | { field: { \$all: [ value1, value2, ..... , valueN ] } } |
| \$elemMatch | Selects documents if element in the array field matches all the specified \$elemMatch conditions. | { field: { \$elemMatch: { <query1>, <query2>, ... } } }  |

## MongoDB - Data Models

Uma das vantagens do Mongo é seu esquema flexível; isso nos permite criar documentos com diferentes estruturas. Embora possamos ter alguns com vários campos na mesma coleção, o cenário mais comum é agruparmos aqueles com propriedades semelhantes na mesma coleção.

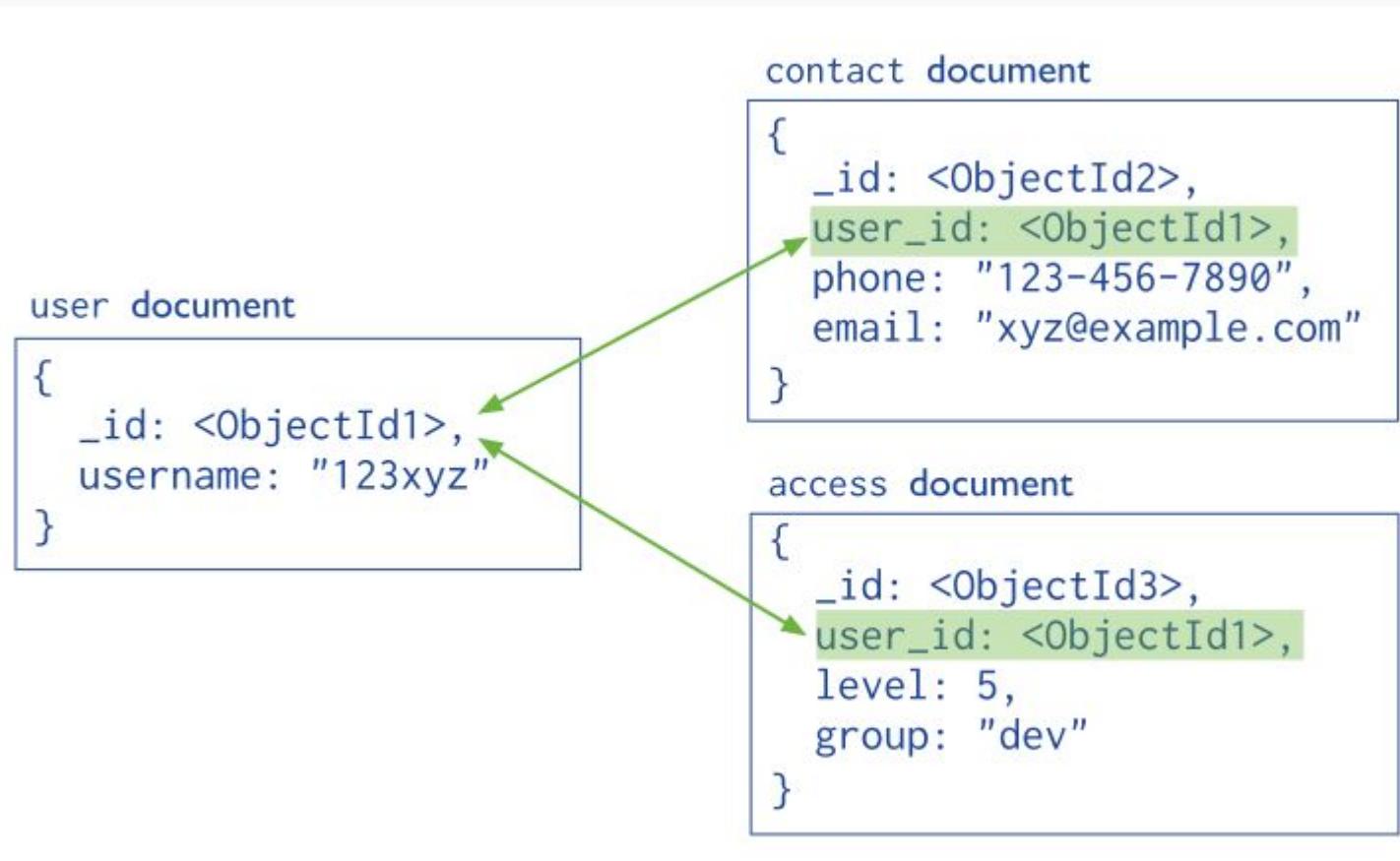
Ao projetar o modelo de dados de um Banco de Dados, devemos considerar os tipos de dados que usaremos na aplicação. A chave é compreender a estrutura dos documentos e como são representadas as relações entre eles.

# MongoDB - Data Models

A decisão crítica ao projetar modelos de dados para aplicativos MongoDB gira em torno da estrutura dos documentos e como ela representa os relacionamentos entre os dados. Duas ferramentas nos permitem mostrar esses links:

- Referencing documents
- Embedding documents

## MongoDB - Referencing documents - Relations



# MongoDB - Referencing documents - Relations

## User Collection

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Willy",
 lastName: "Doe",
 email: "willydow@gmail.com",
 birthday: 1990-12-14 01:00:00.000,
 phone: "123412399"
}
```

## Address Collection

```
{
 _id: ObjectId("59f30dd86f0b06a96e31bbb9"),
 user_id: ObjectId("593e7a1a4299c306d656200f"),
 street: "123 Fake Street",
 city: "Faketown",
 state: "MA",
 zip: "12345"
}
```

## MongoDB - Embedding documents

```
{
 _id: <ObjectId1>,
 username: "123xyz",
 contact: {
 phone: "123-456-7890",
 email: "xyz@example.com"
 },
 access: {
 level: 5,
 group: "dev"
 }
}
```

Embedded sub-document

Embedded sub-document

# MongoDB - Embedding documents

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Willy",
 lastName: "Doe",
 email: "willydow@gmail.com",
 birthday: 1990-12-14 01:00:00.000,
 phone: "123412399",
 address: {
 street: "123 Fake Street",
 city: "Faketown",
 state: "MA",
 zip: "12345"
 }
}
```

## MongoDB - Embedding documents - Multiple Sub-documents

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Willy",
 lastName: "Doe",
 email: "willydow@gmail.com",
 birthday: 1990-12-14 01:00:00.000,
 phone: "123412399",
 addresses: [
 {
 street: "123 Fake Street",
 city: "Faketon",
 state: "MA",
 zip: "12345"
 },
 {
 street: "1 Some Other Street",
 city: "Boston",
 state: "MA",
 zip: "12345"
 }
]
}
```

## Model One-to-One Relationships with Embedded Documents

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Joe Bookreader"
}

{
 patron_id: ObjectId("593e7a1a4299c306d656200f"),
 street: "123 Fake Street",
 city: "Faketon",
 state: "MA",
 zip: "12345"
}
```

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Joe Bookreader",
 address: {
 street: "123 Fake Street",
 city: "Faketon",
 state: "MA",
 zip: "12345"
 }
}
```

## Model One-to-Many Relationships with Embedded Documents

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Joe Bookreader"
}

{
 patron_id: ObjectId("593e7a1a4299c306d656200f"),
 street: "123 Fake Street",
 city: "Faketon",
 state: "MA",
 zip: "12345"
}

{
 patron_id: ObjectId("593e7a1a4299c306d656200f"),
 street: "1 Some Other Street",
 city: "Boston",
 state: "MA",
 zip: "12345"
}
```

```
{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 name: "Joe Bookreader",
 addresses: [
 {
 street: "123 Fake Street",
 city: "Faketon",
 state: "MA",
 zip: "12345"
 },
 {
 street: "1 Some Other Street",
 city: "Boston",
 state: "MA",
 zip: "12345"
 }
]
}
```

# Model Many-to-Many Relationships with Document References

```
{
 title: "MongoDB: The Definitive Guide",
 author: ["Kristina Chodorow", "Mike Dirolf"],
 published_date: ISODate("2010-09-24"),
 pages: 216,
 language: "English",
 publisher: {
 name: "O'Reilly Media",
 founded: 1980,
 location: "CA"
 }
}

{
 title: "50 Tips and Tricks for MongoDB Developer",
 author: "Kristina Chodorow",
 published_date: ISODate("2011-05-06"),
 pages: 68,
 language: "English",
 publisher: {
 name: "O'Reilly Media",
 founded: 1980,
 location: "CA"
 }
}
```

## Model Many-to-Many Relationships with Document References

```
{
 name: "O'Reilly Media",
 founded: 1980,
 location: "CA",
 books: [123456789, 234567890, ...]
}

{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 title: "MongoDB: The Definitive Guide",
 author: ["Kristina Chodorow", "Mike Dirolf"],
 published_date: ISODate("2010-09-24"),
 pages: 216,
 language: "English"
}

{
 _id: ObjectId("593e7b2b4299c306d656299d"),
 title: "50 Tips and Tricks for MongoDB Developer",
 author: "Kristina Chodorow",
 published_date: ISODate("2011-05-06"),
 pages: 68,
 language: "English"
}
```

## Model Many-to-Many Relationships with Document References

```
{
 _id: ObjectId("593e7a1a2312c306d321323g"),
 name: "O'Reilly Media",
 founded: 1980,
 location: "CA"
}

{
 _id: ObjectId("593e7a1a4299c306d656200f"),
 title: "MongoDB: The Definitive Guide",
 author: ["Kristina Chodorow", "Mike Dirolf"],
 published_date: ISODate("2010-09-24"),
 pages: 216,
 language: "English",
 publisher_id: ObjectId("593e7a1a2312c306d321323g")
}

{
 _id: ObjectId("593e7b2b4299c306d656299d"),
 title: "50 Tips and Tricks for MongoDB Developer",
 author: "Kristina Chodorow",
 published_date: ISODate("2011-05-06"),
 pages: 68,
 language: "English",
 publisher_id: ObjectId("593e7a1a2312c306d321323g")
}
```

# Referencing documents - Relations ou Embedded Documents

## Twitter

- Users
- Tweets
- Followers
- Favorites

## Spotify

- Users
- Artists
- Songs
- Albums
- Genres
- Favorites

## Airbnb

- Users
- Homes
- Bookings
- Reviews

# mongoose

elegant `mongodb` object modeling for `node.js`

## Setup

```
$ mkdir mongoose-example
```

```
$ cd mongoose-example
```

```
$ npm init
```

```
$ npm install mongoose
```

```
$ touch example.js
```

# Connecting to the Database

Criar arquivo “example.js”

```
const mongoose = require('mongoose');

mongoose

.connect('mongodb://localhost/exampleApp', {useNewUrlParser: true})

.then(x => console.log(`Connected to Mongo! Database name: "${x.connections[0].name}"`))

.catch(err => console.error('Error connecting to mongo', err));
```

## Criar uma instância e salvar

```
const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'kelly' });

kitty
 .save()
 .then(newCat => console.log(`A new cat is created: ${newCat}`))
 .catch(err => console.log(`Error while creating a new cat: ${err}`));
```

Connect View Collection Help

Local

5 DBS 3 COLLECTIONS

★ FAVORITE

HOST  
localhost:27017CLUSTER  
StandaloneEDITION  
MongoDB 4.4.1 Community

Filter your data

&gt; admin

&gt; config

&gt; detran

&gt; exampleApp

cats

exampleApp.cats  
Documents

## exampleApp.cats

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

DOCUMENTS 1 TOTAL SIZE 47B AVG. SIZE 47B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

FILTER

OPTIONS

FIND

RESET

...

ADD DATA



VIEW



Displaying documents 1 - 1 of 1



REFRESH

```
_id: ObjectId("5f91e08761d0a74e3019d837")
name: "Kelly"
__v: 0
```