Introdução ao Banco de dados

Reprograma - Backend

Profª: 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.

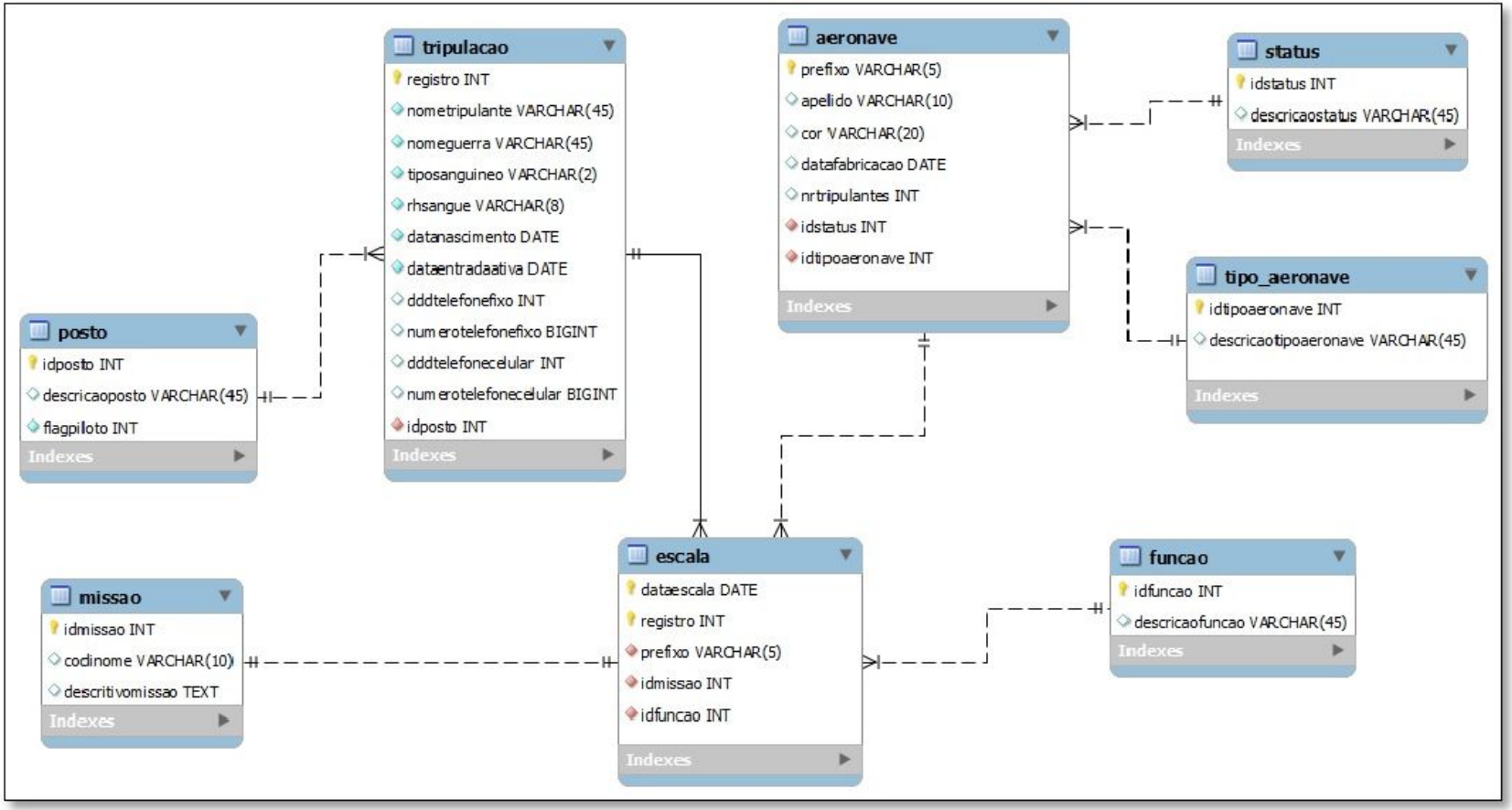
Outra característica importante deste modelo é o uso de elementos para garantir a integridade dos dados. As restrições mais comuns são as chaves primárias e as estrangeiras.

Banco de dados relacionais

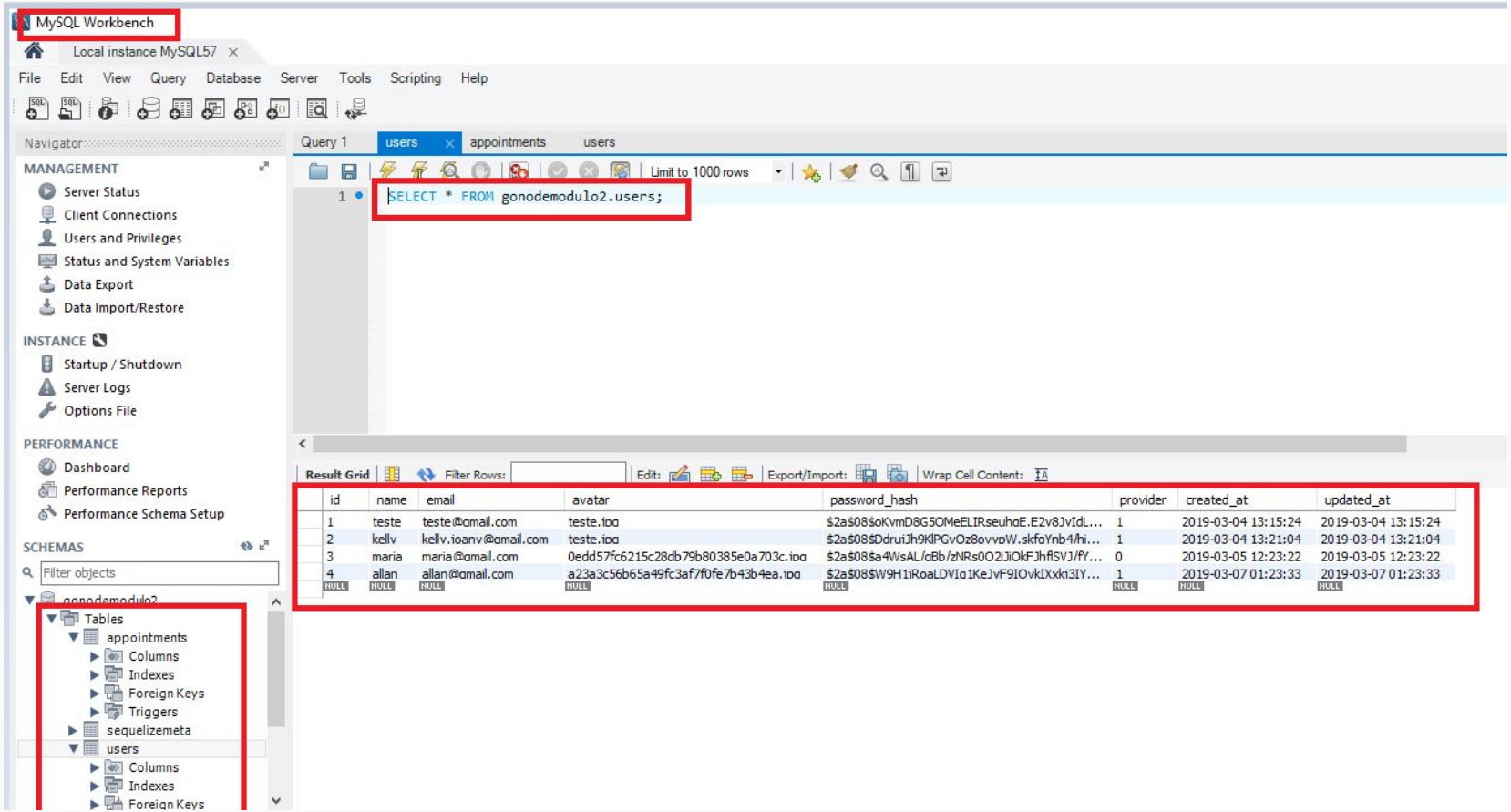
O termo chave primária é utilizado para identificar o atributo que foi escolhido pelo projetista do banco para identificar unicamente os registros que são armazenados em uma determinada tabela.

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



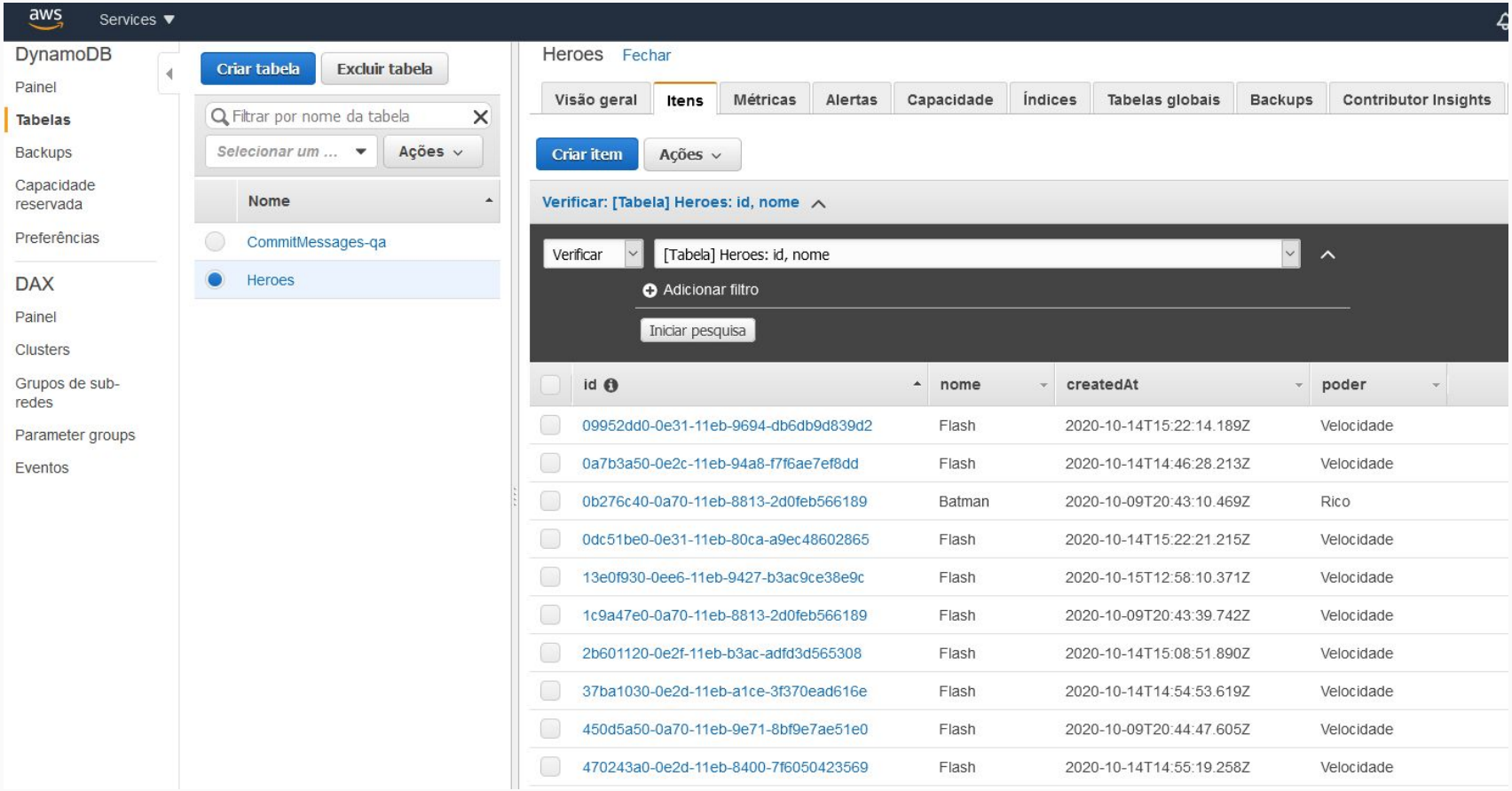
Banco de dados não relacionais

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.

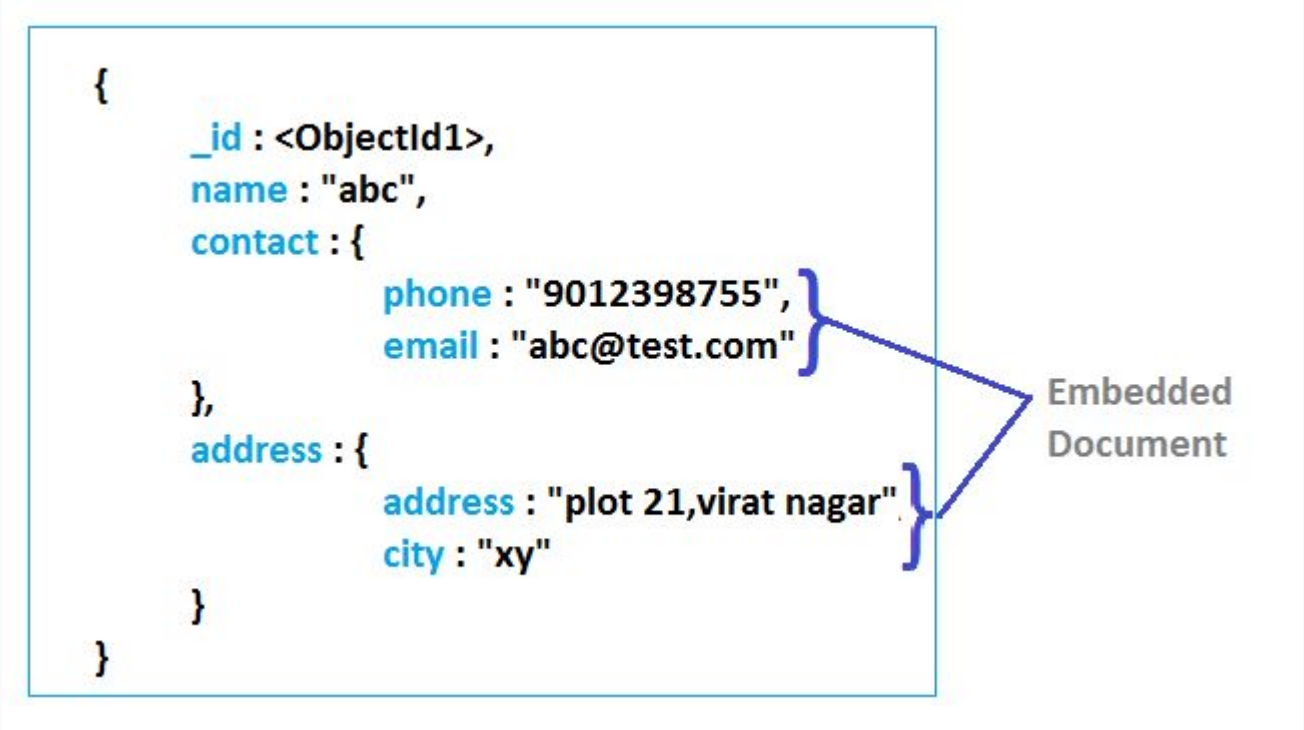
- **Chave valor:** é precisamente o que o nome descreve você tem uma chave e um valor, o uso geralmente é para os casos onde você faz buscas somente pela chave, não sendo possível buscar pelo valor, casos de uso desse modelo geralmente são armazenamento de sessões de usuários e carrinhos de compras, 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



Banco de dados não relacionais



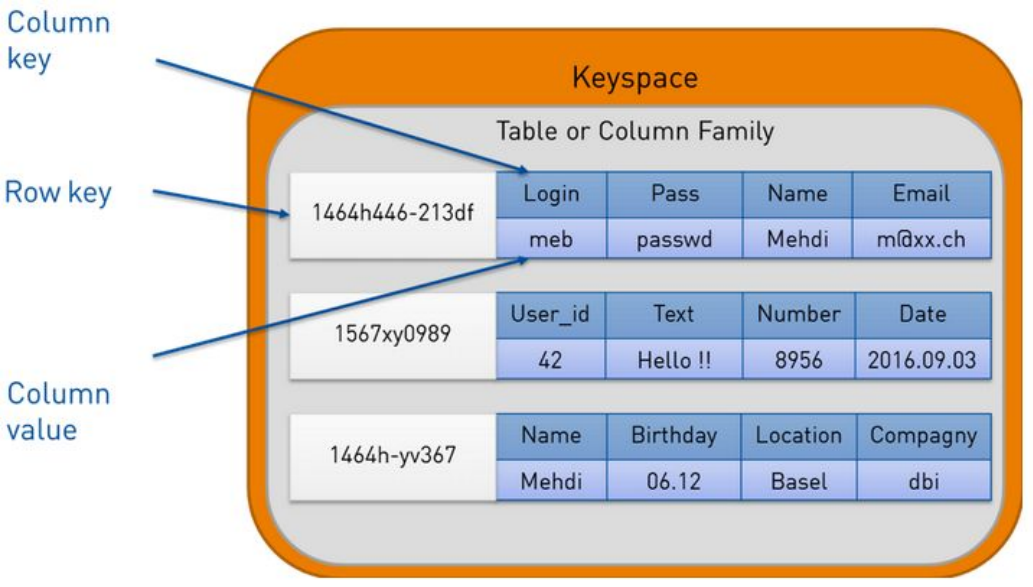
Banco de dados não relacionais

- **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.

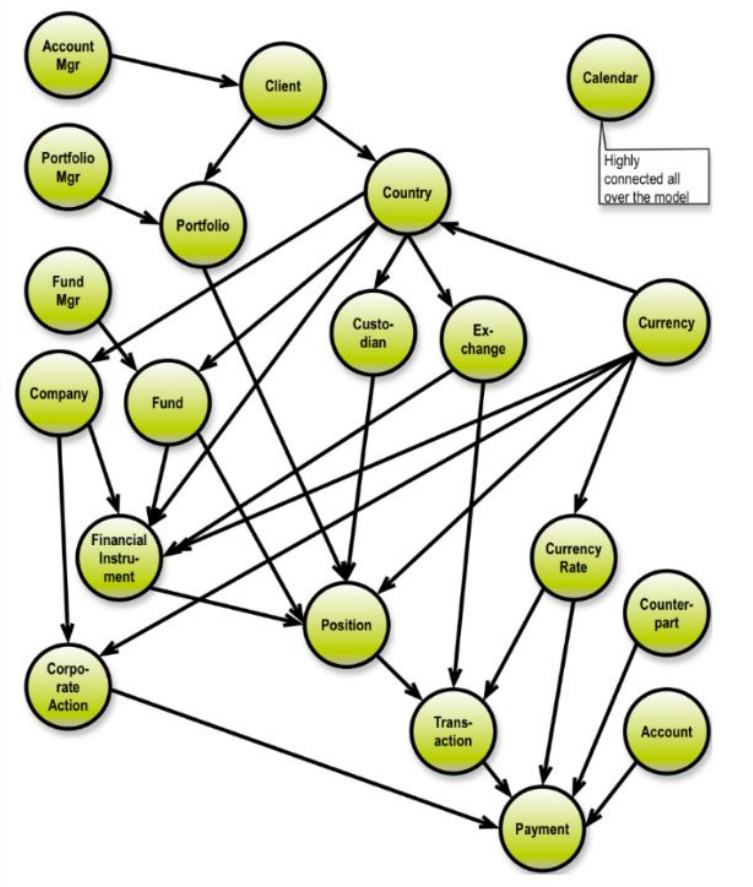
- **Grafo:** este é válido para modelos de dados onde é importante o relacionamento entre eles e o relacionamento é complexo, por exemplo você quer saber se o João e a Maria foram no restaurante Y. Os casos de uso mais comuns para este modelo são redes sociais e serviços baseados em localização. Bancos: 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

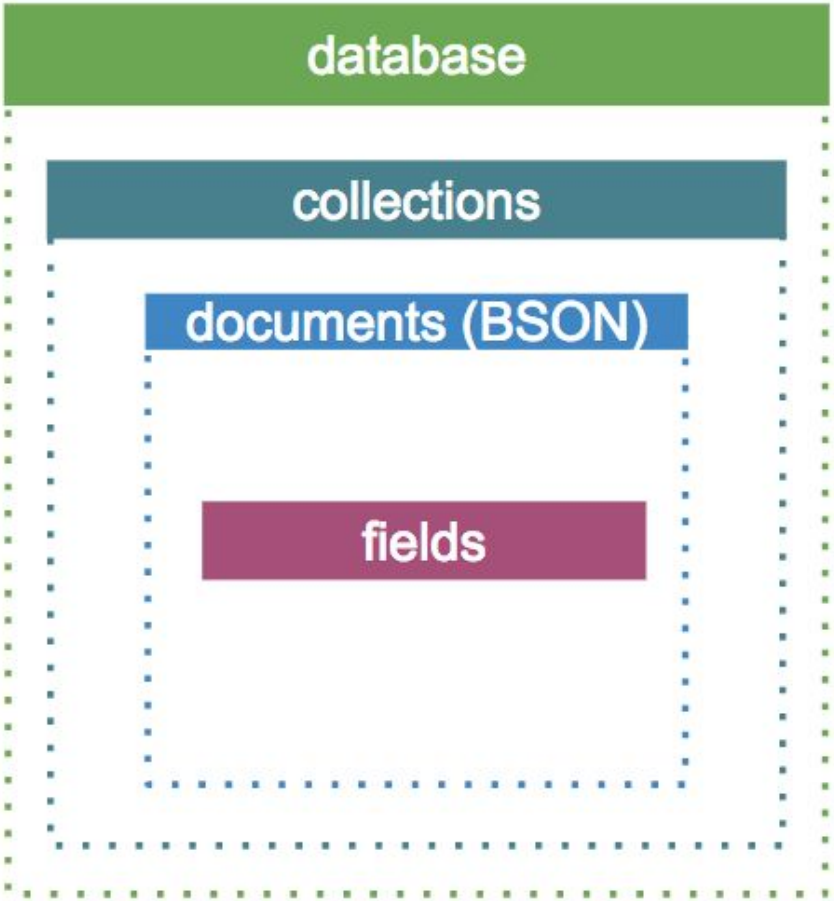


Introdução ao MongoDB

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

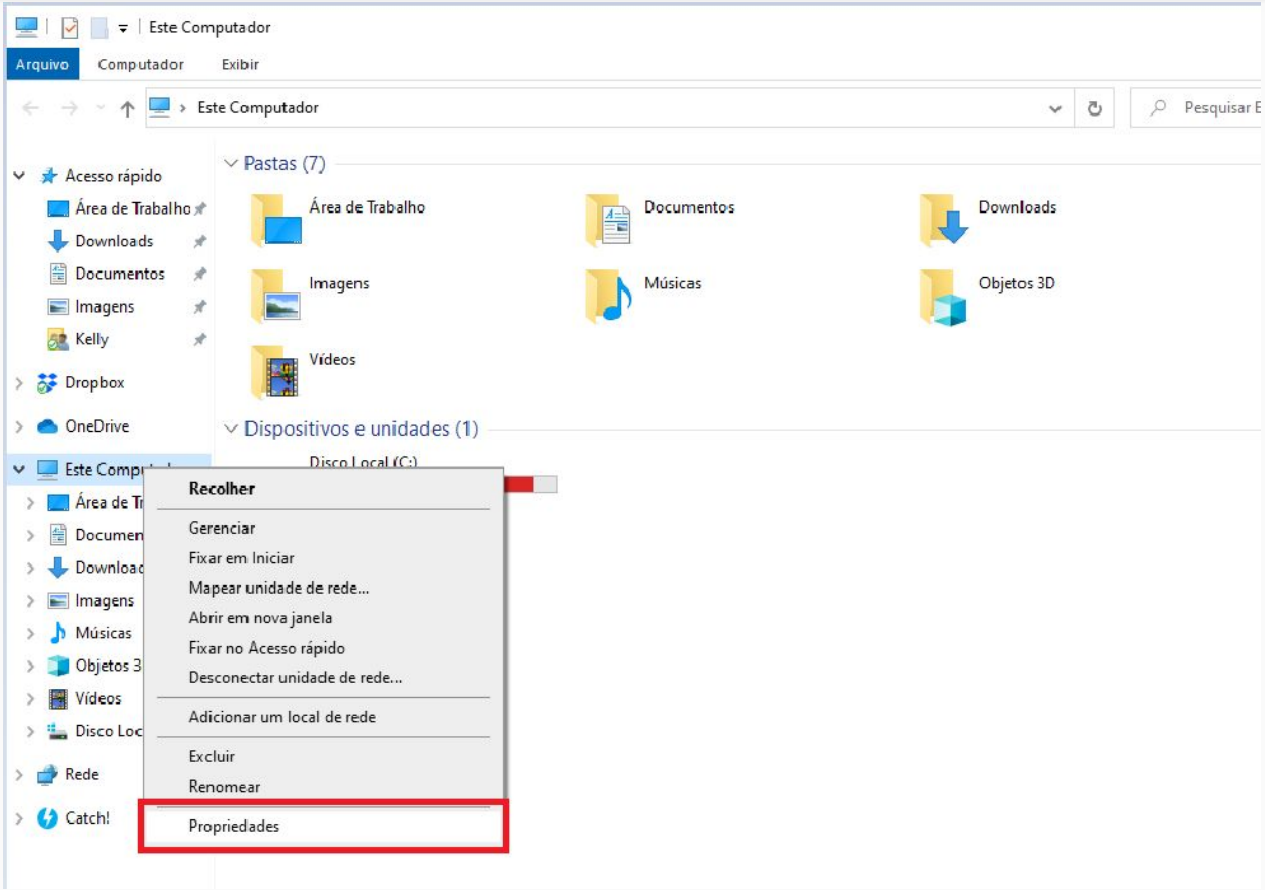
- Windows x64
  - Archive: mongodb-windows-x86_64-4.4.1.zip

Criar uma pasta no C:\MongoDB

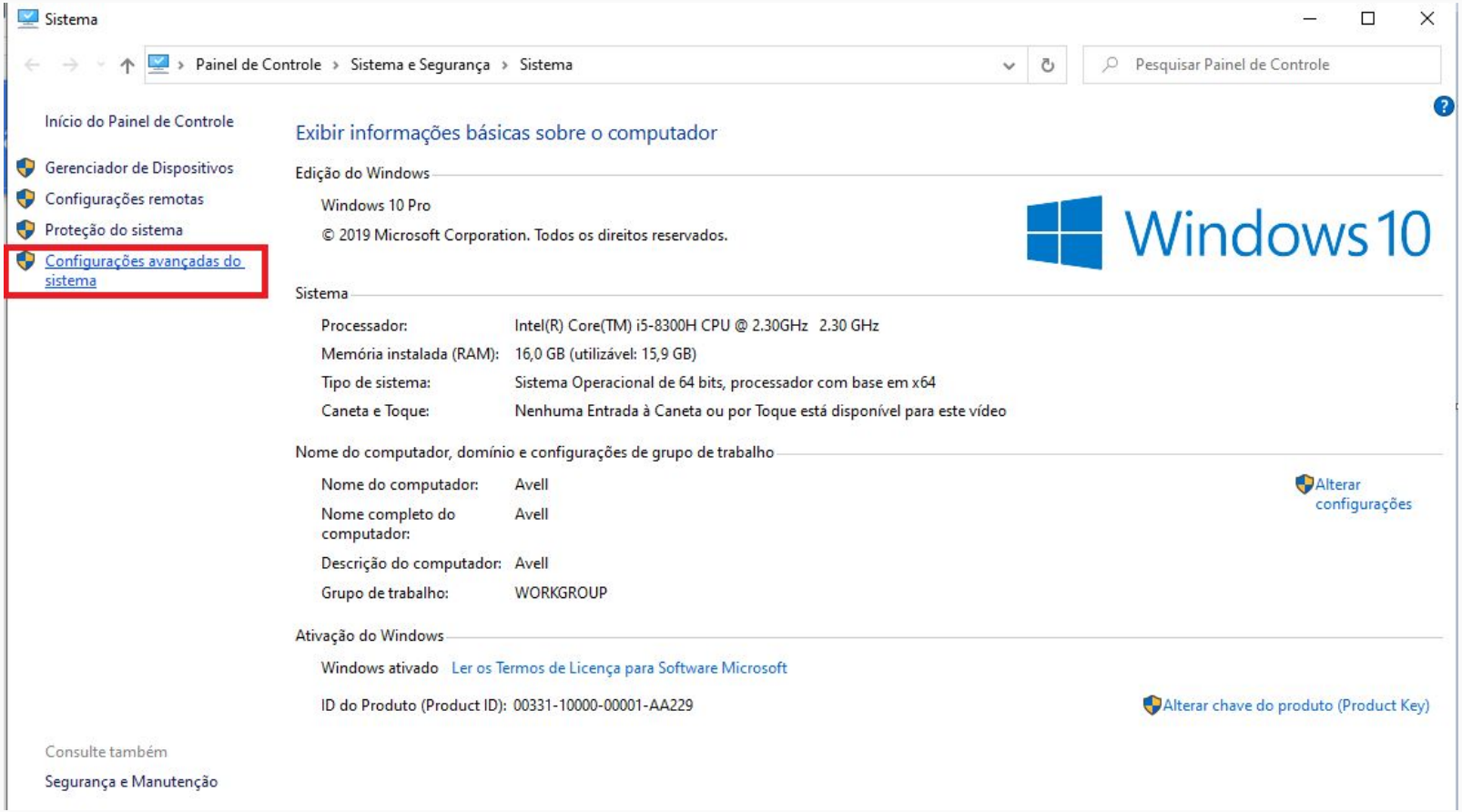
Dezipar os arquivos e mover os arquivos para a pasta recém criada.

Criar uma pasta no C:\data\db

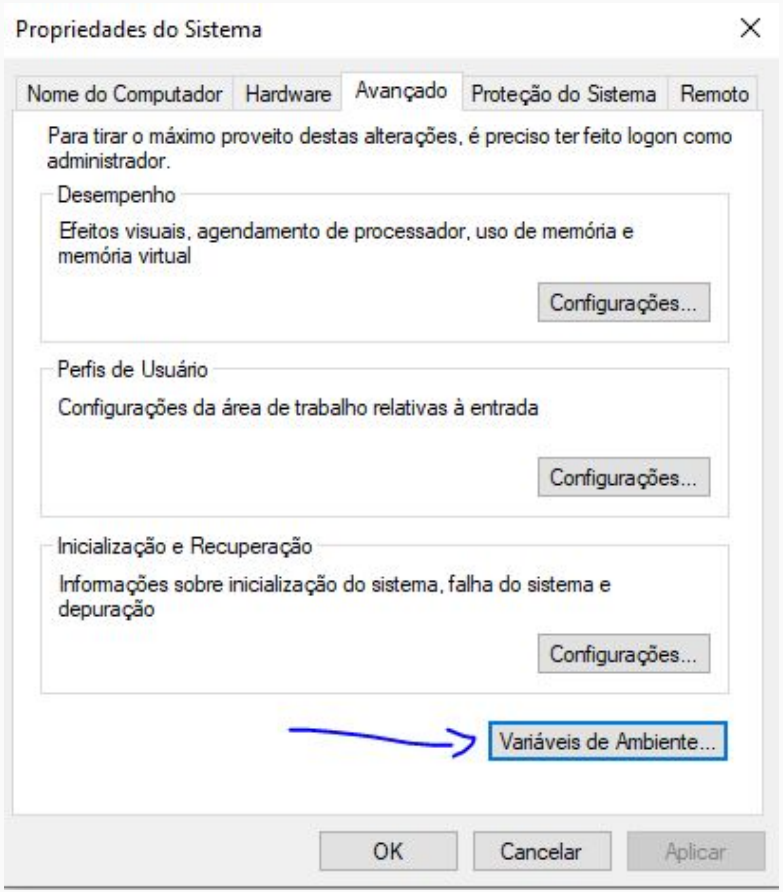
Configuração do MongoDB

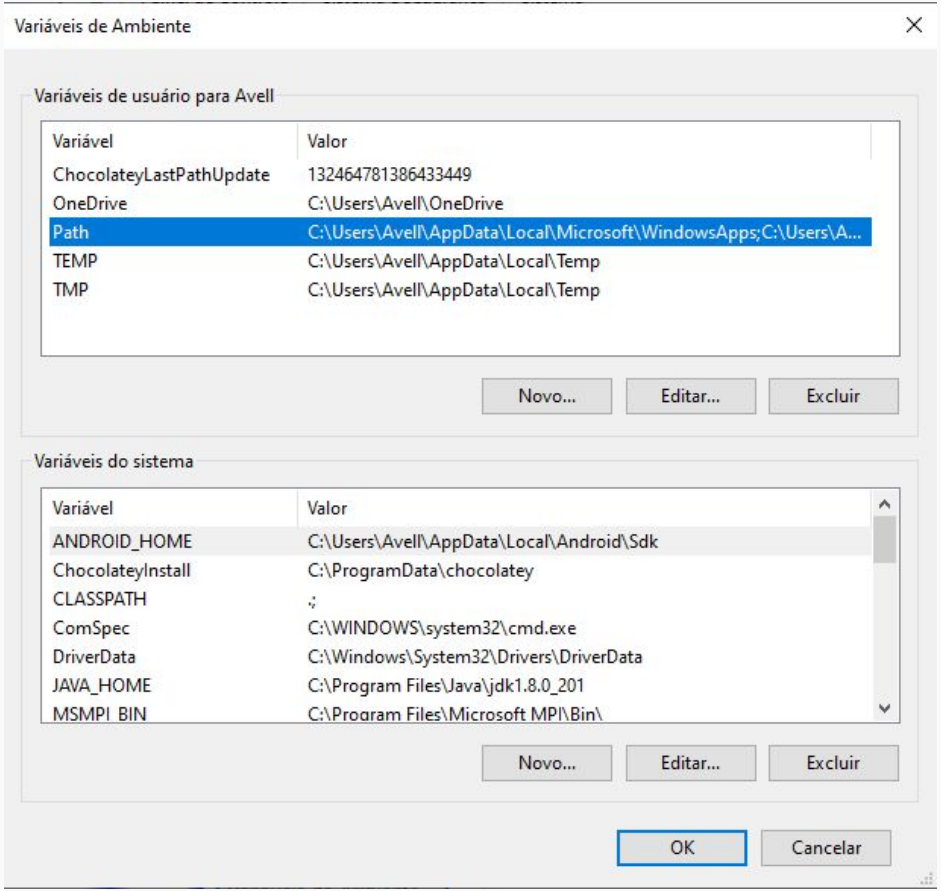


Configuração do MongoDB

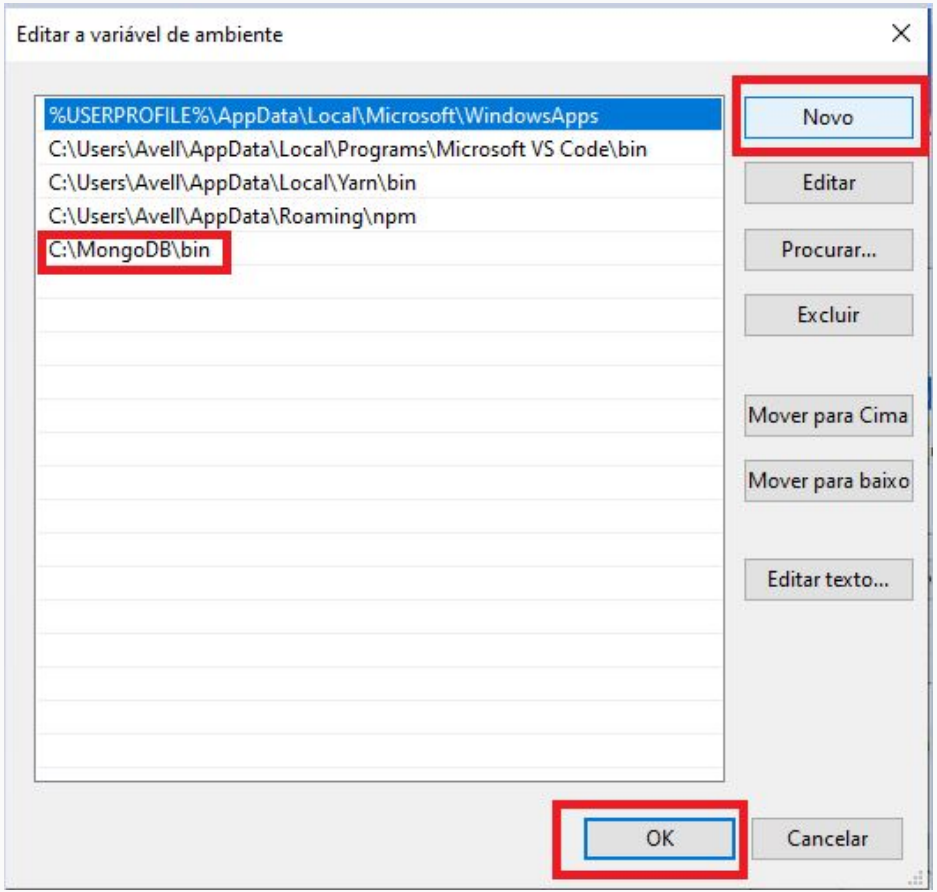


Configuração do MongoDB



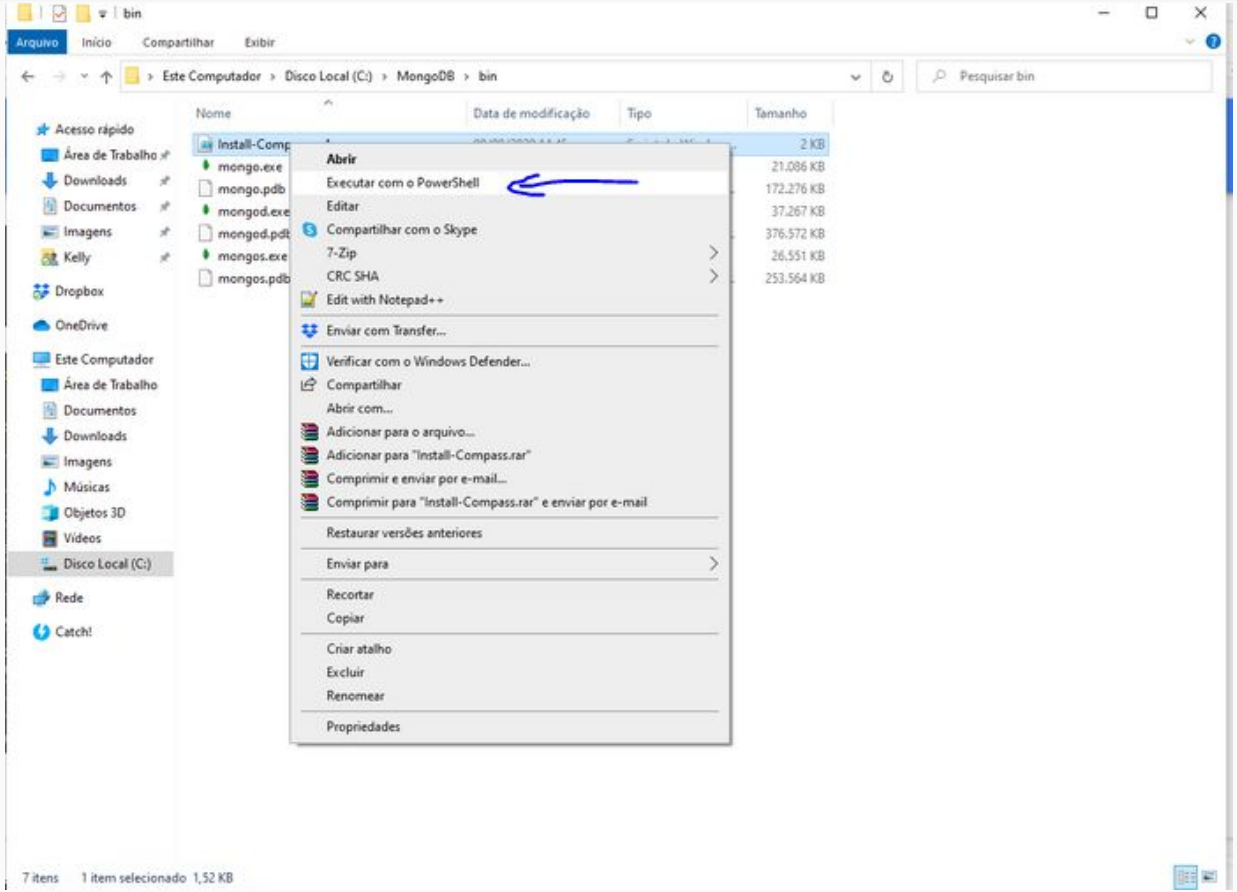


Configuração do MongoDB



Fechar e abrir o CMD :)

Instalação do MongoDB Compass



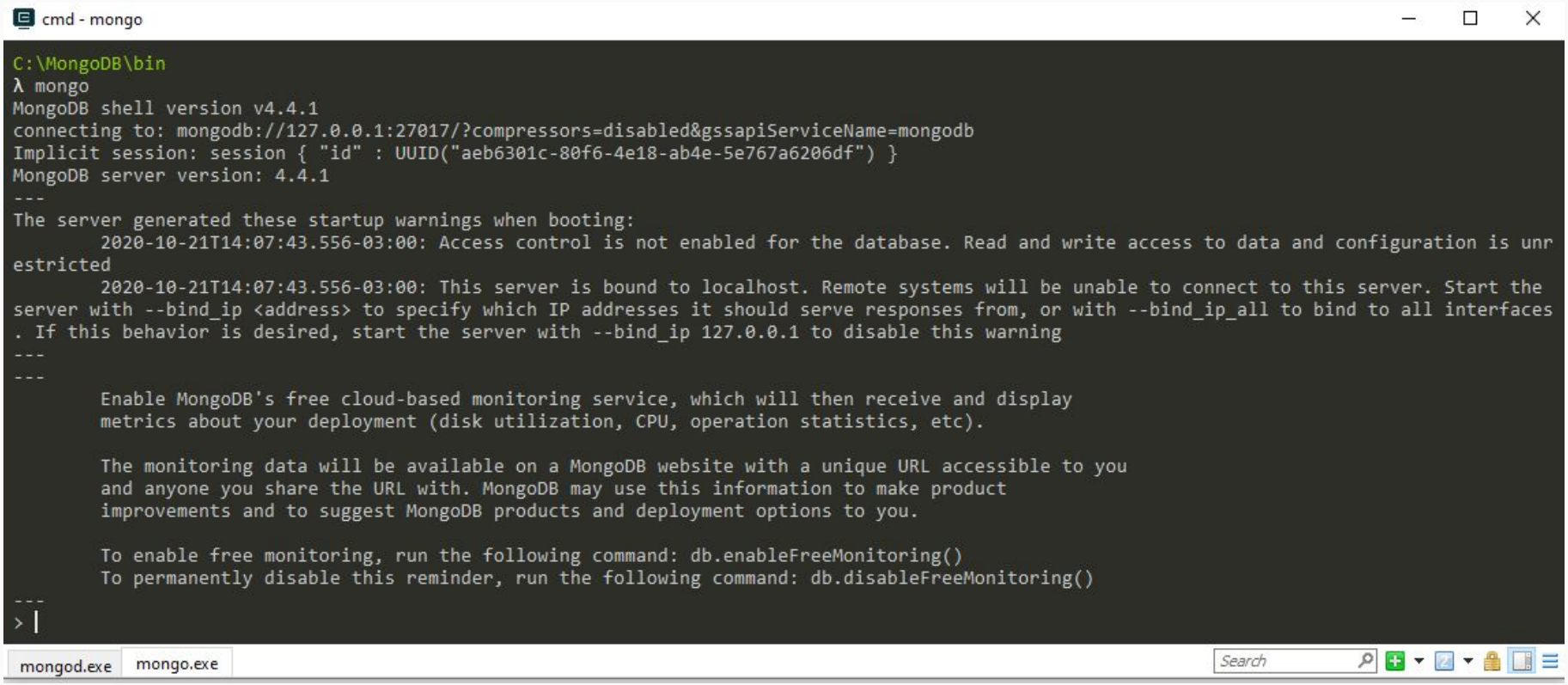
CRUD no MongoDB

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":{"$date":"2020-10-21T14:07:42.754-03:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TL
S 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2020-10-21T14:07:43.167-03:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No TransportLayer configur
ed during NetworkInterface startup"}
{"t":{"$date":"2020-10-21T14:07:43.167-03:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in u
se."}
{"t":{"$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","enviro
nment":{"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 co
mmand line","attr":{"options":{}}}
```

Testar o MongoDB



CRUD no MongoDB

Create

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

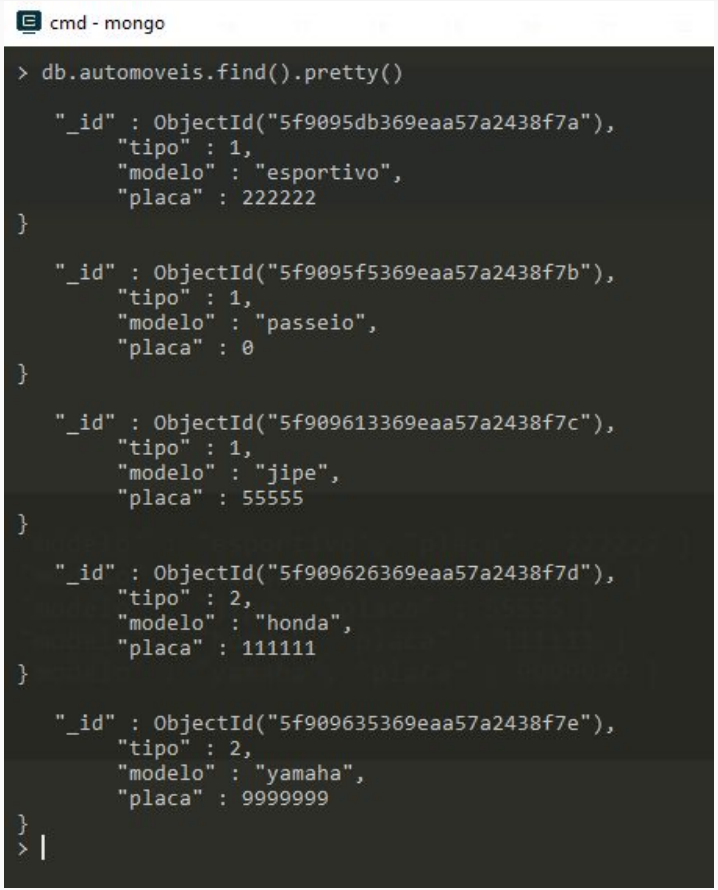
CRUD no MongoDB

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



CRUD no MongoDB

Update

Type	Command	Description
Update/Replace	`.update({query}, { $set: {query} }, options)`	The first argument is used as the query to find the document. The second argument specifies which field which to update. Exclude `$set:` and the entire document will be Replaced. Common options: `upsert: <boolean>` to keep it unique, and `multi: <boolean>` will update multiple documents if set to true. Docs and Field Operator Docs
Update One/Many	`.updateOne()` and `.updateMany()`	Basically the same as the above function, except the `multi: <boolean>` option is basically defaulted to `false` and `true`, and isnt' 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	`.deleteOne({query})`	Deletes the first document that matches the query. Recommend to search by `_id` or another unique field. Doc
Delete Many/All	`.deleteMany({query})`	Deletes all records that match the query. Leave the query blank to delete all documents. Doc

cmd - mongo

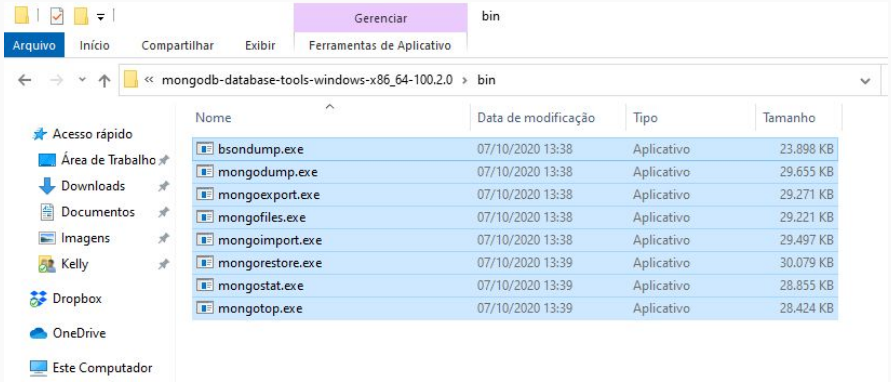
```
> db.automoveis.deleteOne({modelo: "passeio"})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

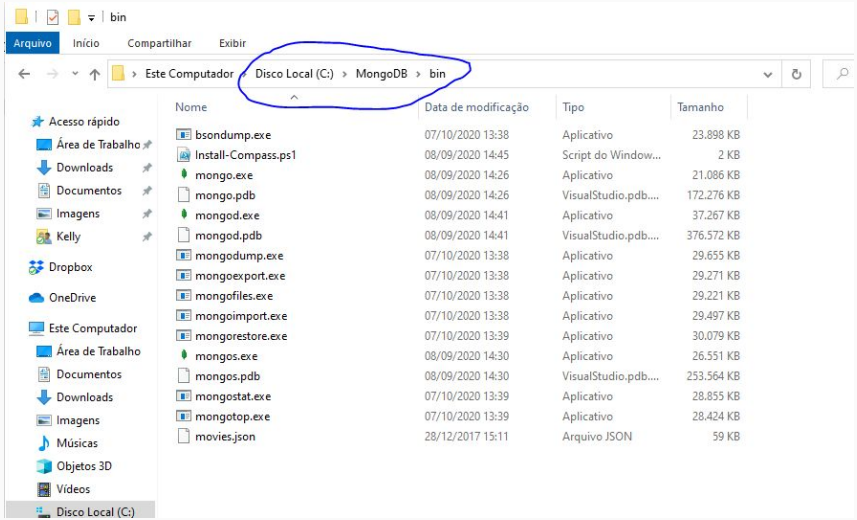
https://gist.github.com/michaeltreat/d3bdc989b54cff969df86484e091fd0c

Importar Dados para o MongoDB

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

Copiar





Importar Dados para o MongoDB

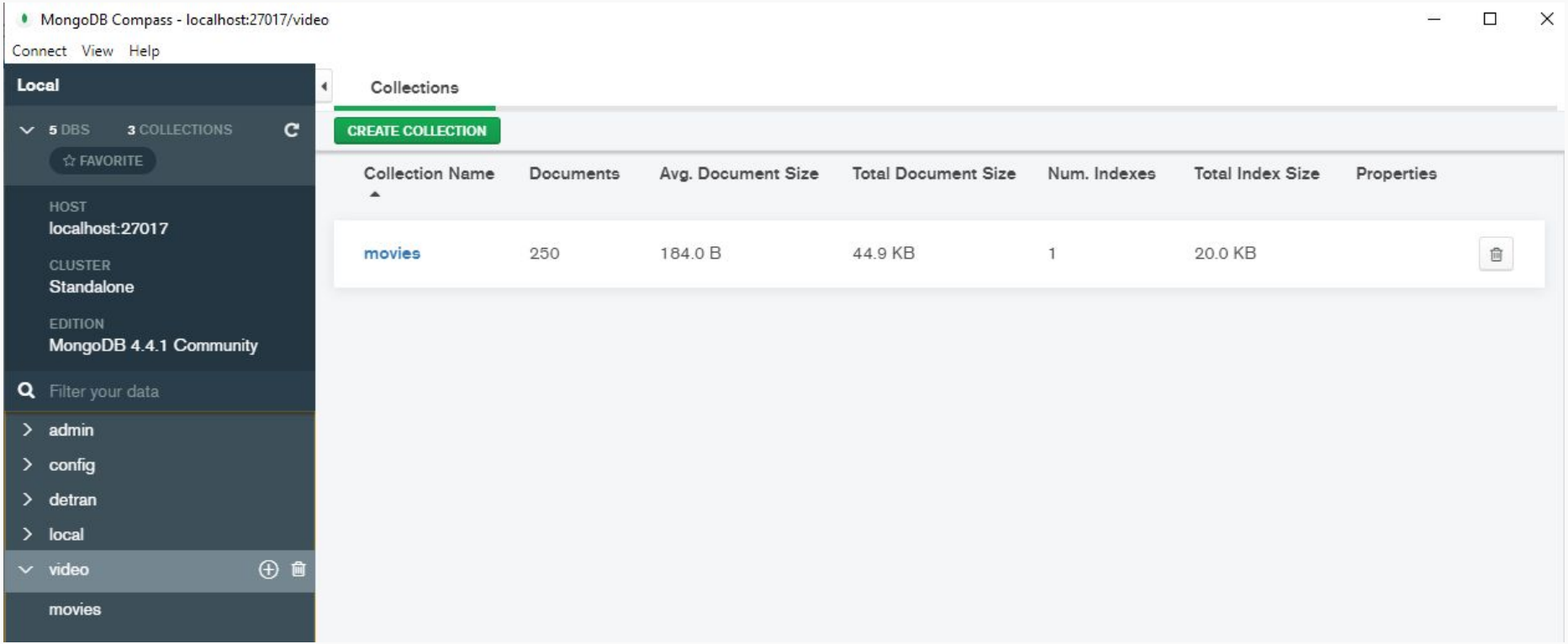
cmd

```
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
```

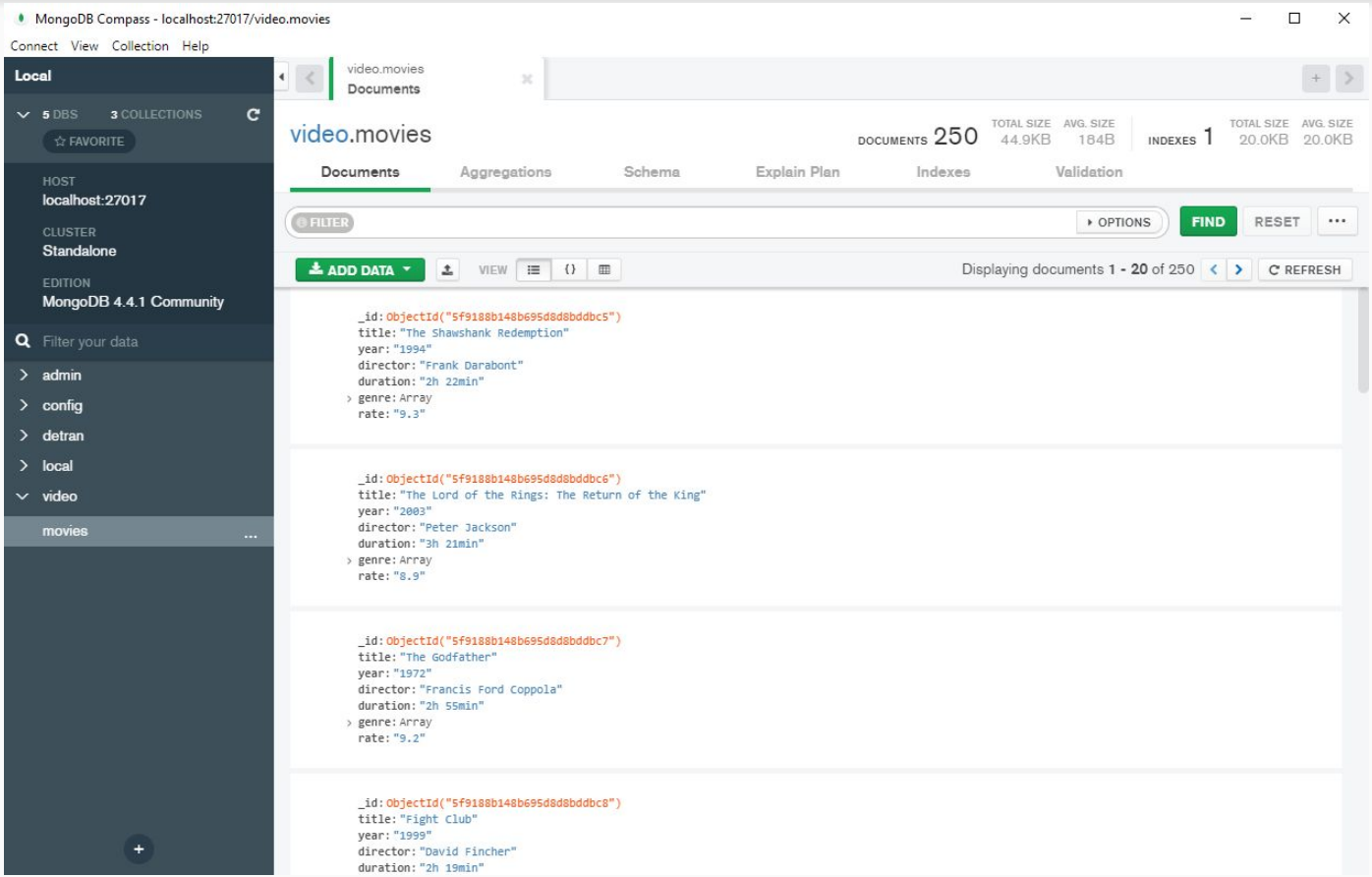
```
C:\MongoDB\bin
λ mongoimport --db video --collection movies --file movies.json --jsonArray
```

```
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.
```

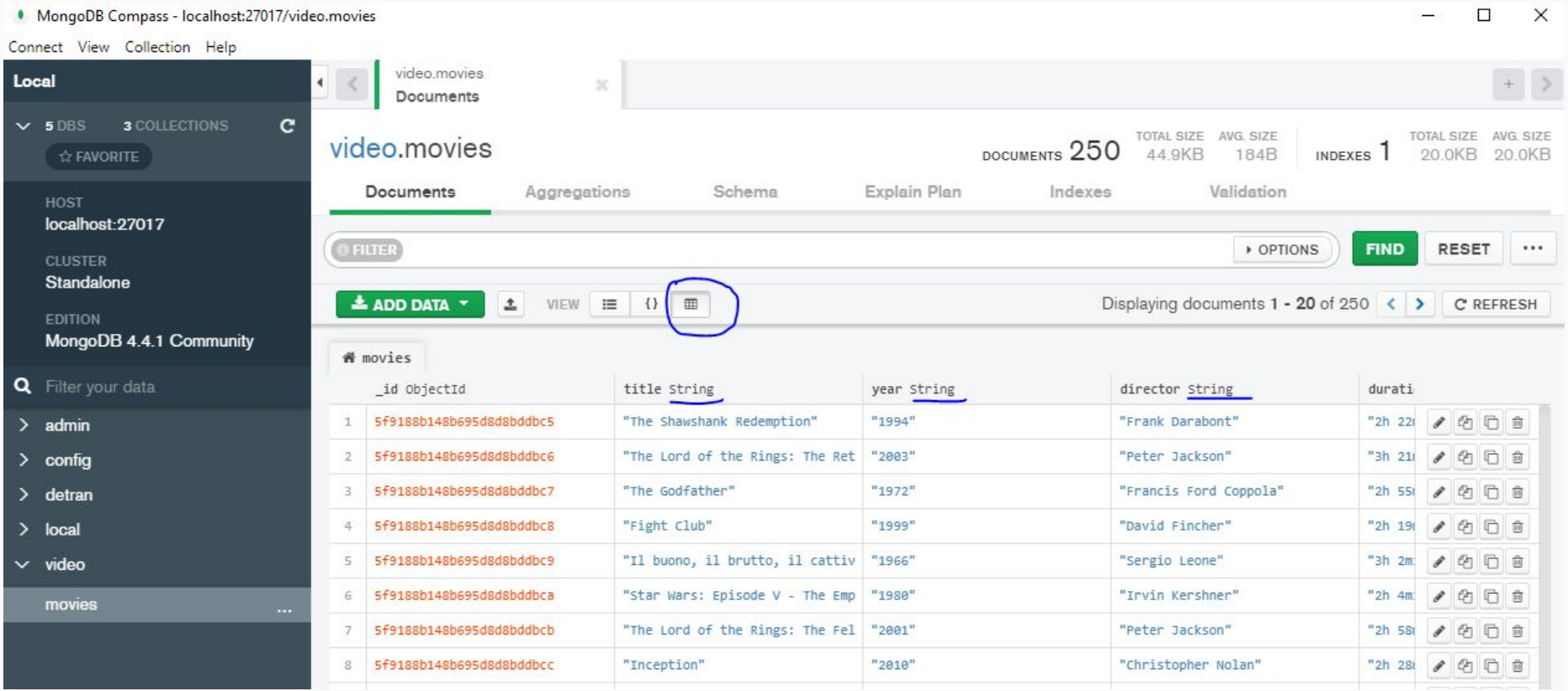
Importar Dados para o MongoDB



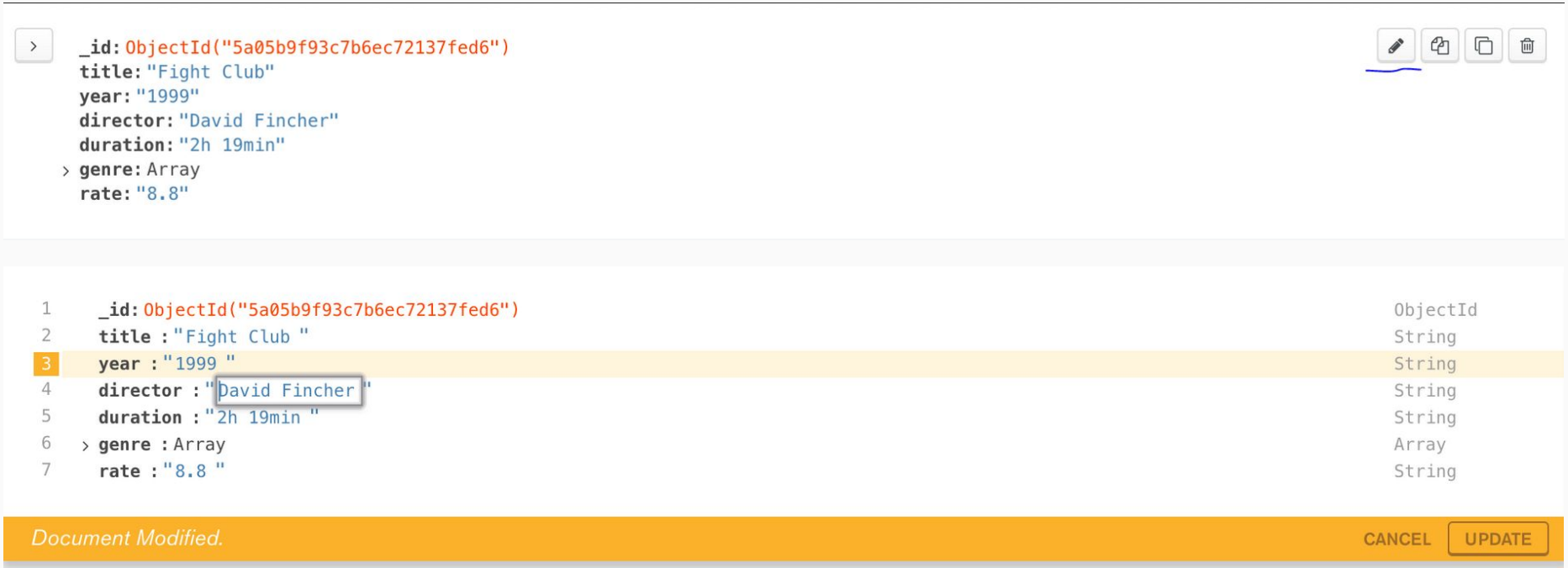
Importar Dados para o MongoDB



Tipo de dados



Editando Dados



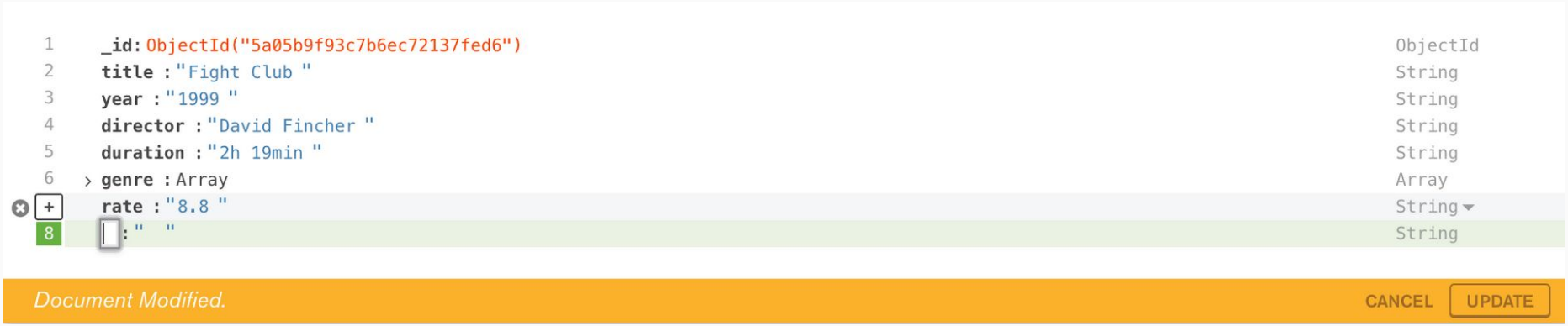
Deletando Campos

```
1   _id: ObjectId("5a05b9f93c7b6ec72137fed6")          ObjectId
2   title : "Fight Club "                              String
    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
    duration : "2h 19min "                             String
6 › genre : Array                                      Array
7   rate : "8.8 "                                      String
```

Document Modified. CANCEL UPDATE

Adicionando Campos



Deletando documentos

Filtros básicos

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 { 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

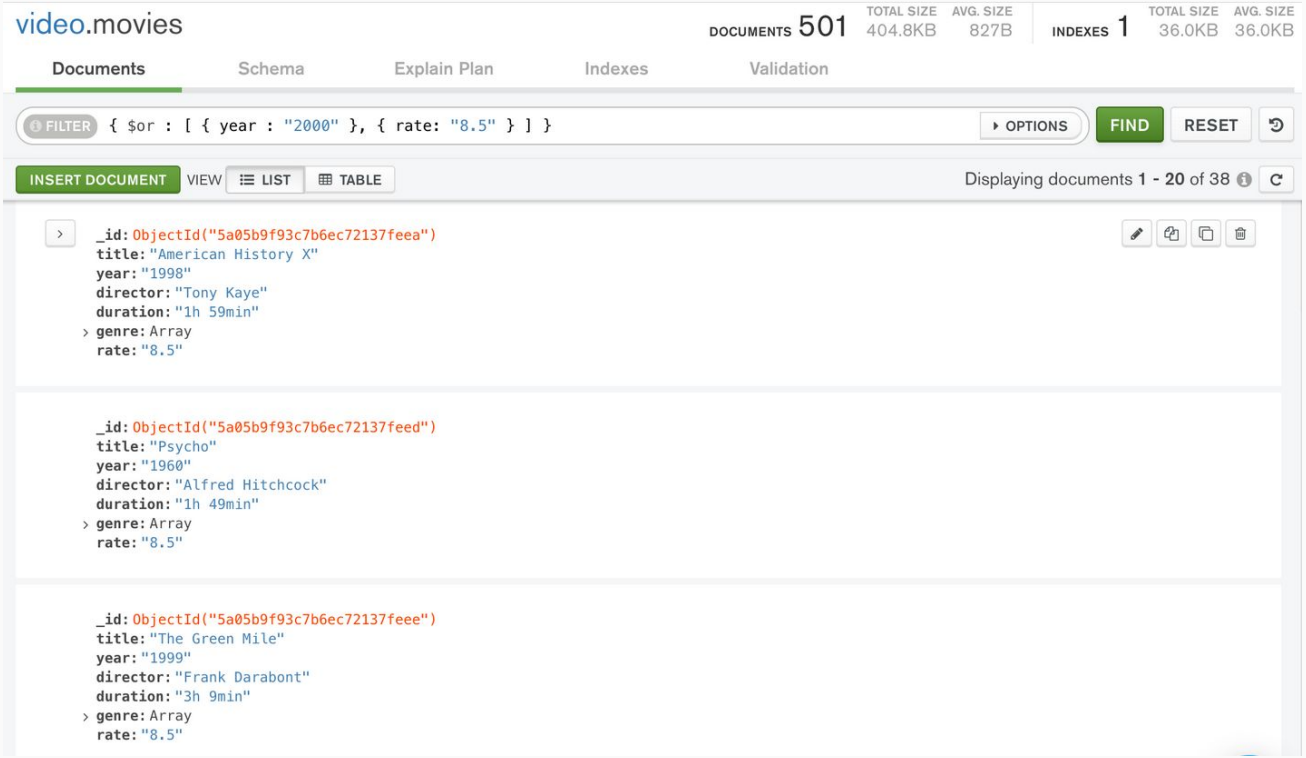
INSERT DOCUMENT VIEW LIST TABLE Displaying documents 1 - 2 of 2

```
_id: ObjectId("5a05b9f93c7b6ec72137fefa")
title: "Gladiator"
year: "2000"
director: "Ridley Scott"
duration: "2h 35min"
› genre: Array
rate: "8.5"
```

```
_id: ObjectId("5a05b9f93c7b6ec72137fefd")
title: "Memento"
year: "2000"
director: "Christopher Nolan"
duration: "1h 53min"
› genre: Array
rate: "8.5"
```

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

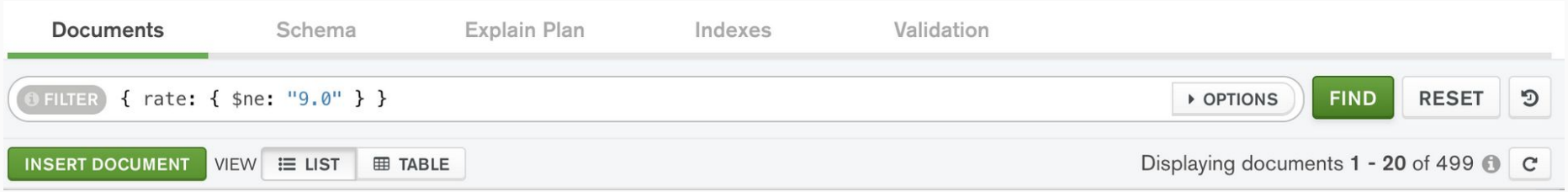
```
{ $or: [ { <expression1> }, { <expression2> }, ...  { <expressionN> } ] }
```



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

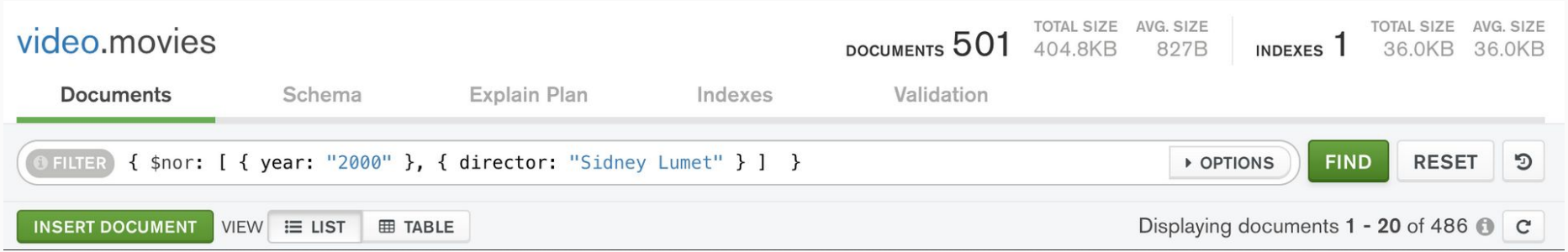
Not equal -> inclui documentos que não contém o valor informado.

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



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

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



Operadores de comparação

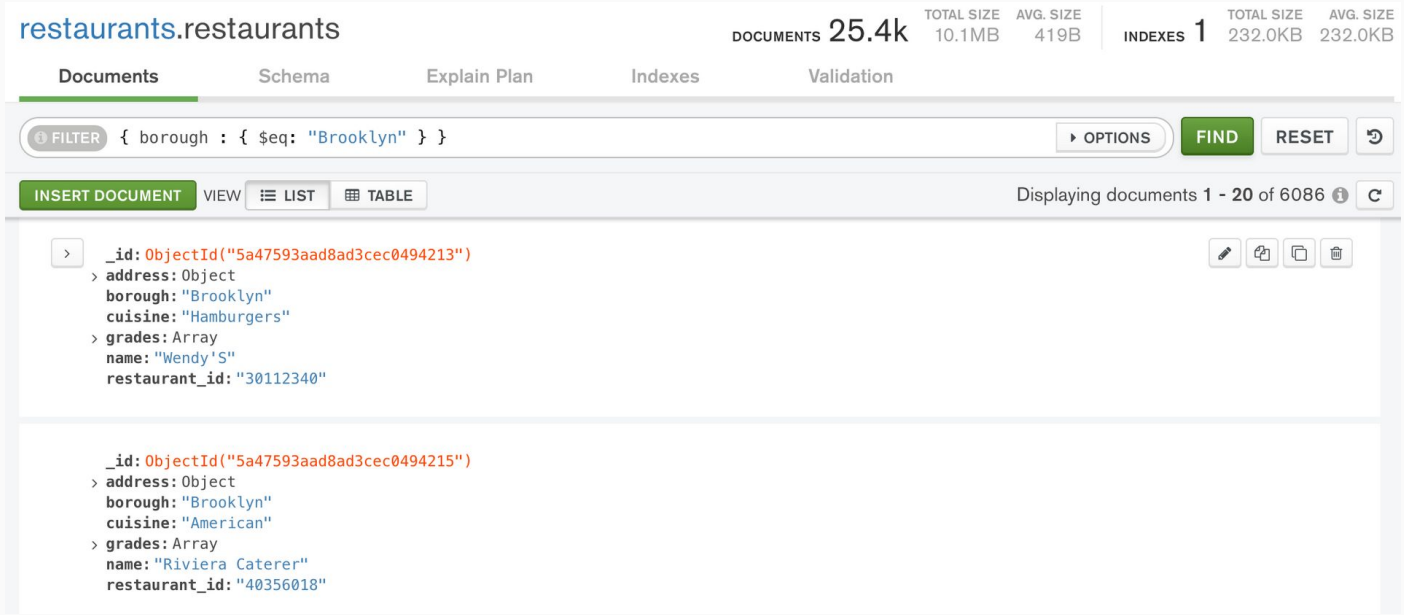
```
C:\MongoDB\bin
λ mongoimport --db restaurants --collection restaurants --drop --file primer-dataset.json
2020-10-22T11:55:24.466-0300    connected to: mongodb://localhost/
2020-10-22T11:55:24.499-0300    dropping: restaurants.restaurants
2020-10-22T11:55:25.285-0300    25359 document(s) imported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

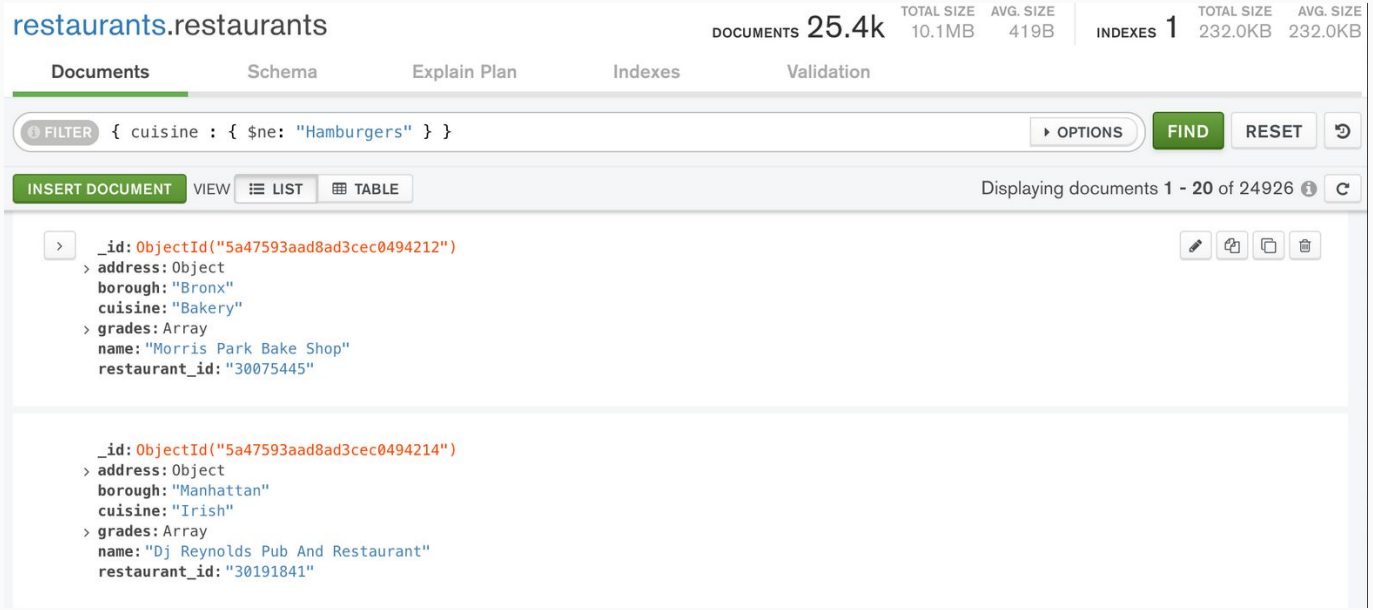
```
{ borough : { $eq: "Brooklyn" } }
```



Operadores de comparação: $ne - Not Equal

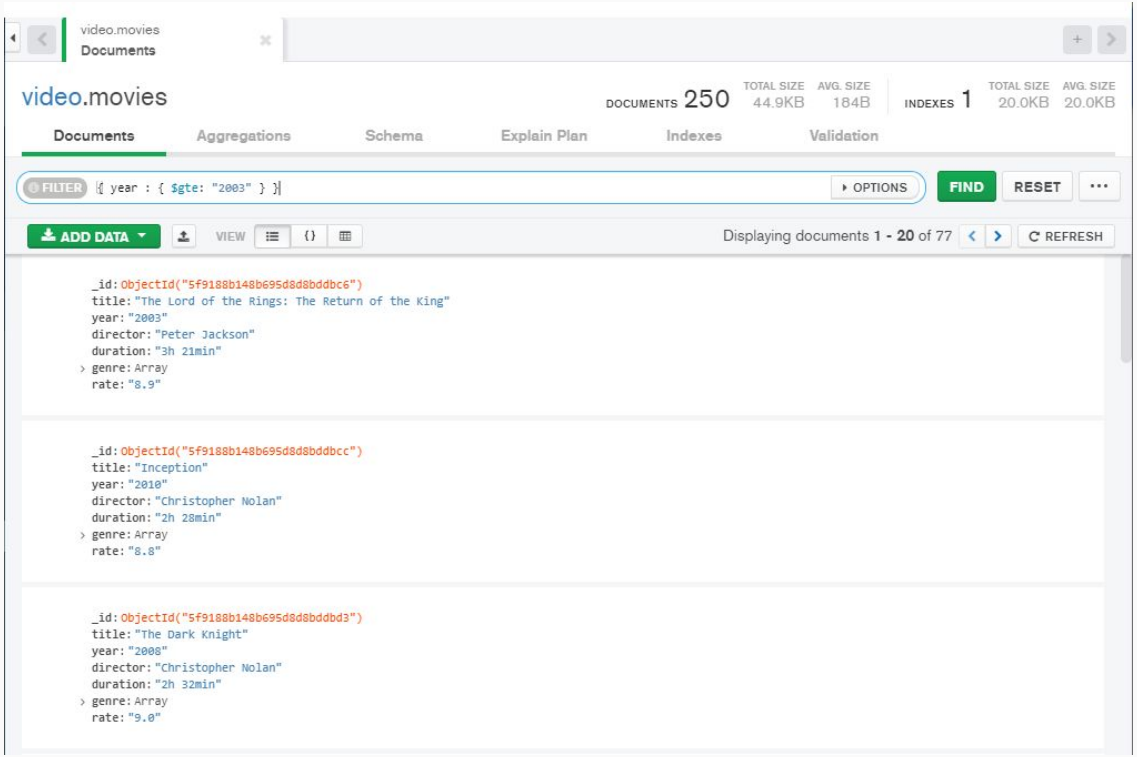
```
{ field: { $ne: value } }
```

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



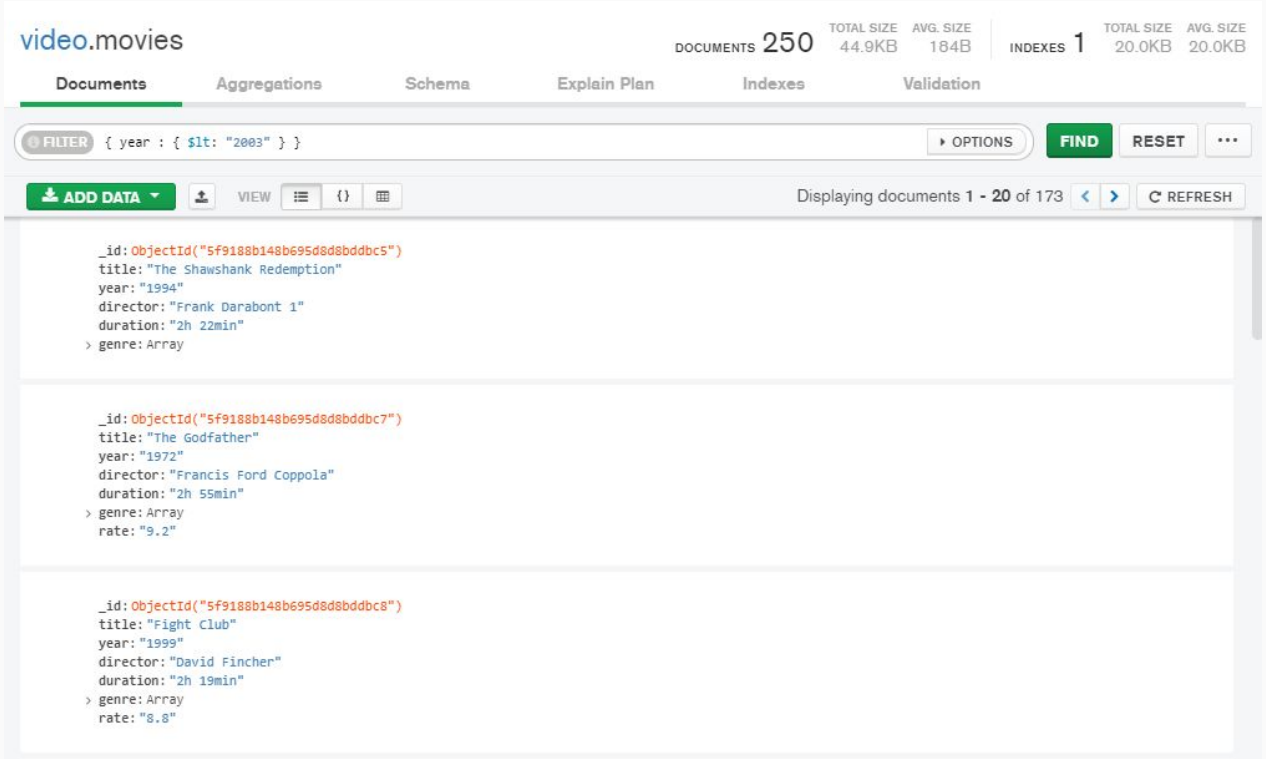
Operadores de comparação: $gt - greater than / $gt - greater than equal

```
{ field: { $gt: value } }
```



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

```
{ field: { $lt: value } }
```

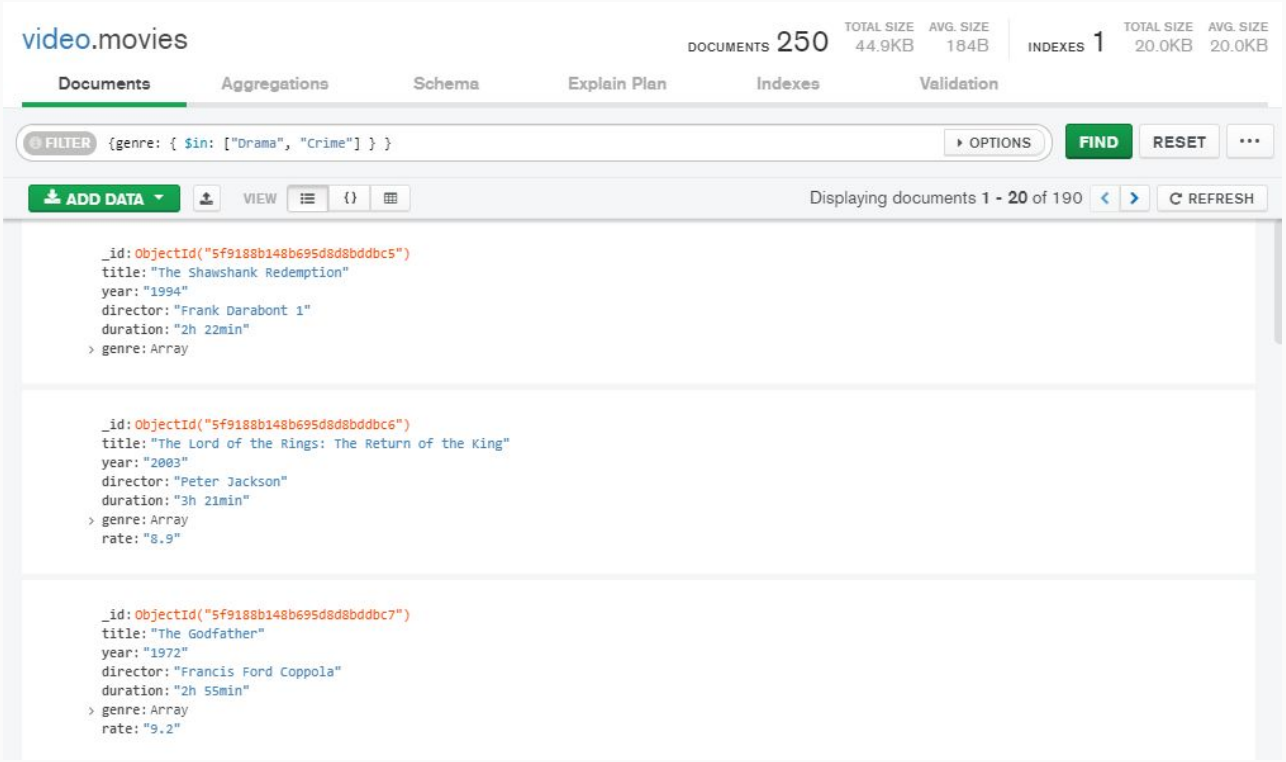


Operadores de comparação

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 } }`

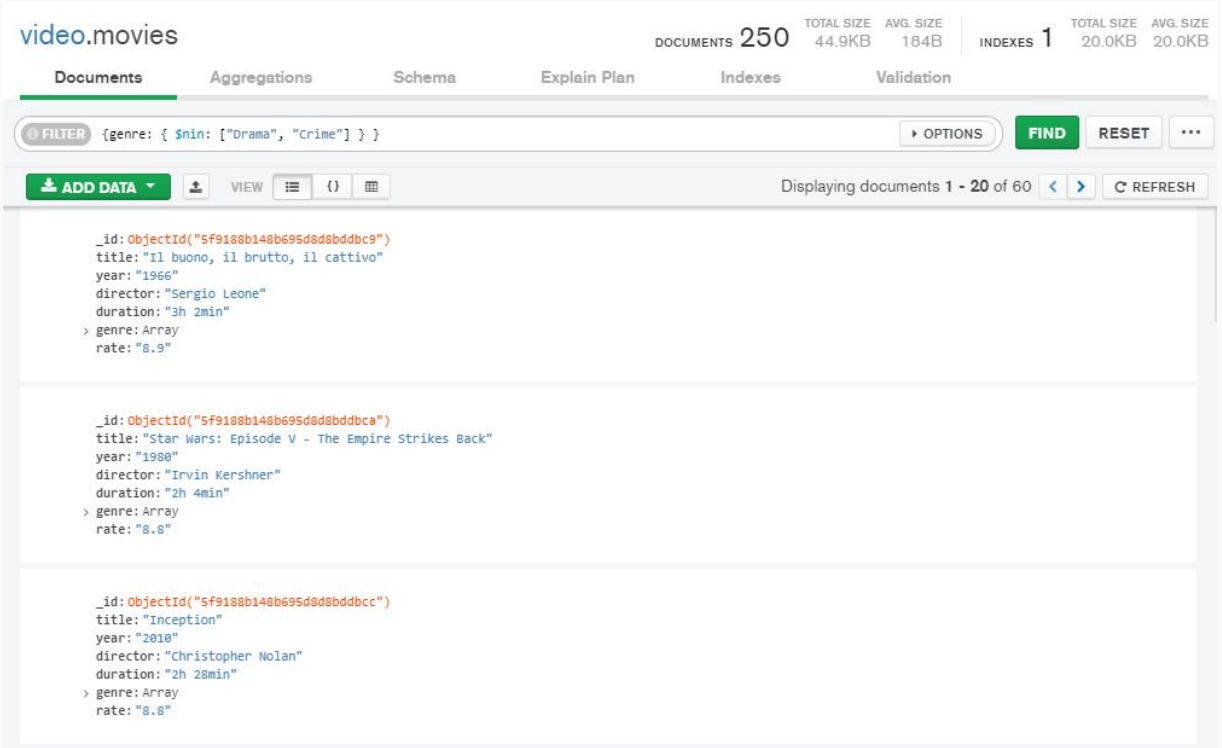
Operações em Array: $in

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



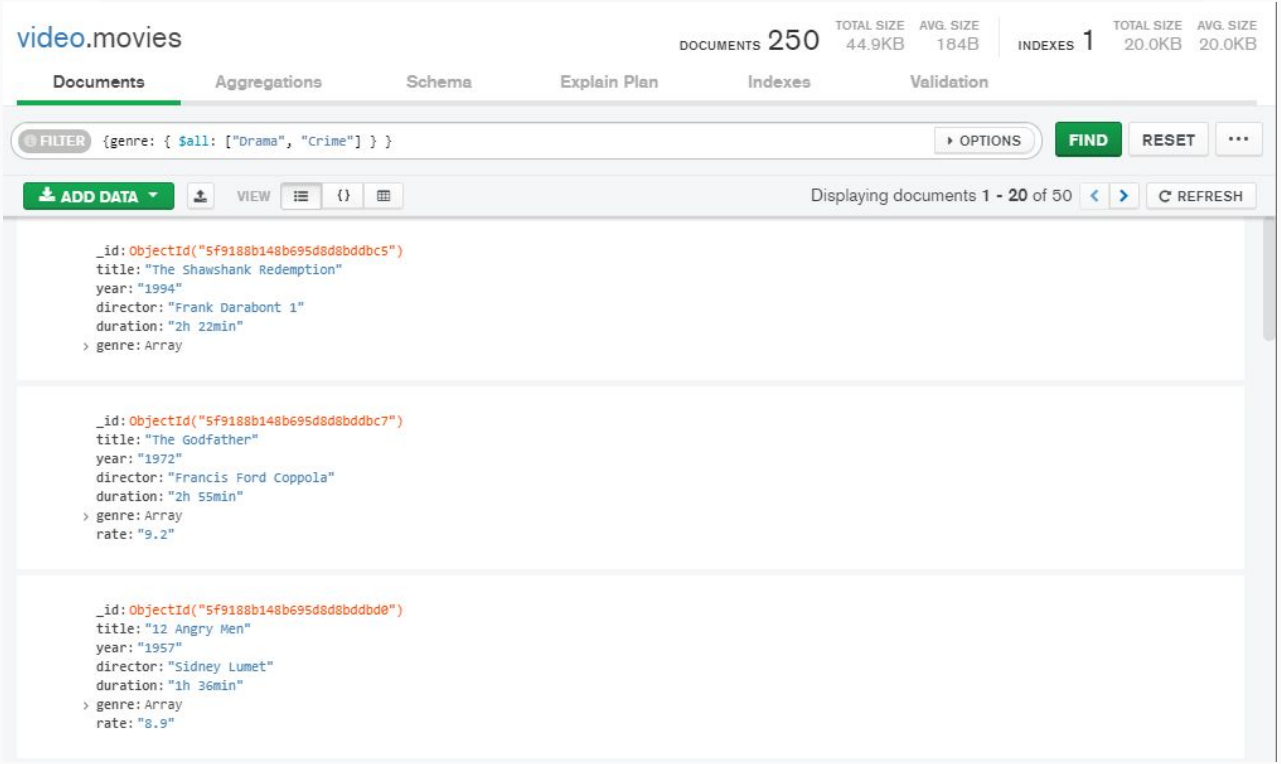
Operações em Array: $nin

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



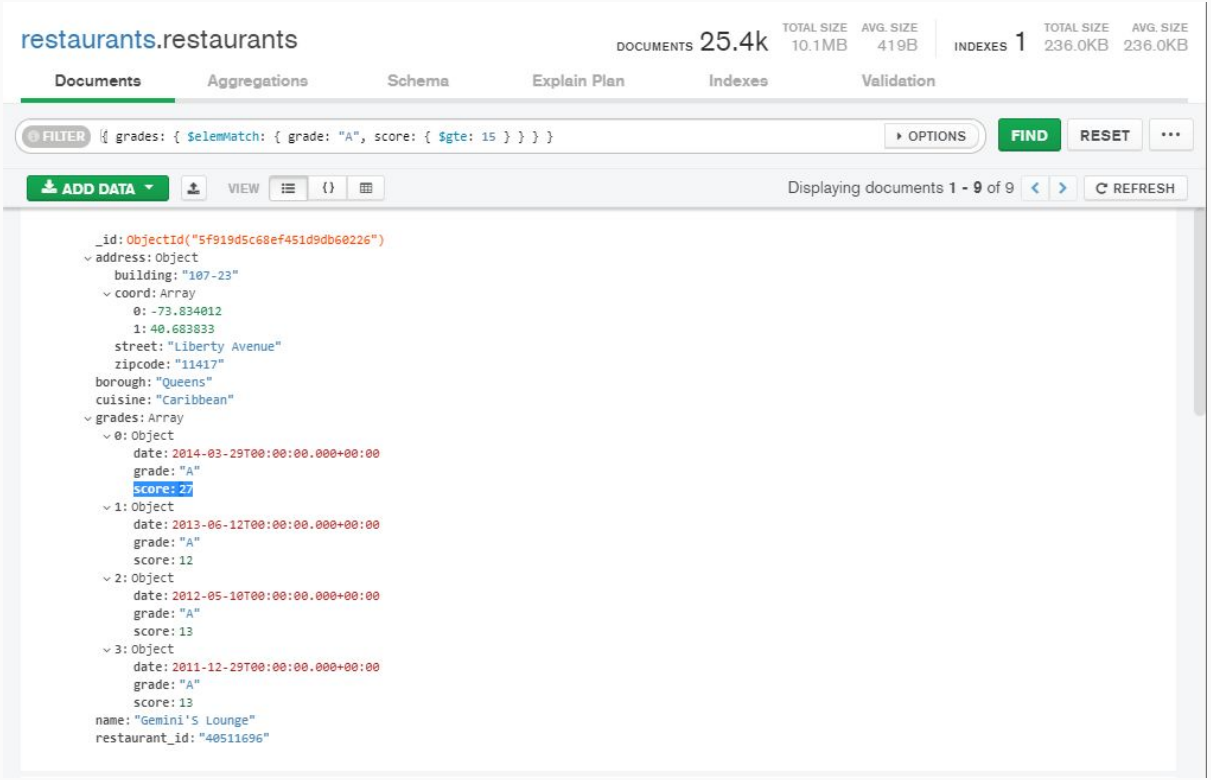
Operações em Array: $all

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



Operações em Array: $all

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



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.

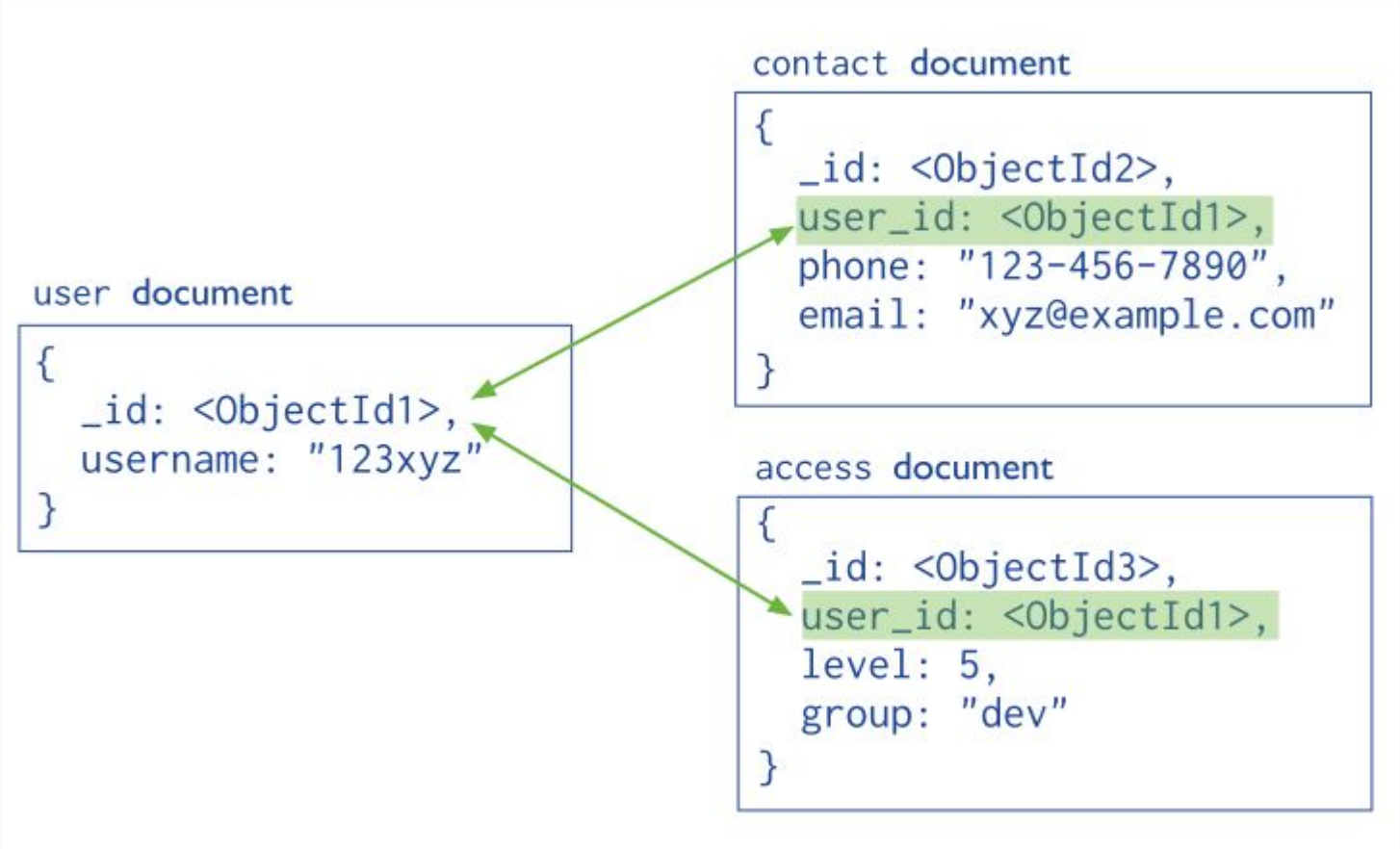
Ao projetar o modelo de dados de um Banco de Dados, devemos considerar os tipos de dados que usaremos na aplicação. A chave é compreender a estrutura dos documentos e como são representadas as relações entre eles.

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: "Faketon",
    state: "MA",
    zip: "12345"
}
```

MongoDB - Embedding documents

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: "Faketon",
    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

Object Document Mapper (ODM)

Setup

```
$ mkdir mongoose-example

$ cd mongoose-example

$ npm init

$ npm install mongoose

$ touch example.js
```

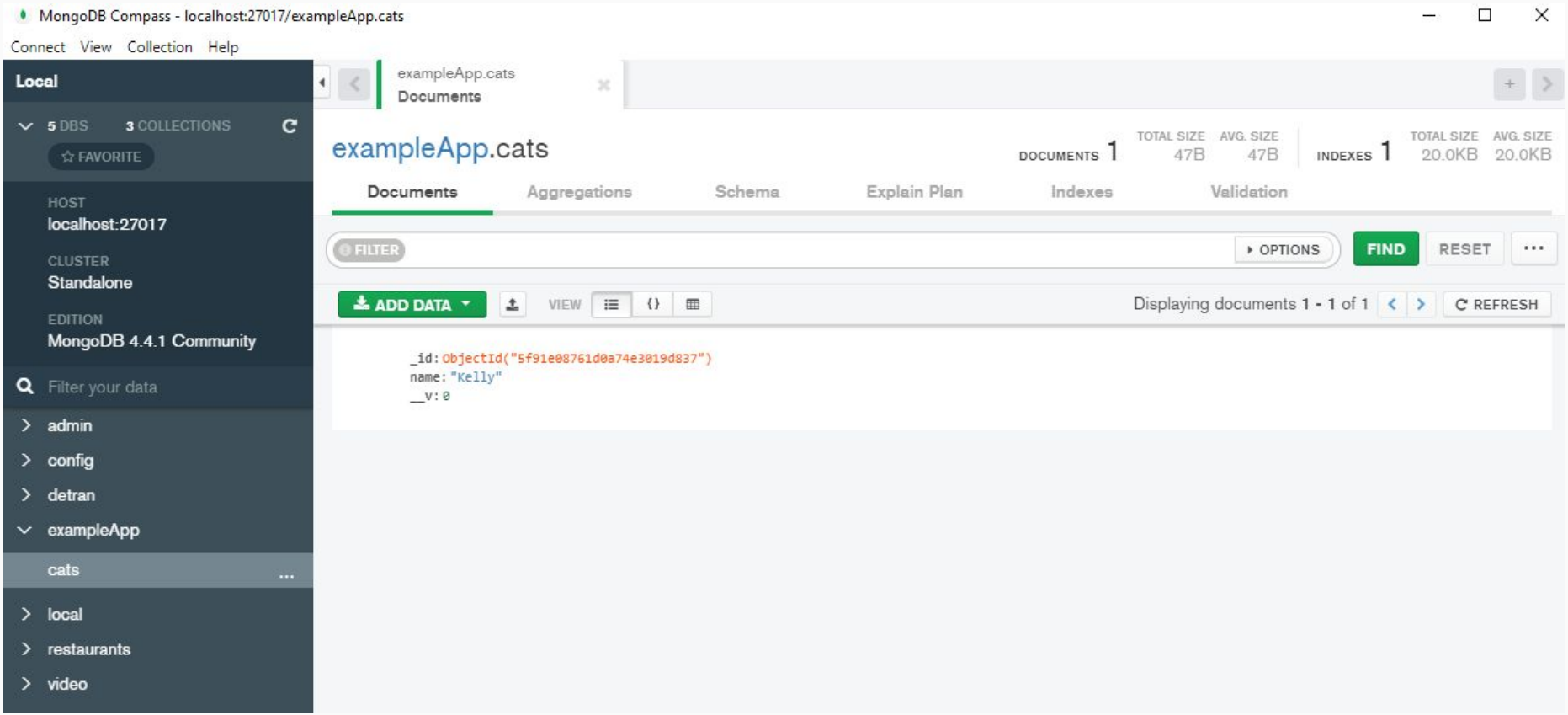
Connecting to the Database

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}`));
```


MongoDB Compass - localhost:27017/exampleApp.cats
Connect View Collection Help
Local
5 DBS 3 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.1 Community
Filter your data
admin
config
detran
exampleApp
cats
local
restaurants
video
exampleApp.cats
Documents
exampleApp.cats
DOCUMENTS 1
TOTAL SIZE 47B
AVG. SIZE 47B
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 - 1 of 1
REFRESH
_id: ObjectId("5f91e08761d0a74e3019d837")
name: "Kelly"
__v: 0