



BANCO DE DADOS NÃO-RELACIONAL

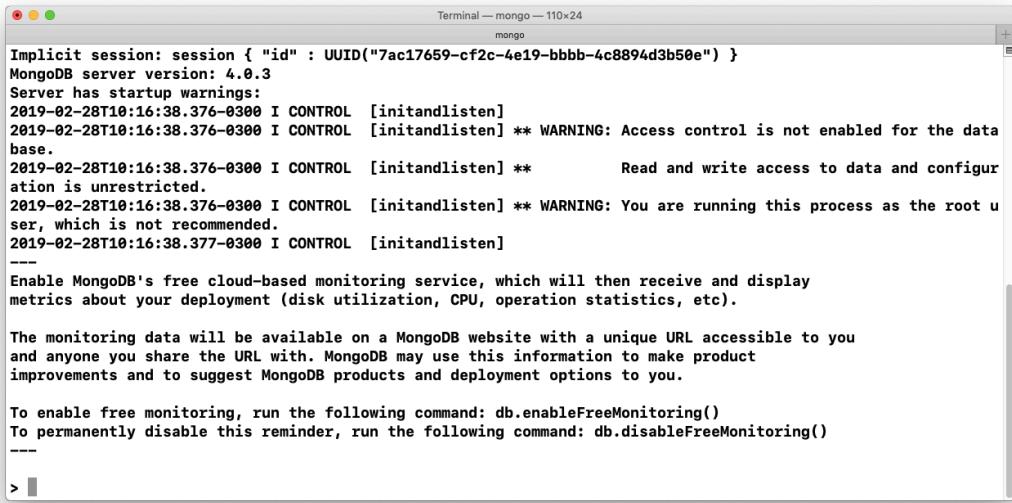
mongoDB 2

Banco Relacional x mongoDB

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
tables	collections
rows	documents (BSON)
columns	fields

Shell do MongoDB

- Antes de executar o shell (**mongo**) do SGBD **mongoDB** é necessário subir o servidor **mongod**. Em seguida, lançar o shell, sendo o prompt denotado por '**>**'. Agora estamos prontos para manipular o banco de dados.



```

Implicit session: session { "id" : UUID("7ac17659-cf2c-4e19-bbbb-4c8894d3b50e") }
MongoDB server version: 4.0.3
Server has startup warnings:
2019-02-28T10:16:38.376-0300 I CONTROL [initandlisten]
2019-02-28T10:16:38.376-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the data
base.
2019-02-28T10:16:38.376-0300 I CONTROL [initandlisten] ** Read and write access to data and configur
ation is unrestricted.
2019-02-28T10:16:38.376-0300 I CONTROL [initandlisten] ** WARNING: You are running this process as the root u
ser, which is not recommended.
2019-02-28T10:16:38.377-0300 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()
---

> 

```

Criar um novo usuário

- O próximo passo (opcional) será criar um usuário na base de dados e definir os privilégios (*roles*). Para isto vamos executar o seguinte comando:



```

Terminal — mongo — 110x11
mongo
> cls
> db.createUser({user:"marcio", pwd:"uemg", roles:["readWrite", "dbAdmin"]})
Successfully added user: { "user" : "marcio", "roles" : [ "readWrite", "dbAdmin" ] }
>

```

→ Para limpar a tela do terminal do shell, execute o comando **cls**.

Criar um novo banco de dados

- Não há um comando como em **SQL CREATE DATABASE**. Para criar um banco de dados em mongoDB é suficiente apenas confirmar o nome do atual e inserir os dados.

✓ O banco será criado somente quando um dado é inserido ou uma *collection* criada.

```
Terminal — mongo — 73x19
mongo
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
> use bd2
switched to db bd2
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
> db.createCollection("Veiculos")
{ "ok" : 1 }
> show dbs
admin 0.000GB
bd2 0.000GB
config 0.000GB
local 0.000GB
>
```

Criar uma Coleção (Collection)

- Há duas maneiras de criar uma coleção: por meio do método **createCollection** (slide anterior) ou pela inserção de um documento.



```
Terminal — mongo — 112x7
mongo
> db.Veiculos.insert({"Placa":"IOS-0078", "Fabricante":"Renault", "Marca":"Sandeiro", "Ano":2009, "Cor":"Vermelho"})
WriteResult({ "nInserted" : 1 })
>
```

Operações de CRUD em MongoDB

C db.users.insertOne(← collection
 { ← field: value
 name: "sue", ← field: value
 age: 26, ← field: value
 status: "pending" } document
)

R db.users.find(← collection
 { age: { \$gt: 18 } }, ← query criteria
 { name: 1, address: 1 } ← projection
).limit(5) ← cursor modifier

U db.users.updateMany(← collection
 { age: { \$lt: 18 } }, ← update filter
 { \$set: { status: "reject" } } ← update action
)

D db.users.deleteMany(← collection
 { status: "reject" } ← delete filter
)

Método insert()

- Para inserir um documento há o método **insert()** (slide anterior) e dois outros:

- **insertOne()**
- **insertMany()**

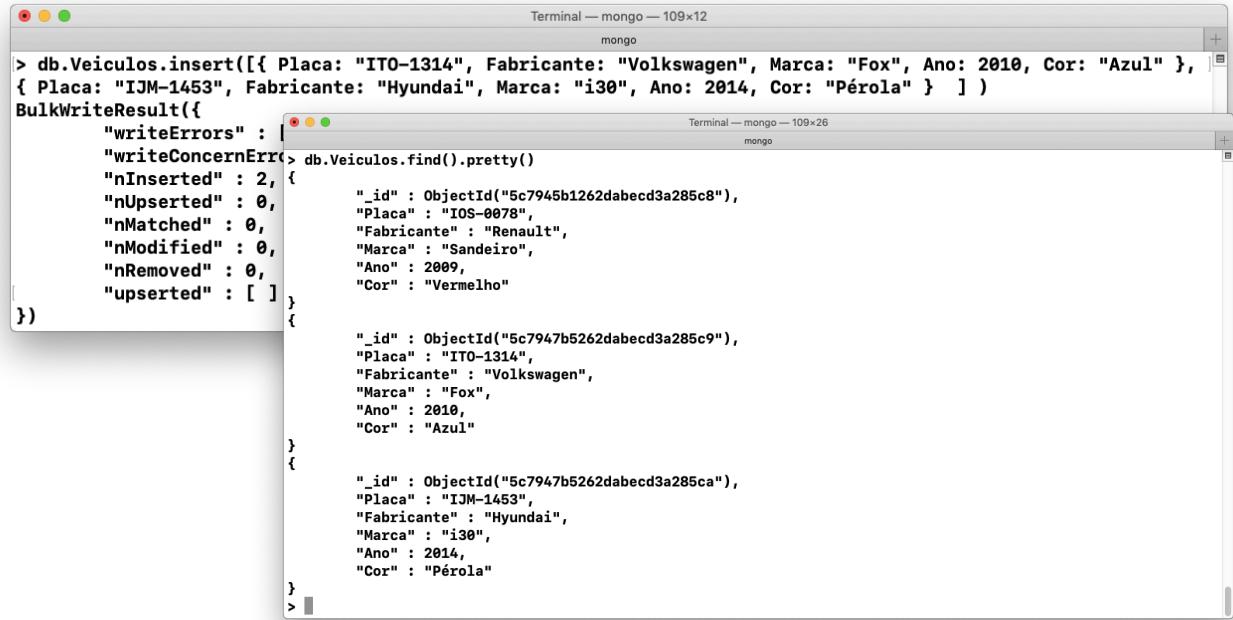


```
Terminal — mongo — 95x13
mongo
> db.Veiculos.find()
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault",
"Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Vermelho" }
> db.Veiculos.find().pretty()
{
    "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
    "Placa" : "IOS-0078",
    "Fabricante" : "Renault",
    "Marca" : "Sandeiro",
    "Ano" : 2009,
    "Cor" : "Vermelho"
}
>
```

- ✓ Nota-se que MongoDB adicionou um novo campo ao documento: **_id**. Você também pode definir o seu **_id**. Pode-se pensar neste campo como uma chave primária.

Métodos insert(), insertOne(), insertMany()

- Pode-se inserir **múltiplos documentos** em uma coleção com um único método **insert()**.



```

Terminal — mongo — 109x12
mongo
> db.Veiculos.insert([
  { Placa: "ITO-1314", Fabricante: "Volkswagen", Marca: "Fox", Ano: 2010, Cor: "Azul" },
  { Placa: "IJM-1453", Fabricante: "Hyundai", Marca: "i30", Ano: 2014, Cor: "Pérola" }
])
BulkWriteResult({
  "writeErrors" : [],
  "writeConcernError" : null,
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})

Terminal — mongo — 109x26
mongo
> db.Veiculos.find().pretty()
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
  "Placa" : "IOS-0078",
  "Fabricante" : "Renault",
  "Marca" : "Sandeiro",
  "Ano" : 2009,
  "Cor" : "Vermelho"
}
{
  "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
  "Placa" : "ITO-1314",
  "Fabricante" : "Volkswagen",
  "Marca" : "Fox",
  "Ano" : 2010,
  "Cor" : "Azul"
}
{
  "_id" : ObjectId("5c7947b5262dabecd3a285ca"),
  "Placa" : "IJM-1453",
  "Fabricante" : "Hyundai",
  "Marca" : "i30",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
>

```

Consulta de uma coleção

- **db.collection.find()** seleciona documentos em uma coleção e retorna um cursor para os documentos selecionados the selected documents.



```

Terminal — mongo — 105x26
mongo
> db.Veiculos.find().pretty()
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
  "Placa" : "IOS-0078",
  "Fabricante" : "Renault",
  "Marca" : "Sandeiro",
  "Ano" : 2009,
  "Cor" : "Vermelho"
}
{
  "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
  "Placa" : "ITO-1314",
  "Fabricante" : "Volkswagen",
  "Marca" : "Fox",
}

Terminal — mongo — 105x11
mongo
> db.Veiculos.findOne()
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
  "Placa" : "IOS-0078",
  "Fabricante" : "Renault",
  "Marca" : "Sandeiro",
  "Ano" : 2009,
  "Cor" : "Vermelho"
}
>

```

Consulta de uma coleção: opções de filtro



The image shows two side-by-side terminal windows for MongoDB. The left window has a title bar 'Terminal — mongo — 80x19' and contains the command:

```
> db.Veiculos.find({Ano:<2014}).pretty()
```

The right window has a title bar 'Terminal — mongo — 87x19' and contains the command:

```
> db.Veiculos.find( { Fabricante: { $in: [ "Renault", "Volkswagen" ] } } ).pretty()
```

Both windows display the results of their respective queries.

Consulta de uma coleção: opções de filtro



The image shows two side-by-side terminal windows for MongoDB. The left window has a title bar 'Terminal — mongo — 105x19' and contains the command:

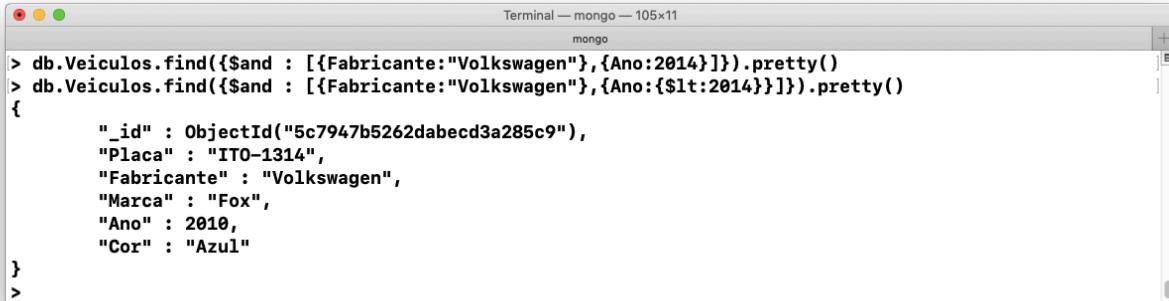
```
> db.Veiculos.find({$or : [{Fabricante:"Volkswagen"},{Fabricante:"Hyundai"}]}).pretty()
```

The right window has a title bar 'Terminal — mongo — 105x11' and contains the command:

```
> db.Veiculos.find({$or : [{Fabricante:"Volkswagen"},{Fabricante:"Hyundai"}]}).pretty()
```

Both windows display the results of their respective queries.

Consulta de uma coleção: opções de filtro



```
Terminal — mongo — 105x11
mongo
> db.Veiculos.find({$and : [{Fabricante:"Volkswagen"},{Ano:2014}]}).pretty()
> db.Veiculos.find({$and : [{Fabricante:"Volkswagen"},{Ano:{$lt:2014}}]}).pretty()
{
    "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
    "Placa" : "ITO-1314",
    "Fabricante" : "Volkswagen",
    "Marca" : "Fox",
    "Ano" : 2010,
    "Cor" : "Azul"
}
>
```

Operadores de Comparação em Consultas

Name	Description
<code>\$eq</code>	Matches values that are equal to a specified value.
<code>\$gt</code>	Matches values that are greater than a specified value.
<code>\$gte</code>	Matches values that are greater than or equal to a specified value.
<code>\$in</code>	Matches any of the values specified in an array.
<code>\$lt</code>	Matches values that are less than a specified value.
<code>\$lte</code>	Matches values that are less than or equal to a specified value.
<code>\$ne</code>	Matches all values that are not equal to a specified value.
<code>\$nin</code>	Matches none of the values specified in an array.

Operadores Lógicos e de Existências em Consultas

Name	Description
\$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
\$not	Inverts the effect of a query expression and returns documents that do <i>not</i> match the query expression.
\$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
\$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

Name	Description
\$exists	Matches documents that have the specified field.
\$type	Selects documents if a field is of the specified type.

Consultas de Projeção

```
> db.Veiculos.find({Fabricante:"Hyundai"}).pretty()
{
  "_id" : ObjectId("5c7947b5262dabecd3a285ca"),
  "Placa" : "IJM-1453",
  "Fabricante" : "Hyundai",
  "Marca" : "i30",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
{
  "_id" : ObjectId("5c7c1096262dabecd3a285cb"),
  "Placa" : "HOE-4973",
  "Fabricante" : "Hyundai",
  "Marca" : "Creta",
  "Ano" : 2018,
  "Cor" :
}
```

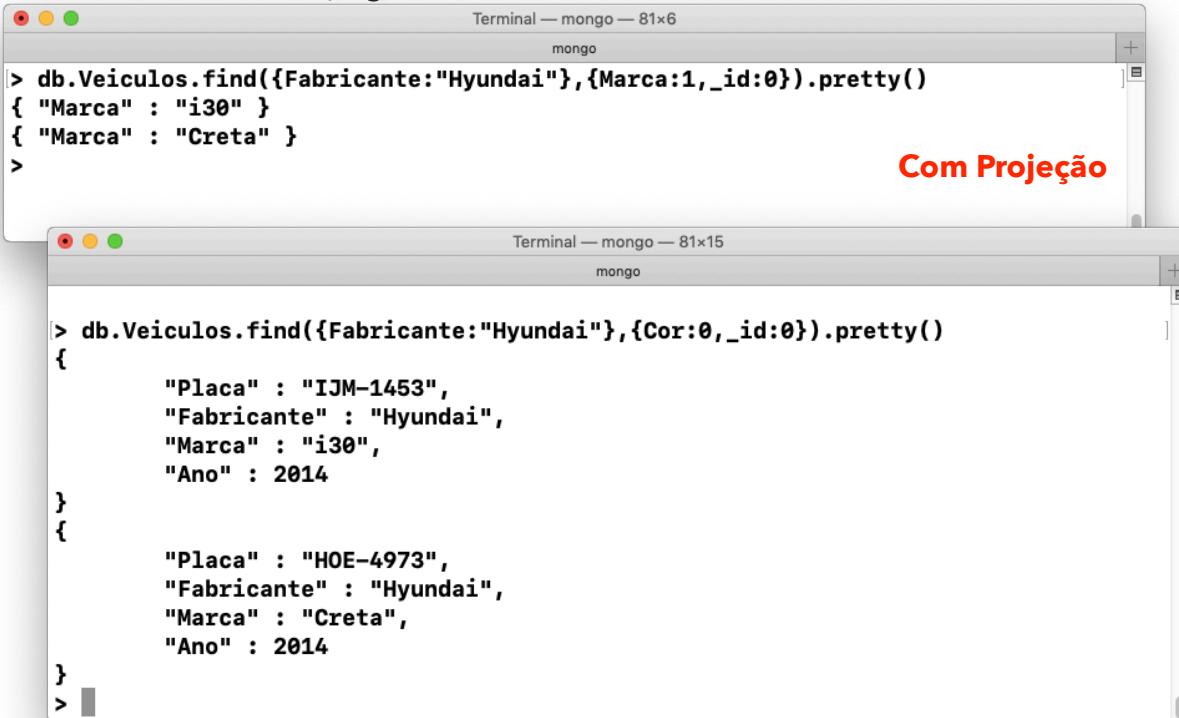
Sem Projeção

```
> db.Veiculos.find({Fabricante:"Hyundai"},{Marca:1}).pretty() Com Projeção
{
  "_id" : ObjectId("5c7947b5262dabecd3a285ca"), "Marca" : "i30" }
{
  "_id" : ObjectId("5c7c1096262dabecd3a285cb"), "Marca" : "Creta" }
```

```
> db.Veiculos.find({}, {Marca:1}).pretty()
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Marca" : "Sandeiro" }
{
  "_id" : ObjectId("5c7947b5262dabecd3a285c9"), "Marca" : "Fox" }
{
  "_id" : ObjectId("5c7947b5262dabecd3a285ca"), "Marca" : "i30" }
{
  "_id" : ObjectId("5c7c1096262dabecd3a285cb"), "Marca" : "Creta" }
```

```
> db.Veiculos.find({}, {Marca:1, _id:0})
{
  "Marca" : "Sandeiro" }
{
  "Marca" : "Fox" }
{
  "Marca" : "i30" }
{
  "Marca" : "Creta" }
```

Consultas de Projeção



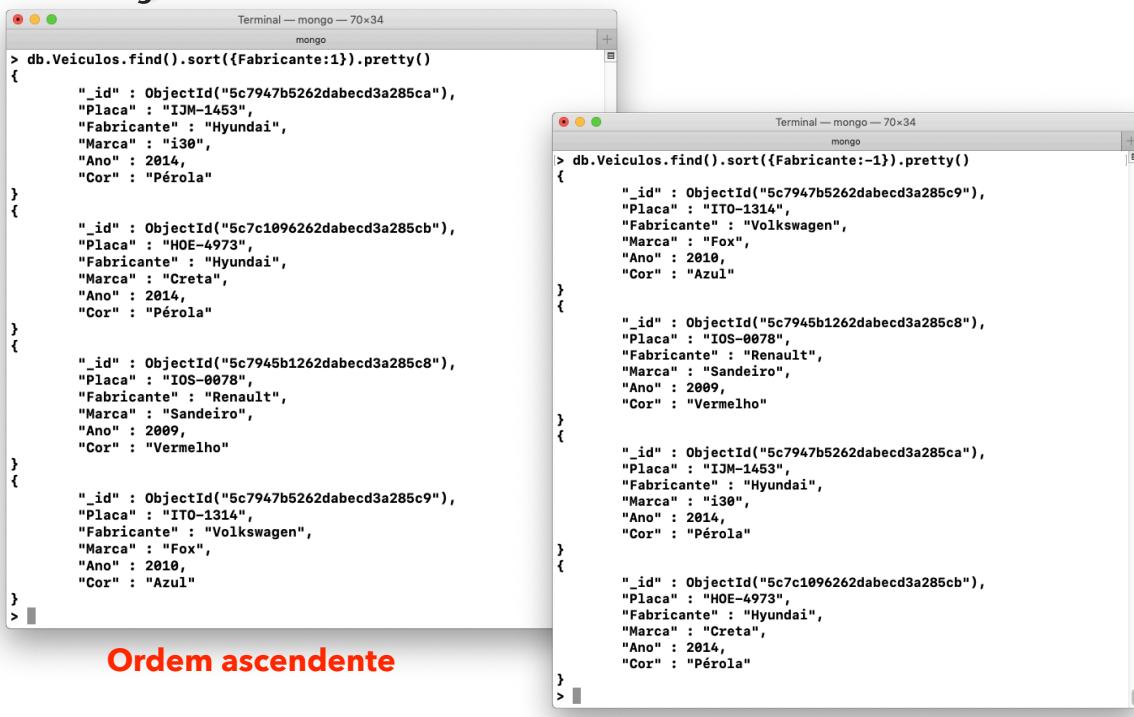
```
Terminal — mongo — 81x6
mongo
> db.Veiculos.find({Fabricante:"Hyundai"},{Marca:1,_id:0}).pretty()
{
  "Marca" : "i30"
}
{
  "Marca" : "Creta"
}

Com Projeção
```



```
Terminal — mongo — 81x15
mongo
> db.Veiculos.find({Fabricante:"Hyundai"},{Cor:0,_id:0}).pretty()
{
  "Placa" : "IJM-1453",
  "Fabricante" : "Hyundai",
  "Marca" : "i30",
  "Ano" : 2014
}
{
  "Placa" : "HOE-4973",
  "Fabricante" : "Hyundai",
  "Marca" : "Creta",
  "Ano" : 2014
}
```

Ordenação do Resultado de uma Consulta



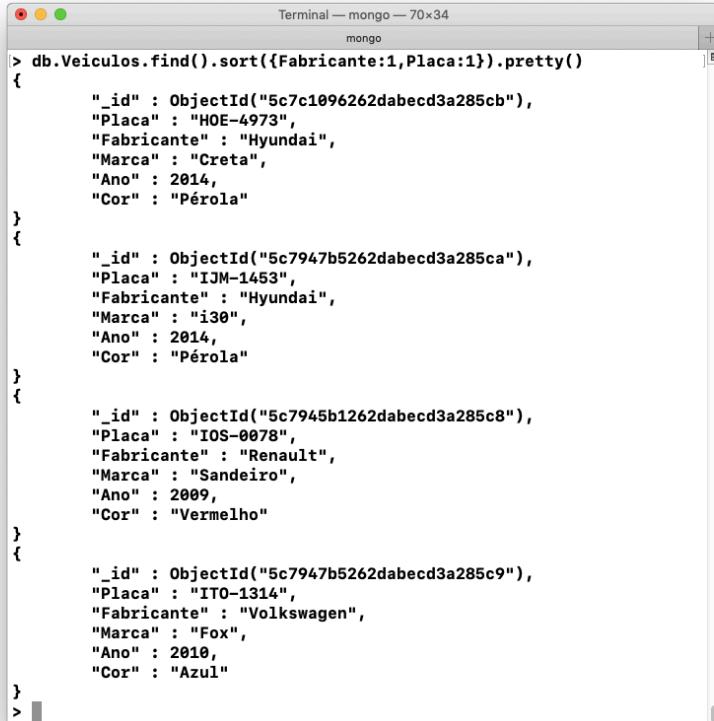
```
Terminal — mongo — 70x34
mongo
> db.Veiculos.find().sort({Fabricante:1}).pretty()
{
  "_id" : ObjectId("5c7947b5262dabecd3a285ca"),
  "Placa" : "IJM-1453",
  "Fabricante" : "Hyundai",
  "Marca" : "i30",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
{
  "_id" : ObjectId("5c7c1096262dabecd3a285cb"),
  "Placa" : "HOE-4973",
  "Fabricante" : "Hyundai",
  "Marca" : "Creta",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
  "Placa" : "IOS-0078",
  "Fabricante" : "Renault",
  "Marca" : "Sandeiro",
  "Ano" : 2009,
  "Cor" : "Vermelho"
}
{
  "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
  "Placa" : "ITO-1314",
  "Fabricante" : "Volkswagen",
  "Marca" : "Fox",
  "Ano" : 2010,
  "Cor" : "Azul"
}
>
```

Ordem ascendente

```
Terminal — mongo — 70x34
mongo
> db.Veiculos.find().sort({Fabricante:-1}).pretty()
{
  "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
  "Placa" : "IJM-1453",
  "Fabricante" : "Hyundai",
  "Marca" : "i30",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
{
  "_id" : ObjectId("5c7c1096262dabecd3a285cb"),
  "Placa" : "HOE-4973",
  "Fabricante" : "Hyundai",
  "Marca" : "Creta",
  "Ano" : 2014,
  "Cor" : "Pérola"
}
{
  "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
  "Placa" : "IOS-0078",
  "Fabricante" : "Renault",
  "Marca" : "Sandeiro",
  "Ano" : 2009,
  "Cor" : "Vermelho"
}
>
```

Ordem descendente

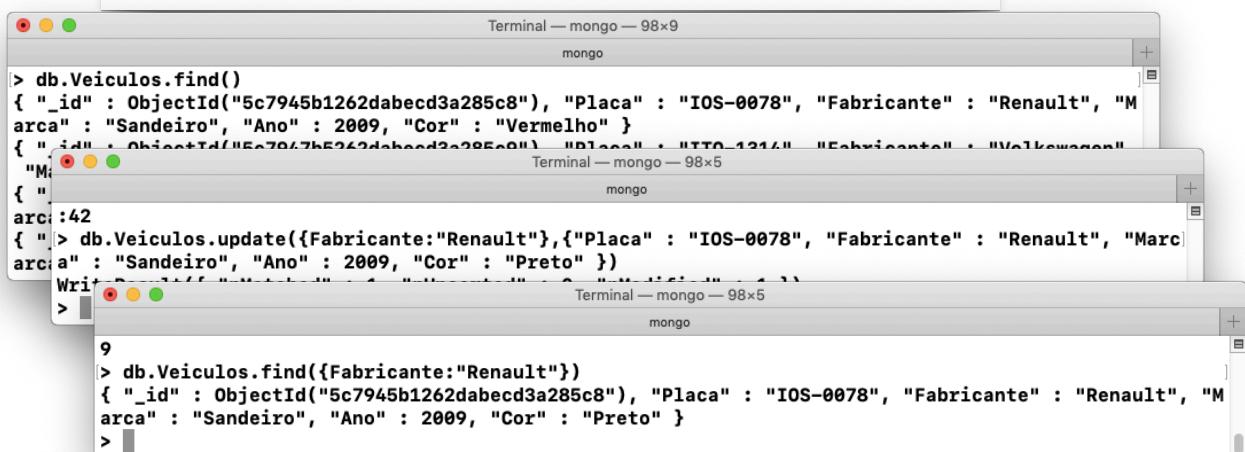
Ordenação do Resultado de uma Consulta: múltiplos campos



```
Terminal — mongo — 70x34
mongo
> db.Veiculos.find().sort({Fabricante:1,Placa:1}).pretty()
{
    "_id" : ObjectId("5c7c109626dabecd3a285cb"),
    "Placa" : "H0E-4973",
    "Fabricante" : "Hyundai",
    "Marca" : "Creta",
    "Ano" : 2014,
    "Cor" : "Pérola"
}
{
    "_id" : ObjectId("5c7947b5262dabecd3a285ca"),
    "Placa" : "IJM-1453",
    "Fabricante" : "Hyundai",
    "Marca" : "i30",
    "Ano" : 2014,
    "Cor" : "Pérola"
}
{
    "_id" : ObjectId("5c7945b1262dabecd3a285c8"),
    "Placa" : "IOS-0078",
    "Fabricante" : "Renault",
    "Marca" : "Sandeiro",
    "Ano" : 2009,
    "Cor" : "Vermelho"
}
{
    "_id" : ObjectId("5c7947b5262dabecd3a285c9"),
    "Placa" : "ITO-1314",
    "Fabricante" : "Volkswagen",
    "Marca" : "Fox",
    "Ano" : 2010,
    "Cor" : "Azul"
}
>
```

Atualização de documentos

- `db.collection.updateOne(<filter>, <update>, <options>)`
- `db.collection.updateMany(<filter>, <update>, <options>)`
- `db.collection.replaceOne(<filter>, <update>, <options>)`

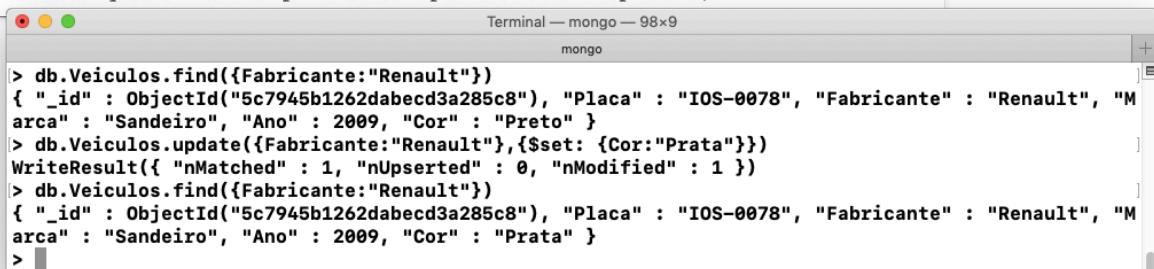


```
Terminal — mongo — 98x9
mongo
> db.Veiculos.find()
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault", "Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Vermelho" }
{ "_id" : ObjectId("5c7947b5262dabecd3a285ca"), "Placa" : "ITO-1314", "Fabricante" : "Volkswagen", "Marca" : "Fox", "Ano" : 2010, "Cor" : "Azul" }
> db.Veiculos.updateOne({Fabricante:"Renault"}, {"$set": {"Placa": "IOS-0078", "Fabricante": "Renault", "Marca": "Sandeiro", "Ano": 2009, "Cor": "Preto"}}, {"upsert": false})
WriteResult { "nModified" : 1, "nMatched" : 1, "nAffected" : 1 }
>
9
> db.Veiculos.find({Fabricante:"Renault"})
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault", "Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Preto" }
>
```

Atualização de documentos: operadores de update

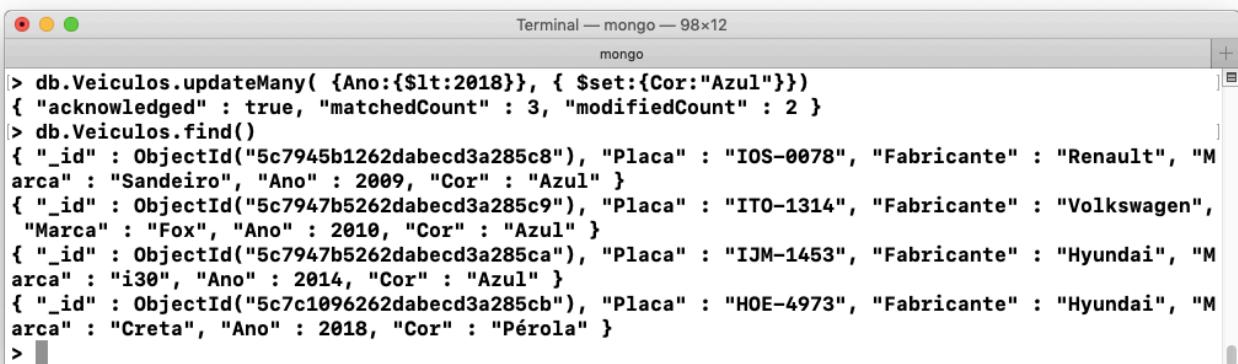
- `db.collection.updateOne(<filter>, <update>, <options>)`
- `db.collection.updateMany(<filter>, <update>, <options>)`
- `db.collection.replaceOne(<filter>, <update>, <options>)`

- `$set`: muda o valor de um campo;
- `$unset`: remove um campo do documento;
- `$rename`: muda o nome de um campo do documento;
- `$inc`: quando um campo numérico precisa ser incrementado (ou decrementado, usando um valor negativo);
- `$mul`: quando um campo numérico precisa ser multiplicado;



```
Terminal — mongo — 98x9
mongo
> db.Veiculos.find({Fabricante:"Renault"})
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault", "Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Preto" }
> db.Veiculos.update({Fabricante:"Renault"},{$set: {Cor:"Prata"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Veiculos.find({Fabricante:"Renault"})
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault", "Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Prata" }
> 
```

Atualização de documentos: operadores de update



```
Terminal — mongo — 98x12
mongo
> db.Veiculos.updateMany( {Ano:{$lt:2018}}, { $set:{Cor:"Azul"}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
> db.Veiculos.find()
{ "_id" : ObjectId("5c7945b1262dabecd3a285c8"), "Placa" : "IOS-0078", "Fabricante" : "Renault", "Marca" : "Sandeiro", "Ano" : 2009, "Cor" : "Azul" }
{ "_id" : ObjectId("5c7947b5262dabecd3a285c9"), "Placa" : "ITO-1314", "Fabricante" : "Volkswagen", "Marca" : "Fox", "Ano" : 2010, "Cor" : "Azul" }
{ "_id" : ObjectId("5c7947b5262dabecd3a285ca"), "Placa" : "IJM-1453", "Fabricante" : "Hyundai", "Marca" : "i30", "Ano" : 2014, "Cor" : "Azul" }
{ "_id" : ObjectId("5c7c1096262dabecd3a285cb"), "Placa" : "HOE-4973", "Fabricante" : "Hyundai", "Marca" : "Creta", "Ano" : 2018, "Cor" : "Pérola" }
> 
```

Exclusão de documentos

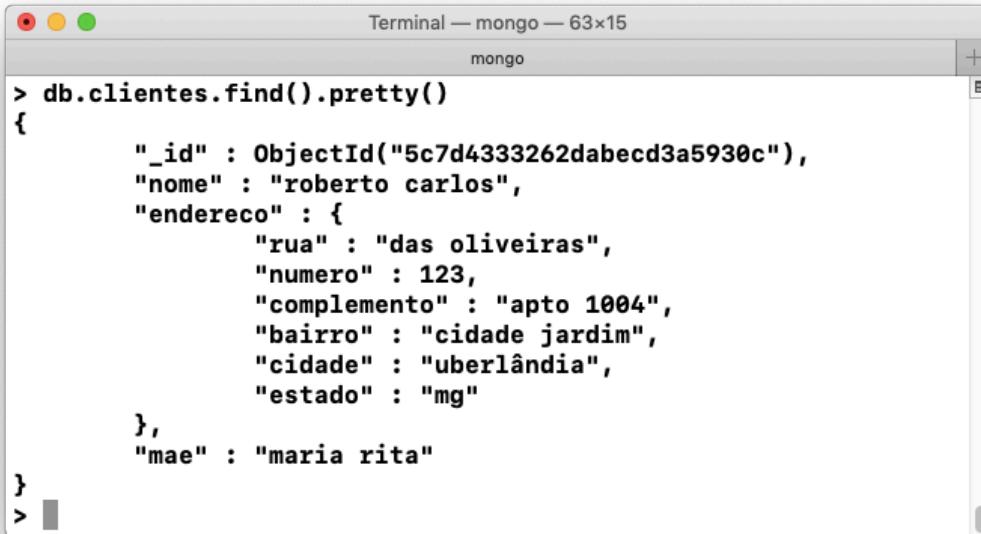
- `db.collection.deleteMany()`
- `db.collection.deleteOne()`

- Para excluir todos os documentos de uma coleção:
db.Veiculos.deleteMany({}) ou db.Veiculos.deleteMany()
- Para excluir todos os documentos segundo um filtro:
db.Veiculos.deleteMany({Ano:2010})
- Para excluir somente um documento segundo um filtro:
db.Veiculos.deleteOne({Ano:2010}) : exclui o primeiro documento que equivale ao filtro.

Relacionamentos

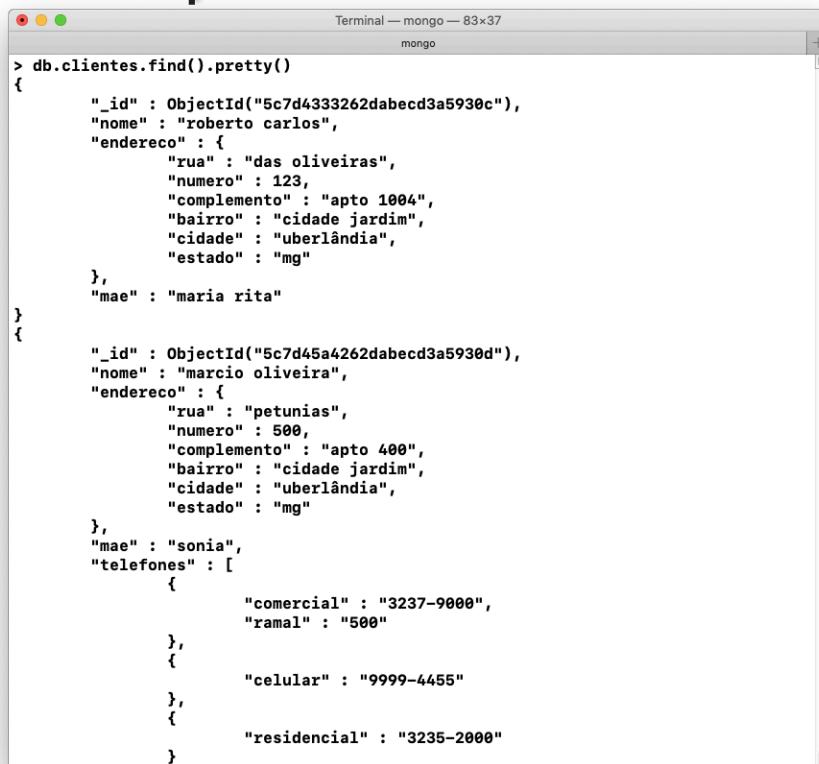
- Para criar uma relacionamento na modelagem, ou um documento é embutido em outro, ou então, um referencia o outro:
 - ▶ **documentos embutidos**
 - ▶ **documentos referenciados**
- O método a ser utilizado depende de como a consulta será realizada.
- Com MongoDB, você pode embutir documentos em outros documentos. Sendo assim, um único documento pode conter seus próprios relacionamentos.

Relacionamento Um para Um (1..1)



```
Terminal — mongo — 63x15
mongo
> db.clientes.find().pretty()
{
    "_id" : ObjectId("5c7d4333262dabecd3a5930c"),
    "nome" : "roberto carlos",
    "endereco" : {
        "rua" : "das oliveiras",
        "numero" : 123,
        "complemento" : "apto 1004",
        "bairro" : "cidade jardim",
        "cidade" : "uberlândia",
        "estado" : "mg"
    },
    "mae" : "maria rita"
}
>
```

Relacionamento Um para Muitos (1..*)



```
Terminal — mongo — 83x17
mongo
> db.clientes.find().pretty()
{
    "_id" : ObjectId("5c7d4333262dabecd3a5930c"),
    "nome" : "roberto carlos",
    "endereco" : {
        "zua" : "das oliveiras",
        "numero" : 123,
        "complemento" : "apto 1004",
        "bairro" : "cidade jardim",
        "cidade" : "uberlândia",
        "estado" : "mg"
    },
    "mae" : "maria rita"
}
{
    "_id" : ObjectId("5c7d45a4262dabecd3a5930d"),
    "nome" : "marcio oliveira",
    "endereco" : {
        "rua" : "petunias",
        "numero" : 500,
        "complemento" : "apto 400",
        "bairro" : "cidade jardim",
        "cidade" : "uberlândia",
        "estado" : "mg"
    },
    "mae" : "sonia",
    "telefones" : [
        {
            "comercial" : "3237-9000",
            "ramal" : "500"
        },
        {
            "celular" : "9999-4455"
        },
        {
            "residencial" : "3235-2000"
        }
    ]
}
```