

Banco de Dados II

MongoDB

Comandos básicos

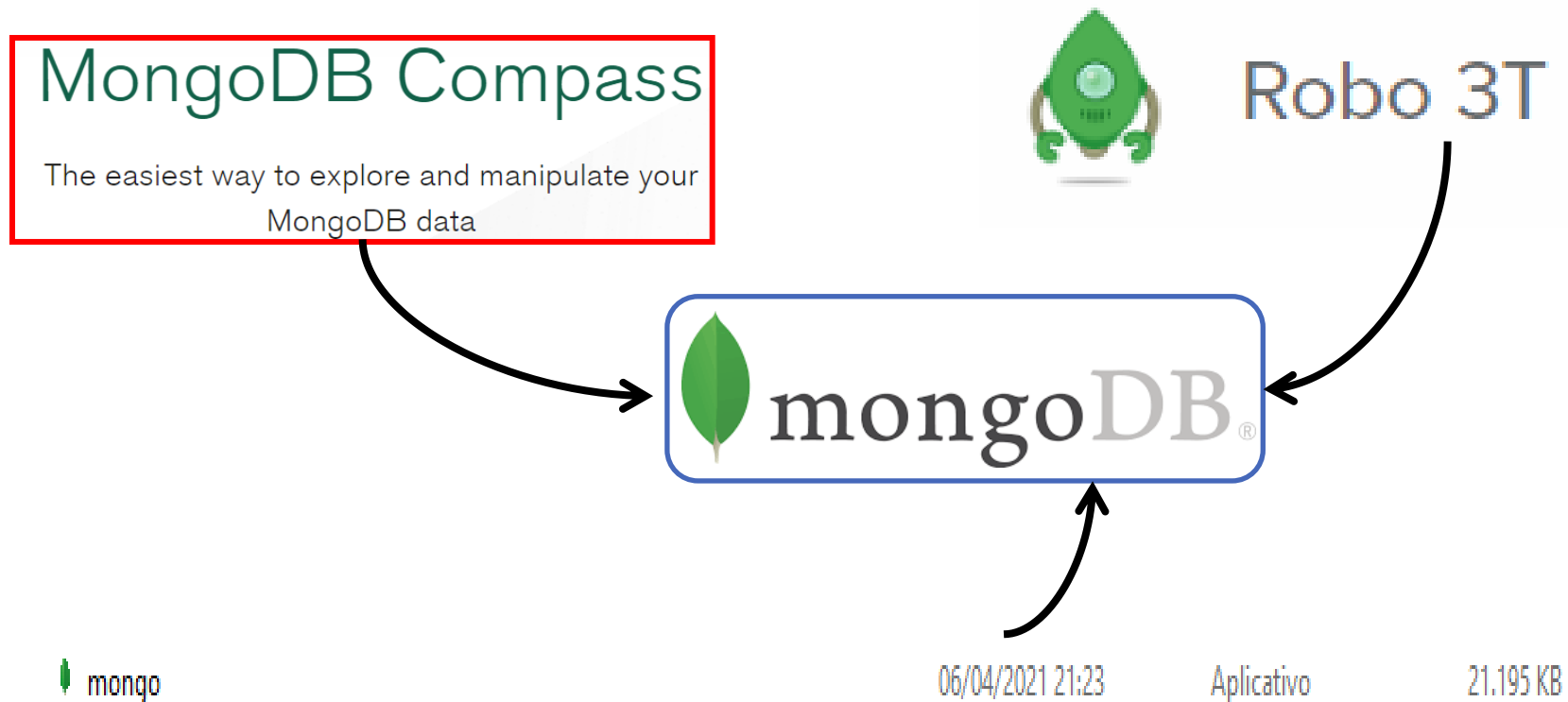


Profa. Damires Souza
damires@ifpb.edu.br



Acessando o MongoDB

- O acesso ao MongoDB pode ser feito via **console** (executável mongo) ou via **ambiente gráfico**



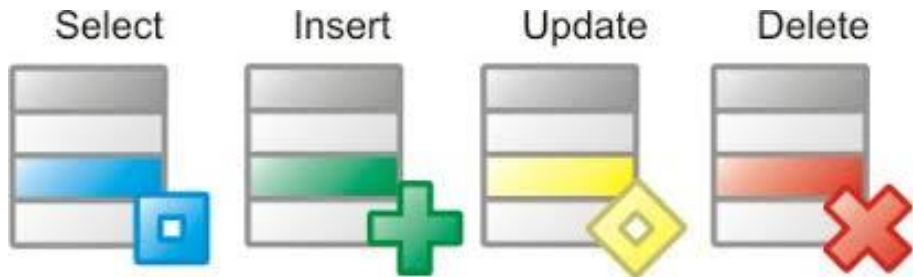
Interface Compass

The screenshot displays the MongoDB Compass web interface. The top bar shows the connection to 'localhost:27017/BD2.Cliente'. The left sidebar lists the database 'BD2' and its collections, with 'Cliente' selected. The main panel shows the 'BD2.Cliente' collection with 5 documents and 1 index. The 'Documents' tab is active, displaying a list of documents. A terminal window is overlaid at the bottom, showing the following commands and output:

```
>_MONGOSH
test>
> use local
< switched to db local
local>
```

O que vamos fazer?

=> Manipular Dados



Create/Insert

- Operações de “criar” ou inserir **adicionam** novos documentos a uma coleção.

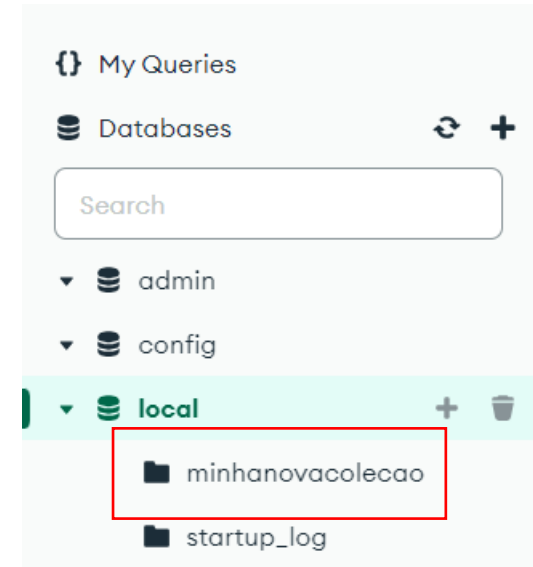
◦ Se a coleção não existir, as operações de inserção criarão a coleção.

- Opções:
 - **db.collection.insertOne ()**
 - **db.collection.insertMany ()**

Create/Insert

- **db.collection.insertOne()**

```
db.minhanovacolecao.insertOne  
({nome: "João"})
```



```
> db.minhanovacolecao.insertOne({nome: "João"})  
< {  
  acknowledged: true,  
  insertedId: ObjectId("6567210e0e7523a43d49bf02")  
}  
local>
```

Create/Insert

- Mais um documento

```
db.minhanovacolecao.insertOne(  
  { nome: “Marcia”, idade: 23, hobbies: [“dança“,  
    “filmes”], endereço: { rua: “JJ”, num: 35, apto: 202 }  
  } )
```

** verificar:

```
db.minhanovacolecao.find( { nome: “Marcia” } )
```

Create/Insert

- **db.collection.insertMany ()**

- pode inserir vários documentos em uma coleção
- passa um **array** de documentos para o método

```
db.minhanovacolecao.insertMany( [  
  { nome: "Alana", idade: 33, hobbies: ["volley", "filmes"],  
    endereço: { rua: "XX", num: 305, apto: 502 } },  
  { nome: "Alvaro", idade: 43, hobbies: ["volley", "squash"],  
    endereço: { rua: "YY", num: 43 } }  
] )
```

** verificar: [db.minhanovacolecao.find\(\)](#)

Insert a partir de variável

```
document = (  
  { nome: "Anísio", idade: 36, hobbies: ["futebol", "rock"],  
    endereço: { rua: "MM", num: 305, apto: 502 } } )
```

```
db.minhanovacolecao.insertOne(document)
```

```
>_MONGOSH  
  
}  
> document = ( { nome: "Anísio", idade: 36, hobbies: ["futebol", "rock"], endereço: { rua: "MM", num: 305, apto: 502 } } )  
< {  
  nome: 'Anísio',  
  idade: 36,  
  hobbies: [ 'futebol', 'rock' ],  
  'endereço': { rua: 'MM', num: 305, apto: 502 }  
}  
> db.minhanovacolecao.insertOne(document)  
< {  
  acknowledged: true,  
  insertedId: ObjectId("6567222b0e7523a43d49bf06")  
}
```

Criação de coleção

- As operações `insert ()` e `createIndex ()` criam sua respectiva coleção **se elas ainda não existirem**

`db.NovaColecaoTeste1.insertOne ({x: 1})`

`db.NovaColecaoTeste2.createIndex ({y: 1})`

Coleções

MongoDB Compass - localhost:27017/local

Connect Edit View Help

localhost:27017 ...

My Queries

Databases

Search

admin

config

local

NovaColecaoTeste1

NovaColecaoTeste2

minhanovacolecao

startup_log

Collections

+ Create collection

Refresh

View

☰ ☳

Sort by

Collection Name

minhanovacolecao

Storage size:
20.48 kB

Documents:
5

Avg. document size:
116.00 B

Indexes:
1

Total index size:
36.86 kB

NovaColecaoTeste1

Storage size:
20.48 kB

Documents:
1

Avg. document size:
29.00 B

Indexes:
1

Total index size:
20.48 kB

NovaColecaoTeste2

Storage size:
4.10 kB

Documents:
0

Avg. document size:
0 B

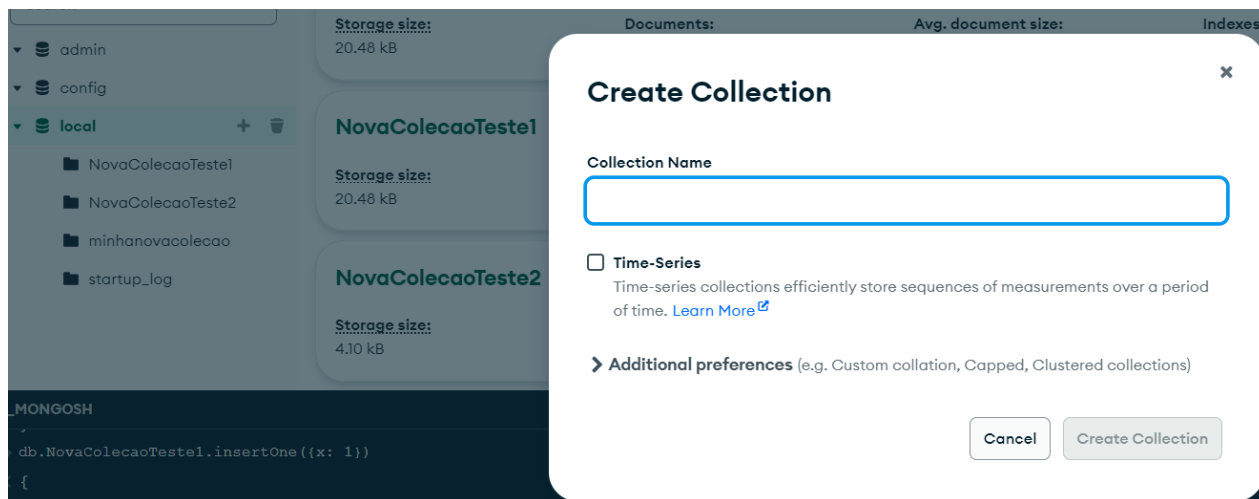
Indexes:
2

Total index size:
8.19 kB

Criação de coleção

- Nomes de coleção devem começar com um sublinhado ou um **caractere de letra** e não podem:
 - Conter o valor \$; ser uma string vazia (por exemplo, ""); começar com o system (reservado)
- Para criar uma coleção explicitamente:

db.createCollection(name,[options])



Ou seja...

- Um dado MongoDB é salvo a partir de um **objeto do tipo JSON/BSON**
- Há a abstração de diversos comandos SQL/DDL
 - Exemplo: alter table...
- Estruturas são criadas conforme se tornem necessárias.

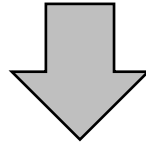


Formato geral dos comandos

- A maioria dos comandos segue a seguinte sintaxe:

```
db.<nome-da-collection>.<operacao-desejada>;
```

db.minhanovacolecao.countDocuments({})



```
Select count(*)  
From minhanovacolecao;
```

Comando Find()

```
db.collection.find(query, projection)
```

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
) .limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

➤ Retorna um **cursor** com os documentos selecionados

- Filtro de seleção é definido usando operadores
- Para retornar todos os documentos em uma coleção, omitir esse parâmetro ou passar um documento vazio ({})
- **Projeção = opcional**
 - Especifica os campos a serem retornados nos documentos que correspondem ao filtro de consulta.
 - Para retornar todos os campos nos documentos correspondentes, omitir este parâmetro.

Comando Find()

```
db.minhanovacolecao.find( {nome : "Marcia"})
```

```
< {
  _id: ObjectId("656721b10e7523a43d49bf03"),
  nome: 'Marcia',
  idade: 23,
  hobbies: [
    'dança',
    'filmes'
  ],
  'endereço': {
    rua: 'JJ',
    num: 35,
    apto: 202
  }
}
```


Find() com seleção

Condição
de
igualdade

- `db.minhanovacolecao.find({ nome: "João" })`

=> **Select** * from minhanovacoleção

Where nome = "João"

- `db.minhanovacolecao.find({ nome: { $in: ["João", "Marcia"] } })`

- `db.minhanovacolecao.find({ nome: /^A/ })`

=> **Select** * From minhanovacoleção

Where nome LIKE "A%"

****** <https://docs.mongodb.com/manual/reference/operator/query/regex/>

Alguns operadores de seleção

Operação	Syntaxe	Exemplo	RDBMS Equivalente
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{\$lt:<value>}}	db.mycol.find({"likes":{\$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{\$lte:<value>}}	db.mycol.find({"likes":{\$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{\$gt:<value>}}	db.mycol.find({"likes":{\$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{\$gte:<value>}}	db.mycol.find({"likes":{\$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{\$ne:<value>}}	db.mycol.find({"likes":{\$ne:50}}).pretty()	where likes != 50

Operador OR e IN

```
db.minhanovacolecao.find({  
  $or :  
    [ {"nome" : "Alvara"},  
      {"nome" : "Alvaro"} ] })
```

```
db.minhanovacolecao.find(  
  { "nome" :  
    { $in : ["Alvara", "Alvaro"] } })
```

Find() com Projeção

- Por padrão, consultas retornam **todos os campos** em documentos correspondentes
- Pode-se especificar ou restringir os campos a serem retornados usando a **projeção**

<campo>: 1

```
db.minhanovacolecao.find( { nome: { $in: [ "João",  
"Marcia" ] } }, { nome: 1, idade: 1 } )
```

```
db.minhanovacolecao.find( { nome: { $in: [ "João",  
"Marcia" ] } }, { nome: 1, idade: 1, _id: 0 } )
```

```
⇒ Select nome, idade From minhanovacoleção  
Where nome in ('João', 'Marcia');
```

Find() com acesso a documentos incorporados ou embutidos

- Para especificar ou acessar um campo de um documento incorporado:

"<Embedded document>. <Field>"

- Exemplos:

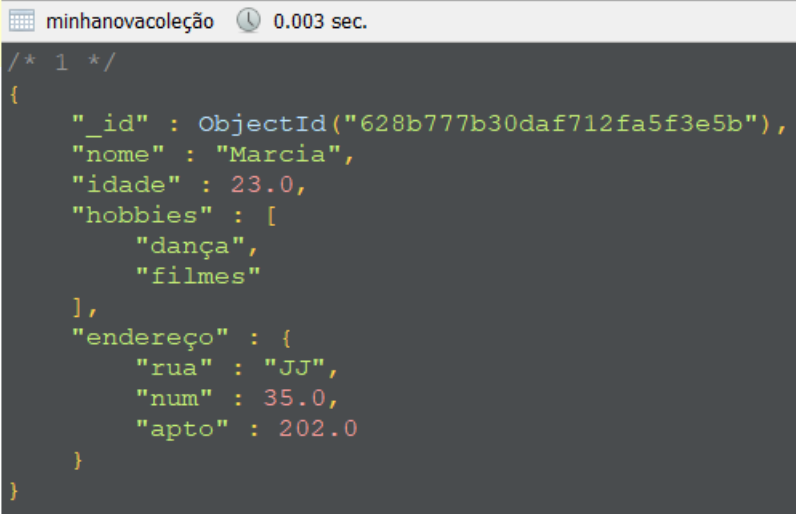
"name.last"

"contact.phone.number"

Find() com acesso a documentos incorporados ou embutidos

```
db.minhanovacolecao.find( {  
  endereço: { rua: "JJ", num: 35, apto: 202 } } )
```

```
db.minhanovacolecao.find( {  
  "endereço.rua": "JJ" } )
```



The screenshot shows a MongoDB console window with the title "minhanovacoleção" and a timer showing "0.003 sec.". The console displays a single document found by the query. The document is a JSON object with the following fields: "_id" (ObjectId), "nome" (Marcia), "idade" (23.0), "hobbies" (array containing "dança" and "filmes"), and "endereço" (object containing "rua" (JJ), "num" (35.0), and "apto" (202.0)).

```
/* 1 */  
{  
  "_id" : ObjectId("628b777b30daf712fa5f3e5b"),  
  "nome" : "Marcia",  
  "idade" : 23.0,  
  "hobbies" : [  
    "dança",  
    "filmes"  
  ],  
  "endereço" : {  
    "rua" : "JJ",  
    "num" : 35.0,  
    "apto" : 202.0  
  }  
}
```

Acesso a Array




- Para especificar ou acessar um elemento de um array pela posição de índice:

"<Array>. <Index>"

- Exemplo:

"hobbies.1"

=> **segundo elemento** no array hobbies

▼  hobbies
  [0]
  [1]

[2 elements]
volley
squash

Find() com acesso a array

- `db.minhanovacolecao.find({ "hobbies.1": "filmes" })`
- `db.minhanovacolecao.find({ hobbies: ["volley", "filmes"] })`
 - Considera a ordem dos elementos
- `db.minhanovacolecao.find({ hobbies: { $all: ["filmes", "volley"] } })`
 - Sem considerar a ordem

Key	Value
▼ (1) ObjectId("58d3c79f87fb0777be62bcef")	{ 5 fields }
_id	ObjectId("58d3c79f87fb0777be62bcef")
nome	Alana
idade	33.0
▼ hobbies	[2 elements]
[0]	volley
[1]	filmes
> endereço	{ 3 fields }

O que os comandos retornam?

- `db.minhanovacolecao.find({ nome: "Alvaro" }, { idade: 0, _id: 0 })`
- `db.minhanovacolecao.find({ nome: "Alvaro" }, { idade: 1, "endereço.rua": 1 })`

Verifique!



Update

- As operações de atualização modificam os documentos existentes em uma coleção.
 - `db.collection.updateOne()`
 - `db.collection.updateMany()`

```
db.users.update(  
  { age: { $gt: 18 } },  
  { $set: { status: "A" } },  
  { multi: true }  
)
```

← collection
← update criteria
← update action
← update option

Update

- `db.minhanovacolecao.updateOne(
 { "nome" : "Alvaro" },
 { $set: { "idade" : 38 } })`
- `db.minhanovacolecao.updateMany(
 { idade: { $gt: 40 } },
 { $set: { "Bônus" : true } })`

Update

- `db.minhanovacoletão.updateOne(
 { "nome" : "Alvara" },
 { $set: {"idade" : 57,
 hobbies: ["judo", "filmes"],
 endereço: { rua: "KK", num: 305, apto: 202 } } },
 { upsert: true })`

=> **Upsert** -> Busca um objeto; se ele existir, atualiza; senão, insere

Delete

- Operação que remove documentos de uma coleção
 - `db.collection.deleteOne()`
 - `db.collection.deleteMany()`

```
db.minhanovacoção.deleteOne(  
  { Bônus: true } )
```