

**INSTITUTO INFNET**

FÁBIO LUÍS G. G. DE OLIVEIRA

**PROJETO DA DISCIPLINA**

Infraestrutura MongoDB

CURITIBA - PR

2022

## PARTE 1 – CONCEITOS

1. Descreva com suas palavras 3 principais diferenças e 3 semelhanças entre bases de dados SQL e bases noSQL.

Principais Diferenças:

- Bases de dados SQL são usualmente utilizadas para dados estruturados (mesma estrutura e esquema bem definido) enquanto que as bases noSQL são usadas para dados não estruturados e semi-estruturados (não há uma estrutura única e há a adição de novas propriedades, se necessário);
- Bases de dados SQL são focadas em consistência e disponibilidade enquanto que as bases de dados noSQL podem ter foco em consistência e tolerância a partição de rede ou disponibilidade e tolerância a partição de rede;
- As bases de dados relacionais não são indicadas e aplicadas em infraestruturas big data, ou seja, clusters de computadores que trabalham de forma paralela e distribuída com o uso da escalabilidade horizontal. Já as bases de dados não relacionais possuem escalabilidade horizontal pois trabalham de forma distribuída e paralela. São ideais para ambientes big data.

Principais Semelhanças:

- Ambas permitem operações de criação, leitura, update e delete dos dados;
- As bases de dados noSQL tentam, na maioria das vezes, resgatar alguns conceitos que estão bem consolidados pelas bases de dados SQL, como: database, tabelas, registro, join; aplicando esses conceitos na infraestrutura bigdata;
- Buscam solucionar bases de dados de problemas de negócios distintos.

2. Em sua opinião as bases noSQL são melhores ou piores que as bases SQL? Em que elas se destacam? Justifique sua resposta.

As bases de dados noSQL geralmente são usadas em ambientes onde há um grande volume, variedade e velocidade de dados. Ou seja, em ambientes de big data, onde é necessário um grande armazenamento e processamento de um grande volume de dados.

Isto não significa que bases noSQL sejam melhores que SQL, elas só possuem um foco diferente das bases relacionais. Em sistemas onde o modelo relacional é a melhor solução, o SQL pode (e deve) ser utilizado.

3. Descreva com suas palavras um cenário onde você optaria por um modelo noSQL e um cenário onde você optaria por um modelo SQL.

Modelo noSQL para cenários big data onde é necessário grande armazenamento de dados, uma alta performance e a possibilidade de escalabilidade horizontal em um modelo de processamento paralelo e distribuído.

Modelo SQL para cenários onde um modelo relacional é o suficiente, onde as base de dados são guardadas em tabelas que podem ser alteradas, apagadas, consultadas, etc. A escalabilidade vertical é o mais adequado para a aplicação deste modelo.

4. Além do material usado em aula, faça uma pequena pesquisa e descreva com suas palavras quais são os tipos de bases noSQL existentes no mercado atual, com as principais características de uso cada um tipo. Cite as fontes da pesquisa.

No geral, há 4 tipos de bancos de dados NoSQL:

- Orientado a Documento – Os dados são armazenados em coleções como documentos. Os documentos podem ser descritos como um conjunto de dados no formato de chave-valor, como por exemplo, o padrão JSON. Um exemplo de banco de dados neste formato é o MongoDB;
- Orientado a Colunas – As chaves apontam para atributos ou colunas múltiplas. Também permitem sub-colunas. Um banco de dados dessa família, por exemplo, é o Cassandra;

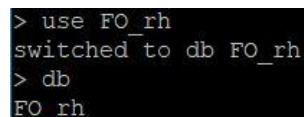
- Orientado a Grafos – Os dados são armazenados na forma de grafos (vértices e arestas). O Neo4j é um famoso exemplo desta família;
- Chave-valor – Consiste de uma chave e um dado (valor). Um banco de dados dessa família, por exemplo, é o Dynamo.

## PARTE 2 – PRÁTICA

### A. Instalações e criações:

1. Comando para criação do database "FO\_rh".

```
use FO_rh
db
```

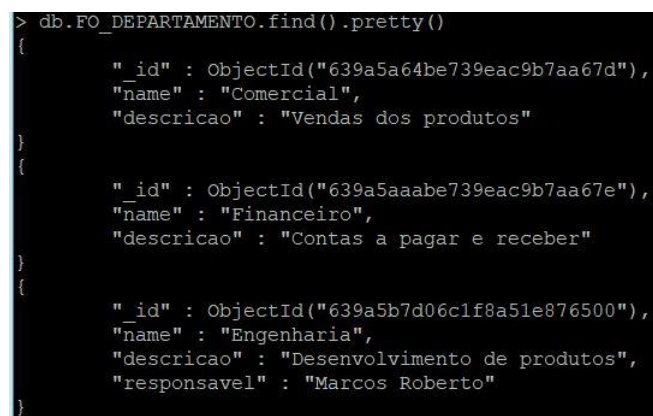


```
> use FO_rh
switched to db FO_rh
> db
FO_rh
```

Figura 1 – Criação do database FO\_rh

2. Comandos para criação das coleções "FO\_DEPARTAMENTO", "FO\_FUNCIONARIO", "FO\_DEPENDENTE".
3. Inserção de 3 documentos para cada coleção.

```
db.FO_DEPARTAMENTO.save({name: "Comercial", descricao: "Vendas dos produtos"})
db.FO_DEPARTAMENTO.save({name: "Financeiro", descricao: "Contas a pagar e receber"})
db.FO_DEPARTAMENTO.save({name: "Engenharia", descricao: "Desenvolvimento de produtos", responsavel: "Marcos Roberto"})
```



```
> db.FO_DEPARTAMENTO.find().pretty()
{
  "_id" : ObjectId("639a5a64be739eac9b7aa67d"),
  "name" : "Comercial",
  "descricao" : "Vendas dos produtos"
}
{
  "_id" : ObjectId("639a5aaabe739eac9b7aa67e"),
  "name" : "Financeiro",
  "descricao" : "Contas a pagar e receber"
}
{
  "_id" : ObjectId("639a5b7d06c1f8a51e876500"),
  "name" : "Engenharia",
  "descricao" : "Desenvolvimento de produtos",
  "responsavel" : "Marcos Roberto"
}
```

Figura 2 – Coleção FO\_DEPARTAMENTO com 3 documentos

```

db.FO_FUNCIONARIO.save({name: "Maria", cargo:"Assessor de Vendas"})
db.FO_FUNCIONARIO.save({name: "Eliseu", cargo:"Desenvolvedor Backend", salario: 5000.00})
db.FO_FUNCIONARIO.save({name: "Roberto", cargo:"Diretor de Engenharia", salario: 25000.00})

```

```

> db.FO_FUNCIONARIO.find().pretty()
{
  "_id" : ObjectId("639a5dc706c1f8a51e876501"),
  "name" : "Maria",
  "cargo" : "Assessor de Vendas"
}
{
  "_id" : ObjectId("639a5e1c06c1f8a51e876502"),
  "name" : "Eliseu",
  "cargo" : "Desenvolvedor Backend",
  "salario" : 5000
}
{
  "_id" : ObjectId("639a5e5d06c1f8a51e876503"),
  "name" : "Roberto",
  "cargo" : "Diretor de Engenharia",
  "salario" : 25000
}

```

Figura 3 – Coleção FO\_FUNCIONARIO com 3 documentos

```

db.FO_DEPENDENTE.save({name: "Carlos", numero_dependente: 2})
db.FO_DEPENDENTE.save({name: "Carlos", numero_dependente: 5,
auxilio: "sim"})
db.FO_DEPENDENTE.save({name: "Denise", numero_dependente: 3,
auxilio: "sim"})

```

```

> db.FO_DEPENDENTE.find().pretty()
{
  "_id" : ObjectId("639a5fd306c1f8a51e876504"),
  "name" : "Carlos",
  "numero_dependente" : 2
}
{
  "_id" : ObjectId("639a601806c1f8a51e876505"),
  "name" : "Carlos",
  "numero_dependente" : 5,
  "auxilio" : "sim"
}
{
  "_id" : ObjectId("639a603506c1f8a51e876506"),
  "name" : "Denise",
  "numero_dependente" : 3,
  "auxilio" : "sim"
}

```

Figura 4 – Coleção FO\_DEPENDENTE com 3 documentos

#### 4. Mostrar o conteúdo de cada coleção.

```
db.FO_DEPARTAMENTO.find().toArray()
```

```
> db.FO_DEPARTAMENTO.find().toArray()
[
  {
    "_id" : ObjectId("639a5a64be739eac9b7aa67d"),
    "name" : "Comercial",
    "descricao" : "Vendas dos produtos"
  },
  {
    "_id" : ObjectId("639a5aaabe739eac9b7aa67e"),
    "name" : "Financeiro",
    "descricao" : "Contas a pagar e receber"
  },
  {
    "_id" : ObjectId("639a5b7d06clf8a51e876500"),
    "name" : "Engenharia",
    "descricao" : "Desenvolvimento de produtos",
    "responsavel" : "Marcos Roberto"
  }
]
```

Figura 5 – Coleção FO\_DEPARTAMENTO – Formato de array

```
db.FO_FUNCIONARIO.find().toArray()
```

```
> db.FO_FUNCIONARIO.find().toArray()
[
  {
    "_id" : ObjectId("639a5dc706clf8a51e876501"),
    "name" : "Maria",
    "cargo" : "Assessor de Vendas"
  },
  {
    "_id" : ObjectId("639a5e1c06clf8a51e876502"),
    "name" : "Eliseu",
    "cargo" : "Desenvolvedor Backend",
    "salario" : 5000
  },
  {
    "_id" : ObjectId("639a5e5d06clf8a51e876503"),
    "name" : "Roberto",
    "cargo" : "Diretor de Engenharia",
    "salario" : 25000
  }
]
```

Figura 6 – Coleção FO\_FUNCIONARIO – Formato de array

```
db.FO_DEPENDENTE.find().toArray()
```

```
> db.FO_DEPENDENTE.find().toArray()
[
  {
    "_id" : ObjectId("639a5fd306clf8a51e876504"),
    "name" : "Carlos",
    "numero_dependente" : 2
  },
  {
    "_id" : ObjectId("639a601806clf8a51e876505"),
    "name" : "Carlos",
    "numero_dependente" : 5,
    "auxilio" : "sim"
  },
  {
    "_id" : ObjectId("639a603506clf8a51e876506"),
    "name" : "Denise",
    "numero_dependente" : 3,
    "auxilio" : "sim"
  }
]
```

Figura 7 – Coleção FO\_DEPENDENTE – Formato de array

5. Comando na coleção "FO\_DEPARTAMENTO" com um filtro baseado na descrição do departamento.

```
db.FO_DEPARTAMENTO.find({descricao: "Desenvolvimento de produtos"},
{name: 1})
```

```
> db.FO_DEPARTAMENTO.find({descricao: "Desenvolvimento de produtos"}, {name: 1})
{ "_id" : ObjectId("639a5b7d06clf8a51e876500"), "name" : "Engenharia" }
>
```

Figura 8 – Coleção FO\_DEPARTAMENTO com filtro na descrição

6. Para a coleção "FO\_FUNCIONARIO" mostrar os funcionários que recebem salário acima de R \$2000,00.

```
db.FO_FUNCIONARIO.find({salario:{$gt:2000}})
```

```
> db.FO_FUNCIONARIO.find({salario:{$gt:2000}})
{ "_id" : ObjectId("639a5e1c06clf8a51e876502"), "name" : "Eliseu", "cargo" : "Desenvolvedor Backend", "salario" : 5000 }
{ "_id" : ObjectId("639a5e5d06clf8a51e876503"), "name" : "Roberto", "cargo" : "Diretor de Engenharia", "salario" : 25000 }
>
```

Figura 9 – Coleção FO\_FUNCIONARIO com salários maiores que 2000

7. Para a coleção "FO\_DEPENDENTE" executar o método distinct para o atributo nome.

```
db.FO_DEPENDENTE.distinct("name")
```

```
> db.FO_DEPENDENTE.distinct("name")
[ "Carlos", "Denise" ]
>
```

Figura 10 – Coleção FO\_DEPENDENTE com método distinct()

8. Executar um update para uma das coleções (alterando salário do Eliseu de 5000 para 7000 na coleção FO\_FUNCIONARIO)

```
db.FO_FUNCIONARIO.update({ name : "Eliseu" }, {$set: { salario :
7000 } })
```

```
> db.FO_FUNCIONARIO.update({ name : "Eliseu" }, {$set: { salario : 7000 } })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.FO_FUNCIONARIO.find()
{ "_id" : ObjectId("639a5dc706c1f8a51e876501"), "name" : "Maria", "cargo" : "Assessor de Vendas" }
{ "_id" : ObjectId("639a5e1c06c1f8a51e876502"), "name" : "Eliseu", "cargo" : "Desenvolvedor Backend", "salario" : 7000 }
{ "_id" : ObjectId("639a5e5d06c1f8a51e876503"), "name" : "Roberto", "cargo" : "Diretor de Engenharia", "salario" : 25000 }
>
```

Figura 11 – Update da Coleção FO\_FUNCIONARIO

9. Executar um delete (remove) para uma das coleções (remoção de Carlos com ajuda auxílio = "sim" da coleção FO\_DEPENDENTE)

```
db.FO_DEPENDENTE.remove({auxilio : "sim", name : "Carlos"})
```

```
> db.FO_DEPENDENTE.remove({auxilio : "sim", name : "Carlos"})
WriteResult({ "nRemoved" : 1 })
> db.FO_DEPENDENTE.find()
{ "_id" : ObjectId("639a5fd306c1f8a51e876504"), "name" : "Carlos", "numero_dependente" : 2 }
{ "_id" : ObjectId("639cb69ff3a991a3561c5ed4"), "name" : "Denise", "numero_dependente" : 3, "auxilio" : "sim" }
>
```

Figura 12 – Delete de documento da Coleção FO\_DEPENDENTE

## B. Modelagem

1. Criação do database "FO\_modelo".

```
> use FO_modelo
switched to db FO_modelo
> db
FO_modelo
>
```

Figura 13 – Criação do database FO\_modelo



2. Criar um pequeno modelo que envolva duas coleções com as cardinalidades 1-N. O nome das coleções deve ser "FO\_col1a" e "FO\_col2a". Use nesse caso o conceito de referenciar o id de uma coleção para a outra (envie para o lado N como se fosse a chave primária do lado 1).

a. Comandos para a inserção de 2 documentos na coleção (lado 1).

Duas coleções: uma de clientes que será a coleção FO\_col1a e carros que será FO\_col2a.

```
db.FO_colla.insert({nome : "Daniel", email : "daniel@gmail.com",
telefone : "99999-9999"})
db.FO_colla.insert({nome : "Juliana", email :
"juliana@gmail.com", telefone : "88888-8888"})
```

```
> db.FO_colla.insert({nome : "Daniel", email : "daniel@gmail.com", telefone : "99999-9999"})
WriteResult({ "nInserted" : 1 })
> db.FO_colla.insert({nome : "Juliana", email : "juliana@gmail.com", telefone : "88888-8888"})
WriteResult({ "nInserted" : 1 })
>
```

Figura 14 – Lado 1 da cardinalidade 1-N

b. Comandos para a inserção de 4 documentos na coleção (lado N).

Obs.: O lado 1 (é como se fosse a coleção mãe) e o lado N (é como se fosse a coleção filha). No item (2.2) cada documento da coleção mãe deverá referenciar 2 documentos da coleção filha.

```
db.FO_col2a.insert({nome: "brasilgia", ano : 1982, chassi :
"D12EFG45", cliente_id : db.FO_colla.find()[0]._id})
db.FO_col2a.insert({nome: "passat", ano : 1988, chassi :
"HIJ987KL25", cliente_id : db.FO_colla.find()[0]._id})
db.FO_col2a.insert({nome: "hilux", ano : 2021, chassi :
"AAAA100000", cliente_id : db.FO_colla.find()[1]._id})
db.FO_col2a.insert({nome: "mclaren", ano : 2022, chassi :
"MAC000001", cliente_id : db.FO_colla.find()[1]._id})
```

```
> db.FO_col2a.insert({nome: "brasilgia", ano : 1982, chassi : "D12EFG45", cliente_id : db.FO_colla.find()[0]._id})
WriteResult({ "nInserted" : 1 })
> db.FO_col2a.insert({nome: "passat", ano : 1988, chassi : "HIJ987KL25", cliente_id : db.FO_colla.find()[0]._id})
WriteResult({ "nInserted" : 1 })
> db.FO_col2a.insert({nome: "hilux", ano : 2021, chassi : "AAAA100000", cliente_id : db.FO_colla.find()[1]._id})
WriteResult({ "nInserted" : 1 })
> db.FO_col2a.insert({nome: "mclaren", ano : 2022, chassi : "MAC000001", cliente_id : db.FO_colla.find()[1]._id})
WriteResult({ "nInserted" : 1 })
```

Figura 15 – Lado N da cardinalidade 1-N

Comando para mostrar `_id` da coleção `FO_col1a` (clientes).

```
db.FO_colla.find({nome : "Daniel"}, {_id : 1})
db.FO_colla.find({nome : "Juliana"}, {_id : 1})
```

```
> db.FO_colla.find({nome : "Daniel"}, {_id : 1})
{ "_id" : ObjectId("639d111bf017c00e8dbe8a1a") }
> db.FO_colla.find({nome : "Juliana"}, {_id : 1})
{ "_id" : ObjectId("639d111bf017c00e8dbe8a1b") }
>
```

Figura 16 – Mostrando `_id` dos clientes

Comando para mostrar relação 1-N entre as coleções “`FO_col1a`” e “`FO_col2a`”.

```
db.FO_col2a.find().forEach(printjson)
```

```
> db.FO_col2a.find().forEach(printjson)
{
  "_id" : ObjectId("639d145cf017c00e8dbe8a20"),
  "nome" : "brasilvia",
  "ano" : 1982,
  "chassi" : "D12EFG45",
  "cliente_id" : ObjectId("639d111bf017c00e8dbe8a1a")
}
{
  "_id" : ObjectId("639d145cf017c00e8dbe8a21"),
  "nome" : "passat",
  "ano" : 1988,
  "chassi" : "HIJ987KL25",
  "cliente_id" : ObjectId("639d111bf017c00e8dbe8a1a")
}
{
  "_id" : ObjectId("639d145cf017c00e8dbe8a22"),
  "nome" : "hilux",
  "ano" : 2021,
  "chassi" : "AAAA100000",
  "cliente_id" : ObjectId("639d111bf017c00e8dbe8a1b")
}
{
  "_id" : ObjectId("639d145cf017c00e8dbe8a23"),
  "nome" : "mclaren",
  "ano" : 2022,
  "chassi" : "MAC000001",
  "cliente_id" : ObjectId("639d111bf017c00e8dbe8a1b")
}
>
```

Figura 17 – Mostrando relação 1-N entre as coleções

3. Criar um pequeno modelo que envolva duas coleções com as cardinalidades N-N. O nome das coleções deve ser "FO\_col1b" e "FO\_col2b". Nesse caso você indicará se usará a referência ou se embarcará o documento no outro documento.

a. Criar duas coleções de nome (embarcado).

Duas coleções: uma de médicos que será a coleção FO\_col1b e pacientes que será FO\_col2b.

```
db.FO_col1b.insert({ _id: "mrp001", nome:"Fernando",
celular:"99999-9999", pacientes: ["prm001","prm002"]})
db.FO_col1b.insert({ _id: "mrp002", nome:"Elaine",
celular:"77777-5555", pacientes: ["prm001","prm002"]})
db.FO_col2b.insert({ _id: "prm001", paciente:"Leandro",
cidade:"Curitiba", medicos: ["mrp001","mrp002"]})
db.FO_col2b.insert({ _id: "prm002", paciente:"Gislaine",
cidade:"Manaus", medicos: ["mrp001","mrp002"]})
```

```
> db.FO_col1b.insert ({ _id: "mrp001", nome:"Fernando", celular:"99999-9999", pacientes: ["prm001","prm002"]})
WriteResult({ "nInserted" : 1 })
> db.FO_col1b.insert ({ _id: "mrp002", nome:"Elaine", celular:"77777-5555", pacientes: ["prm001","prm002"]})
WriteResult({ "nInserted" : 1 })
> db.FO_col2b.insert({ _id: "prm001", paciente:"Leandro", cidade:"Curitiba", medicos: ["mrp001","mrp002"]})
WriteResult({ "nInserted" : 1 })
> db.FO_col2b.insert({ _id: "prm002", paciente:"Gislaine", cidade:"Manaus", medicos: ["mrp001","mrp002"]})
WriteResult({ "nInserted" : 1 })
>
```

Figura 18 – Criação das coleções com 2 documentos cada (relação N-N)

b. Mostrando os documentos de coleção FO\_col1b e FO\_col2b.

```
db.FO_col1b.find()
db.FO_col2b.find()
```

```
> db.FO_col1b.find()
{ "_id" : "mrp001", "nome" : "Fernando", "celular" : "99999-9999", "pacientes" : [ "prm001", "prm002" ] }
{ "_id" : "mrp002", "nome" : "Elaine", "celular" : "77777-5555", "pacientes" : [ "prm001", "prm002" ] }
> db.FO_col2b.find()
{ "_id" : "prm001", "paciente" : "Leandro", "cidade" : "Curitiba", "medicos" : [ "mrp001", "mrp002" ] }
{ "_id" : "prm002", "paciente" : "Gislaine", "cidade" : "Manaus", "medicos" : [ "mrp001", "mrp002" ] }
>
```

Figura 19 – Mostrando relação N-N entre as coleções

## C. Índices

### 1. Criação do database "FO\_indeagreg".

```
> use FO_indeagreg
switched to db FO_indeagreg
> db
FO_indeagreg
>
```

Figura 20 – Criação do database FO\_indeagreg

### 2. Criação da coleção de nome "FO\_indexar1" que apresente pelo menos 7 atributos (tem que ter pelo menos um atributo array).

```
db.FO_indexar1.insert ({ item : "Computador", marca : "Dell",
quantidade: 25, cor : ["preto", "branco"], estoque : "sim", loja :
"centro RJ", valor : 8000});
db.FO_indexar1.insert ({ item : "Computador", marca : "Samsung",
quantidade: 15, cor : ["preto", "branco"], estoque : "sim", loja :
"filial Curitiba", valor : 6000});
db.FO_indexar1.insert ({ item : "Celular", marca : "Apple",
quantidade: 18, cor : ["preto", "prata", "vermelho"], estoque :
"sim", loja : "centro RJ", valor : 6500});
db.FO_indexar1.insert ({ item : "Impressora", marca : "Canon",
quantidade: 5, cor : ["branco", "preto"], estoque : "sim", loja :
"filial BH", valor : 580});
```

```
> db.FO_indexar1.insert ({ item : "Computador", marca : "Dell", quantidade: 25, cor : ["preto", "branco"], estoque :
db.FO_indexar1.insert ({ item : "Impressora", marca : "Canon", quantidade: 5, cor : ["branco", "preto"], estoque :
teResult({ "nInserted" : 1 })
> db.FO_indexar1.insert ({ item : "Computador", marca : "Samsung", quantidade: 15, cor : ["preto", "branco"], estoqu
: 6000});
WriteResult({ "nInserted" : 1 })
> db.FO_indexar1.insert ({ item : "Celular", marca : "Apple", quantidade: 18, cor : ["preto", "prata", "vermelho"],
: 6500});
WriteResult({ "nInserted" : 1 })
> db.FO_indexar1.insert ({ item : "Impressora", marca : "Canon", quantidade: 5, cor : ["branco", "preto"], estoque :
WriteResult({ "nInserted" : 1 })
>
```

Figura 21 – Inserção de 4 documentos na coleção FO\_indexar1

#### a. Criação de um índice (não único e com ordenação ascendente).

```
db.FO_indexar1.createIndex({quantidade: 1})
```

```
> db.FO_indexar1.createIndex({quantidade: 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 6,
  "numIndexesAfter" : 7,
  "ok" : 1
}
```

Figura 22 – Índice não único com ordenação ascendente

b. Criação de um índice (único e com ordenação descendente).

```
db.FO_indexar1.createIndex({item: -1, marca: -1}, {unique: true});
```

```
> db.FO_indexar1.createIndex({item: -1, marca: -1}, {unique: true});
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
>
```

Figura 23 – Índice único com ordenação descendente

c. Criação de um índice (com dois atributos).

```
db.FO_indexar1.createIndex({item: 1, marca: 1})
```

```
> db.FO_indexar1.createIndex({item: 1, marca: 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
>
```

Figura 24 – Índice com dois atributos

d. Criação de um índice para um atributo array.

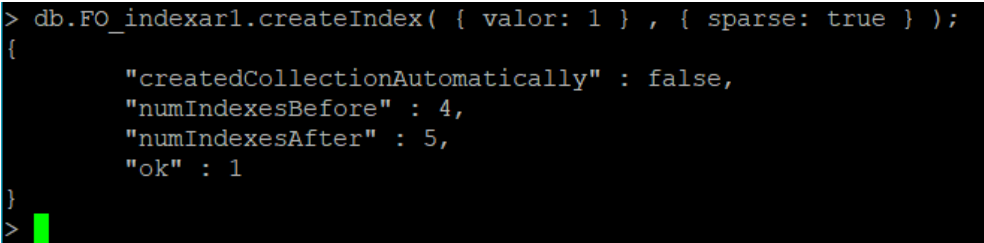
```
db.FO_indexar1.createIndex( { cor: 1 } )
```

```
> db.FO_indexar1.createIndex( { cor: 1 } )
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 3,
  "numIndexesAfter" : 4,
  "ok" : 1
}
>
```

Figura 25 – Índice para um atributo array

e. Criação de um índice esparsos (com um atributo).

```
db.FO_indexar1.createIndex( { valor: 1 } , { sparse: true }  
) ;
```

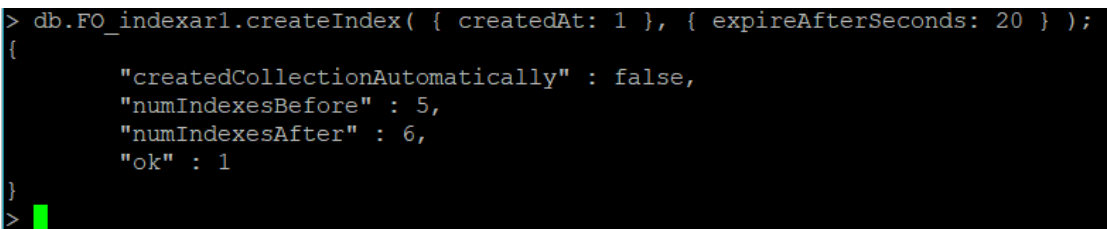


```
> db.FO_indexar1.createIndex( { valor: 1 } , { sparse: true } );  
{  
  "createdCollectionAutomatically" : false,  
  "numIndexesBefore" : 4,  
  "numIndexesAfter" : 5,  
  "ok" : 1  
}  
>
```

Figura 26 – Índice esparsos para atributo valor

f. Criação de um índice com tempo de vida - TTL (com um atributo e com expiração de 20 segundos).

```
db.FO_indexar1.createIndex( { createdAt: 1 } , {  
  expireAfterSeconds: 20 } );
```



```
> db.FO_indexar1.createIndex( { createdAt: 1 } , { expireAfterSeconds: 20 } );  
{  
  "createdCollectionAutomatically" : false,  
  "numIndexesBefore" : 5,  
  "numIndexesAfter" : 6,  
  "ok" : 1  
}  
>
```

Figura 27 – Índice com tempo de vida - TTL

g. Inserção de 4 documentos nesta coleção.

```

> db.FO_indexar1.find().forEach(printjson)
{
  "_id" : ObjectId("639e5c96f41e16720d155a9c"),
  "item" : "Computador",
  "marca" : "Dell",
  "quantidade" : 25,
  "cor" : [
    "preto",
    "branco"
  ],
  "estoque" : "sim",
  "loja" : "centro RJ",
  "valor" : 8000
}
{
  "_id" : ObjectId("639e5c96f41e16720d155a9d"),
  "item" : "Computador",
  "marca" : "Samsung",
  "quantidade" : 15,
  "cor" : [
    "preto",
    "branco"
  ],
  "estoque" : "sim",
  "loja" : "filial Curitiba",
  "valor" : 6000
}
{
  "_id" : ObjectId("639e5c96f41e16720d155a9e"),
  "item" : "Celular",
  "marca" : "Apple",
  "quantidade" : 18,
  "cor" : [
    "preto",
    "prata",
    "vermelho"
  ],
  "estoque" : "sim",
  "loja" : "centro RJ",
  "valor" : 6500
}
{
  "_id" : ObjectId("639e5c9cf41e16720d155a9f"),
  "item" : "Impressora",
  "marca" : "Canon",
  "quantidade" : 5,
  "cor" : [
    "branco",
    "preto"
  ],
  "estoque" : "sim",
  "loja" : "filial BH",
  "valor" : 580
}
>

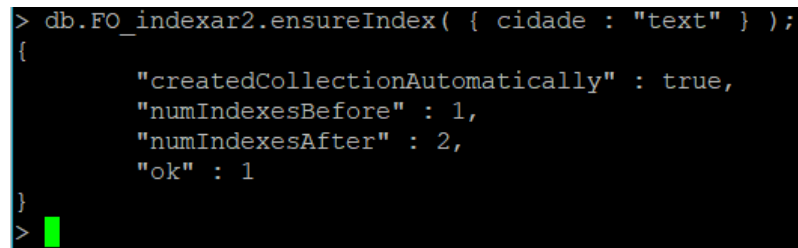
```

Figura 28 – Quatro documentos na coleção FO\_indexar1

3. Criação de uma segunda coleção de nome "FO\_indexar2" (com 4 atributos do tipo string)

a. Criação de um índice textual.

```
db.FO_indexar2.ensureIndex( { cidade : "text" } );
```

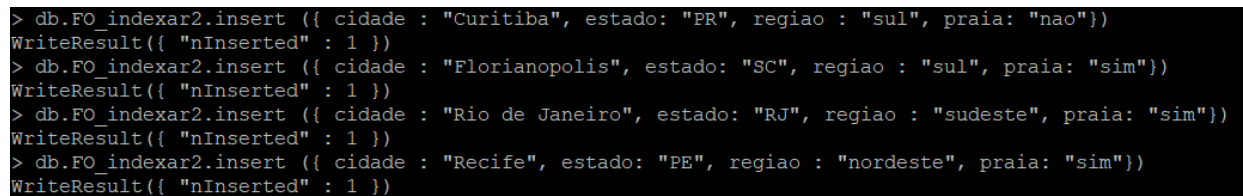


```
> db.FO_indexar2.ensureIndex( { cidade : "text" } );
{
  "createdCollectionAutomatically" : true,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
>
```

Figura 29 – Índice textual para a coleção FO\_indexar2

b. Inserção de 4 documentos nesta coleção.

```
db.FO_indexar2.insert ({ cidade : "Curitiba", estado: "PR",
regiao : "sul", praia: "nao"})
db.FO_indexar2.insert ({ cidade : "Florianopolis", estado:
"SC", regiao : "sul", praia: "sim"})
db.FO_indexar2.insert ({ cidade : "Rio de Janeiro", estado:
"RJ", regiao : "sudeste", praia: "sim"})
db.FO_indexar2.insert ({ cidade : "Recife", estado: "PE",
regiao : "nordeste", praia: "sim"})
```



```
> db.FO_indexar2.insert ({ cidade : "Curitiba", estado: "PR", regiao : "sul", praia: "nao"})
WriteResult({ "nInserted" : 1 })
> db.FO_indexar2.insert ({ cidade : "Florianopolis", estado: "SC", regiao : "sul", praia: "sim"})
WriteResult({ "nInserted" : 1 })
> db.FO_indexar2.insert ({ cidade : "Rio de Janeiro", estado: "RJ", regiao : "sudeste", praia: "sim"})
WriteResult({ "nInserted" : 1 })
> db.FO_indexar2.insert ({ cidade : "Recife", estado: "PE", regiao : "nordeste", praia: "sim"})
WriteResult({ "nInserted" : 1 })
```

Figura 30 – Quatro documentos na coleção FO\_indexar2

4. Criação de uma terceira coleção de nome "FO\_Venda" (com os atributos Cod\_Venda, UF\_Venda, Desc\_Prod\_Vendido, Valor\_Venda).

a. Inserção de 16 documentos (sendo 4 documentos para cada uma das UFs ).

```
db.FO_Venda.insert({Cod_Venda : "001", UF_Venda : "GO",
Desc_Prod_Vendido : 20, Valor_Venda : 600});
db.FO_Venda.insert({Cod_Venda : "002", UF_Venda : "GO",
Desc_Prod_Vendido : 15, Valor_Venda : 730});
db.FO_Venda.insert({Cod_Venda : "003", UF_Venda : "GO",
Desc_Prod_Vendido : 100, Valor_Venda : 1200});
```



```

db.FO_Venda.insert({Cod_Venda : "004", UF_Venda : "GO",
Desc_Prod_Vendido : 0, Valor_Venda : 2000});
db.FO_Venda.insert({Cod_Venda : "005", UF_Venda : "RS",
Desc_Prod_Vendido : 150, Valor_Venda : 3200});
db.FO_Venda.insert({Cod_Venda : "006", UF_Venda : "RS",
Desc_Prod_Vendido : 10, Valor_Venda : 200});
db.FO_Venda.insert({Cod_Venda : "007", UF_Venda : "RS",
Desc_Prod_Vendido : 150, Valor_Venda : 3200});
db.FO_Venda.insert({Cod_Venda : "008", UF_Venda : "RS",
Desc_Prod_Vendido : 80, Valor_Venda : 700});
db.FO_Venda.insert({Cod_Venda : "009", UF_Venda : "PR",
Desc_Prod_Vendido : 200, Valor_Venda : 5200});
db.FO_Venda.insert({Cod_Venda : "010", UF_Venda : "PR",
Desc_Prod_Vendido : 55, Valor_Venda : 330});
db.FO_Venda.insert({Cod_Venda : "011", UF_Venda : "PR",
Desc_Prod_Vendido : 5, Valor_Venda : 70});
db.FO_Venda.insert({Cod_Venda : "012", UF_Venda : "PR",
Desc_Prod_Vendido : 63, Valor_Venda : 3999});
db.FO_Venda.insert({Cod_Venda : "013", UF_Venda : "ES",
Desc_Prod_Vendido : 180, Valor_Venda : 1500});
db.FO_Venda.insert({Cod_Venda : "014", UF_Venda : "ES",
Desc_Prod_Vendido : 37, Valor_Venda : 622});
db.FO_Venda.insert({Cod_Venda : "015", UF_Venda : "ES",
Desc_Prod_Vendido : 998, Valor_Venda : 15700});
db.FO_Venda.insert({Cod_Venda : "016", UF_Venda : "ES",
Desc_Prod_Vendido : 78, Valor_Venda : 847});

db.FO_Venda.find({}, {_id : 0, Cod_Venda : 1, UF_Venda : 1,
Desc_Prod_Vendido : 1, Valor_Venda : 1})

```

```

> db.FO_Venda.find({}, {_id : 0, Cod_Venda : 1, UF_Venda : 1, Desc_Prod_Vendido : 1, Valor_Venda : 1})
{ "Cod_Venda" : "001", "UF_Venda" : "GO", "Desc_Prod_Vendido" : 20, "Valor_Venda" : 600 }
{ "Cod_Venda" : "002", "UF_Venda" : "GO", "Desc_Prod_Vendido" : 15, "Valor_Venda" : 730 }
{ "Cod_Venda" : "003", "UF_Venda" : "GO", "Desc_Prod_Vendido" : 100, "Valor_Venda" : 1200 }
{ "Cod_Venda" : "004", "UF_Venda" : "GO", "Desc_Prod_Vendido" : 0, "Valor_Venda" : 2000 }
{ "Cod_Venda" : "005", "UF_Venda" : "RS", "Desc_Prod_Vendido" : 150, "Valor_Venda" : 3200 }
{ "Cod_Venda" : "006", "UF_Venda" : "RS", "Desc_Prod_Vendido" : 10, "Valor_Venda" : 200 }
{ "Cod_Venda" : "007", "UF_Venda" : "RS", "Desc_Prod_Vendido" : 150, "Valor_Venda" : 3200 }
{ "Cod_Venda" : "008", "UF_Venda" : "RS", "Desc_Prod_Vendido" : 80, "Valor_Venda" : 700 }
{ "Cod_Venda" : "009", "UF_Venda" : "PR", "Desc_Prod_Vendido" : 200, "Valor_Venda" : 5200 }
{ "Cod_Venda" : "010", "UF_Venda" : "PR", "Desc_Prod_Vendido" : 55, "Valor_Venda" : 330 }
{ "Cod_Venda" : "011", "UF_Venda" : "PR", "Desc_Prod_Vendido" : 5, "Valor_Venda" : 70 }
{ "Cod_Venda" : "012", "UF_Venda" : "PR", "Desc_Prod_Vendido" : 63, "Valor_Venda" : 3999 }
{ "Cod_Venda" : "013", "UF_Venda" : "ES", "Desc_Prod_Vendido" : 180, "Valor_Venda" : 1500 }
{ "Cod_Venda" : "014", "UF_Venda" : "ES", "Desc_Prod_Vendido" : 37, "Valor_Venda" : 622 }
{ "Cod_Venda" : "015", "UF_Venda" : "ES", "Desc_Prod_Vendido" : 998, "Valor_Venda" : 15700 }
{ "Cod_Venda" : "016", "UF_Venda" : "ES", "Desc_Prod_Vendido" : 78, "Valor_Venda" : 847 }
>

```

Figura 31 – Mostrando 16 documentos na coleção FO\_Vendas

**b. Criação de uma consulta que mostra o número de documentos por UF.**

```
db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", total :
{$sum : 1} } } ] )
```

```
> db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", total : {$sum : 1} } } ] )
{ "_id" : "ES", "total" : 4 }
{ "_id" : "PR", "total" : 4 }
{ "_id" : "RS", "total" : 4 }
{ "_id" : "GO", "total" : 4 }
>
```

Figura 32 – Total de Documentos por UF

**c. Criação de uma consulta que mostra o valor total das vendas por UF.**

```
db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda",
total_vendas : {$sum : "$Valor_Venda" } } } ] )
```

```
> db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", total_vendas : {$sum : "$Valor_Venda" } } } ] )
{ "_id" : "ES", "total_vendas" : 18669 }
{ "_id" : "PR", "total_vendas" : 9599 }
{ "_id" : "RS", "total_vendas" : 7300 }
{ "_id" : "GO", "total_vendas" : 4530 }
>
```

Figura 33 – Total de Vendas por UF

**d. Criação de uma consulta que mostra o valor de médias das vendas por UF.**

```
db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda",
media_vendas : {$avg : "$Valor_Venda" } } } ] )
```

```
> db.FO_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", media_vendas : {$avg : "$Valor_Venda" } } } ] )
{ "_id" : "ES", "media_vendas" : 4667.25 }
{ "_id" : "PR", "media_vendas" : 2399.75 }
{ "_id" : "RS", "media_vendas" : 1825 }
{ "_id" : "GO", "media_vendas" : 1132.5 }
>
```

Figura 34 – Média de Vendas por UF

**e. Criação de uma consulta que mostra o maior valor de venda por UF.**

```
db.FO_Venda.aggregate([{$group:{_id : "$UF_Venda", maior_valor :
{$max : "$Valor_Venda"}}}])
```

```
> db.FO_Venda.aggregate([{$group:{_id : "$UF_Venda", maior_valor : {$max : "$Valor_Venda"}}}])
{ "_id" : "ES", "maior_valor" : 15700 }
{ "_id" : "PR", "maior_valor" : 5200 }
{ "_id" : "RS", "maior_valor" : 3200 }
{ "_id" : "GO", "maior_valor" : 2000 }
>
```

Figura 35 – Maior valor de Vendas por UF

f. Criação de uma consulta que mostra o menor valor de venda por UF.

```
db.FO_Venda.aggregate([{$group:{_id : "$UF_Venda", menor_valor :
{$min : "$Valor_Venda"}}}])
```

```
> db.FO_Venda.aggregate([{$group:{_id : "$UF_Venda", menor_valor : {$min : "$Valor_Venda"}}}])
{ "_id" : "ES", "menor_valor" : 622 }
{ "_id" : "PR", "menor_valor" : 70 }
{ "_id" : "RS", "menor_valor" : 200 }
{ "_id" : "GO", "menor_valor" : 600 }
>
```

Figura 36 – Menor valor de Vendas por UF

## D. Replicação

1. Criação do replica set "FO\_rposmit".

2. Desenho da arquitetura:

- a. Um primário;
- b. Dois secundários;
- c. Um árbitro.

1. Ativação dos processos no nível do Sistema Operacional;

```
ls -l /home/ubuntu/replica_set
```

```
ubuntu@ubuntu2004:~$ ls -l /home/ubuntu/replica_set
total 16
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 10:45 arbiter
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 10:45 node1
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 10:45 node2
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 10:45 node3
ubuntu@ubuntu2004:~$
```

Figura 37 – Criação dos diretórios do replica Set

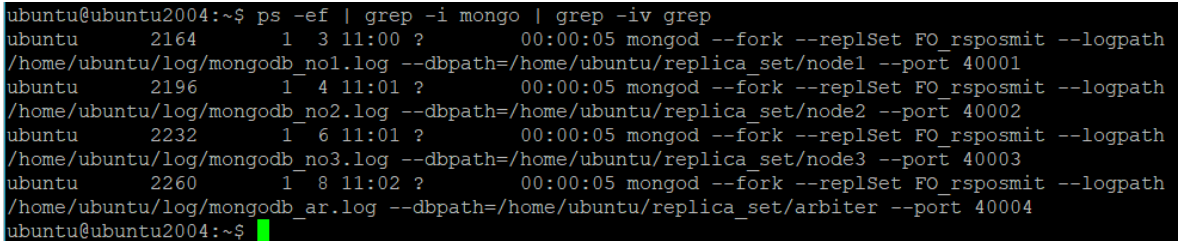
```
mongod --fork --replSet FO_rposmit --logpath
/home/ubuntu/log/mongodb_no1.log --
dbpath=/home/ubuntu/replica_set/node1 --port 40001
```

```

mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_no2.log --
dbpath=/home/ubuntu/replica_set/node2 --port 40002
mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_no3.log --
dbpath=/home/ubuntu/replica_set/node3 --port 40003
mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_ar.log --
dbpath=/home/ubuntu/replica_set/arbiter --port 40004

ps -ef | grep -i mongo | grep -iv grep

```



```

ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      2164      1   3 11:00 ?        00:00:05 mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_no1.log --dbpath=/home/ubuntu/replica_set/node1 --port 40001
ubuntu      2196      1   4 11:01 ?        00:00:05 mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_no2.log --dbpath=/home/ubuntu/replica_set/node2 --port 40002
ubuntu      2232      1   6 11:01 ?        00:00:05 mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_no3.log --dbpath=/home/ubuntu/replica_set/node3 --port 40003
ubuntu      2260      1   8 11:02 ?        00:00:05 mongod --fork --replSet FO_rsposmit --logpath
/home/ubuntu/log/mongodb_ar.log --dbpath=/home/ubuntu/replica_set/arbiter --port 40004
ubuntu@ubuntu2004:~$

```

Figura 38 – Ativação do MongoDB em background com parâmetro replSet, parâmetro logpath, parâmetro dbpath e parâmetro port

## 2. Ativação do replica set através do MongoDB, adicionando todos os membros ao replica set

```

mongo localhost:40001
use bdcad
rs.initiate()
exit

mongo localhost:40001
use bdcad
rs.add('localhost:40002')
rs.add('localhost:40003')
rs.addArb('localhost:40004')
rs.status()

```

```

"members" : [
  {
    "_id" : 0,
    "name" : "localhost:40001",
    "health" : 1,
    "state" : 1,
    "stateStr" : "PRIMARY",
    "uptime" : 116,
    "optime" : {
      "ts" : Timestamp(1671467638, 1),
      "t" : NumberLong(2)
    },
    "optimeDate" : ISODate("2022-12-19T16:33:58Z"),
    "syncingTo" : "",
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "infoMessage" : "could not find member to sync from",
    "electionTime" : Timestamp(1671467555, 1),
    "electionDate" : ISODate("2022-12-19T16:32:35Z"),
    "configVersion" : 6,
    "self" : true,
    "lastHeartbeatMessage" : ""
  },

```

Figura 39 – Parte do comando rs.status() – Mostrando localhost:40001

```

{
  "_id" : 1,
  "name" : "localhost:40002",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 99,
  "optime" : {
    "ts" : Timestamp(1671467638, 1),
    "t" : NumberLong(2)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1671467638, 1),
    "t" : NumberLong(2)
  },
  "optimeDate" : ISODate("2022-12-19T16:33:58Z"),
  "optimeDurableDate" : ISODate("2022-12-19T16:33:58Z"),
  "lastHeartbeat" : ISODate("2022-12-19T16:34:08.566Z"),
  "lastHeartbeatRecv" : ISODate("2022-12-19T16:34:09.090Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "infoMessage" : "",
  "configVersion" : 6
},

```

Figura 40 – Parte do comando rs.status() – Mostrando localhost:40002

```
{
  "_id" : 2,
  "name" : "localhost:40003",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 91,
  "optime" : {
    "ts" : Timestamp(1671467638, 1),
    "t" : NumberLong(2)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1671467638, 1),
    "t" : NumberLong(2)
  },
  "optimeDate" : ISODate("2022-12-19T16:33:58Z"),
  "optimeDurableDate" : ISODate("2022-12-19T16:33:58Z"),
  "lastHeartbeat" : ISODate("2022-12-19T16:34:08.579Z"),
  "lastHeartbeatRecv" : ISODate("2022-12-19T16:34:09.097Z"),
  "pingMs" : NumberLong(2),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "infoMessage" : "",
  "configVersion" : 6
},
```

Figura 41 – Parte do comando rs.status() – Mostrando localhost:40003

```
{
  "_id" : 3,
  "name" : "localhost:40004",
  "health" : 1,
  "state" : 7,
  "stateStr" : "ARBITER",
  "uptime" : 11,
  "lastHeartbeat" : ISODate("2022-12-19T16:34:00.130Z"),
  "lastHeartbeatRecv" : ISODate("2022-12-19T16:34:00.129Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "infoMessage" : "",
  "configVersion" : 6
}
```

Figura 42 – Parte do comando rs.status() – Mostrando localhost:40004

Configuração do localhost:40002 (segunda janela do terminal) e localhost:40003 (terceira janela do terminal) para permissão de leitura do nó secundário:

```
mongo localhost:40002
use bdcad
rs.slaveOk()
mongo localhost:40003
use bdcad
rs.slaveOk()
```

```

ubuntu@ubuntu2004:~$ mongo localhost:40002
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:40002/test
Implicit session: session { "id" : UUID("53333333-3333-3333-3333-333333333333") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-19T11:58:05.377-0500 I STORAGE [initandlisten]
2022-12-19T11:58:05.378-0500 I STORAGE [initandlisten]
2022-12-19T11:58:05.378-0500 I STORAGE [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
2022-12-19T11:58:06.352-0500 I CONTROL [initandlisten]
FO_rsposmit:SECONDARY> use bdcad
switched to db bdcad
FO_rsposmit:SECONDARY> rs.slaveOk()

```

Figura 43 - localhost:40002 – Permissão de leitura para o nó secundário

```

ubuntu@ubuntu2004:~$ mongo localhost:40003
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:40003/test
Implicit session: session { "id" : UUID("aacee333-3333-3333-3333-333333333333") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-19T11:58:27.622-0500 I STORAGE [initandlisten]
2022-12-19T11:58:27.622-0500 I STORAGE [initandlisten]
2022-12-19T11:58:27.622-0500 I STORAGE [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
2022-12-19T11:58:28.381-0500 I CONTROL [initandlisten]
FO_rsposmit:SECONDARY> use bdcad
switched to db bdcad
FO_rsposmit:SECONDARY> rs.slaveOk()

```

Figura 44 - localhost:40003 – Permissão de leitura para o nó secundário



### 3. No nó primário criar a coleção "FO\_col\_filmes".

#### a. Inserção de 5 documentos na coleção "FO\_col\_filmes".

```
mongo localhost:40001
use bdcad
for(i=1; i<6; i++){ db.FO_col_filmes.save({Filme : "Filme "+i}) };
db.FO_col_filmes.find()
```

```
FO_rsposmit:PRIMARY> for(i=1; i<6; i++) { db.FO_col_filmes.save({Filme : "Filme "+i}) };
WriteResult({ "nInserted" : 1 })
FO_rsposmit:PRIMARY> db.FO_col_filmes.find()
{ "_id" : ObjectId("63a09e0e5bdc2eb780156ad9"), "Filme" : "Filme 1" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156ada"), "Filme" : "Filme 2" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156adb"), "Filme" : "Filme 3" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156adc"), "Filme" : "Filme 4" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156add"), "Filme" : "Filme 5" }
FO_rsposmit:PRIMARY>
```

Figura 45 - Cinco documentos na coleção FO\_col\_filmes - localhost:40001

### 4. Acesso ao nó secundário localhost:40002

#### a. Leitura da coleção "FO\_col\_filmes".

```
db.FO_col_filmes.find()
```

```
FO_rsposmit:SECONDARY> db.FO_col_filmes.find()
{ "_id" : ObjectId("63a09e0e5bdc2eb780156ad9"), "Filme" : "Filme 1" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156adb"), "Filme" : "Filme 3" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156ada"), "Filme" : "Filme 2" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156adc"), "Filme" : "Filme 4" }
{ "_id" : ObjectId("63a09e0e5bdc2eb780156add"), "Filme" : "Filme 5" }
FO_rsposmit:SECONDARY>
```

Figura 46 - Leitura dos Cinco documentos na coleção FO\_col\_filmes - localhost:40002

## E. Particionamento

### 1. Criar um shard.

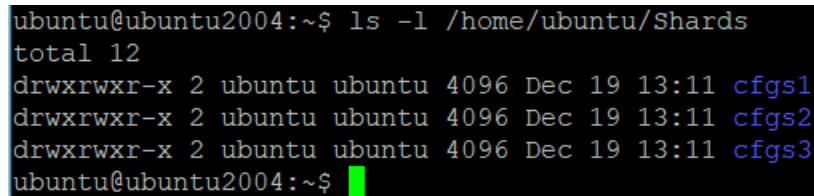
### 2. Desenho da arquitetura:

- Três config servers (no Replica Set);
- Quatro shard servers (sem Replica Set);
- Um mongos.



## 1) Ativação dos processos do Config Servers.

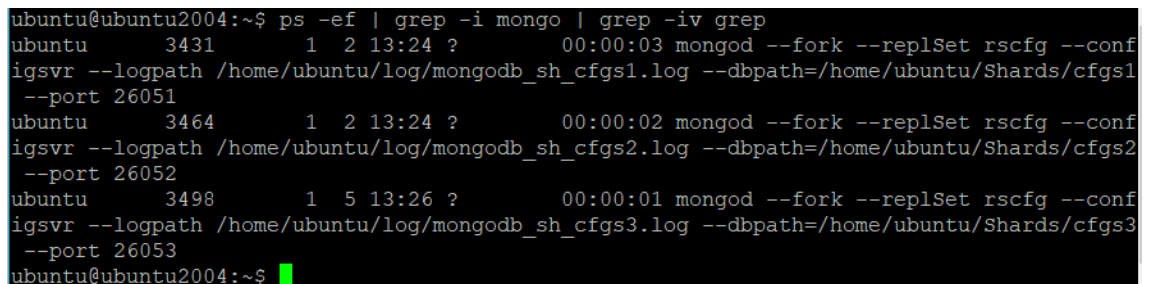
```
ls -l /home/ubuntu/Shards
```



```
ubuntu@ubuntu2004:~$ ls -l /home/ubuntu/Shards
total 12
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 13:11 cfgs1
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 13:11 cfgs2
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 19 13:11 cfgs3
ubuntu@ubuntu2004:~$
```

Figura 47 - Criação dos diretórios de particionamentos para Config Servers

```
mongod -fork -replSet rscfg -configsvr -logpath
/home/ubuntu/log/mongodb_sh_cfgs1.log -
dbpath=/home/ubuntu/Shards/cfgs1 -port 26051
mongod -fork -replSet rscfg -configsvr -logpath
/home/ubuntu/log/mongodb_sh_cfgs2.log -
dbpath=/home/ubuntu/Shards/cfgs2 -port 26052
mongod -fork -replSet rscfg -configsvr -logpath
/home/ubuntu/log/mongodb_sh_cfgs3.log -
dbpath=/home/ubuntu/Shards/cfgs3 -port 26053
```



```
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      3431      1  2 13:24 ?        00:00:03 mongod --fork --replSet rscfg --conf
igsrvr --logpath /home/ubuntu/log/mongodb_sh_cfgs1.log --dbpath=/home/ubuntu/Shards/cfgs1
--port 26051
ubuntu      3464      1  2 13:24 ?        00:00:02 mongod --fork --replSet rscfg --conf
igsrvr --logpath /home/ubuntu/log/mongodb_sh_cfgs2.log --dbpath=/home/ubuntu/Shards/cfgs2
--port 26052
ubuntu      3498      1  5 13:26 ?        00:00:01 mongod --fork --replSet rscfg --conf
igsrvr --logpath /home/ubuntu/log/mongodb_sh_cfgs3.log --dbpath=/home/ubuntu/Shards/cfgs3
--port 26053
ubuntu@ubuntu2004:~$
```

Figura 48 - Ativação do MongoDB em background com parâmetro Config Server, parâmetro logpath, parâmetro dbpath, parâmetro port e com replica set para o Config Server

```
mongo -port 26051
```

```
rs.initiate(
{
  _id: "rscfg",
  configsvr: true,
  members: [
    { _id : 0, host : "localhost:26051" },
    { _id : 1, host : "localhost:26052" },
    { _id : 2, host : "localhost:26053" }
```

```

    ]
  }
)

```

```

{
  "_id" : 0,
  "name" : "localhost:26051",
  "health" : 1,
  "state" : 1,
  "stateStr" : "PRIMARY",
  "uptime" : 704,
  "optime" : {
    "ts" : Timestamp(1671474973, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2022-12-19T18:36:13Z"),
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "infoMessage" : "could not find member to sync from",
  "electionTime" : Timestamp(1671474901, 1),
  "electionDate" : ISODate("2022-12-19T18:35:01Z"),
  "configVersion" : 1,
  "self" : true,
  "lastHeartbeatMessage" : ""
},

```

Figura 49 - Config Server – localhost:26051

```

{
  "_id" : 1,
  "name" : "localhost:26052",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 84,
  "optime" : {
    "ts" : Timestamp(1671474962, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1671474962, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2022-12-19T18:36:02Z"),
  "optimeDurableDate" : ISODate("2022-12-19T18:36:02Z"),
  "lastHeartbeat" : ISODate("2022-12-19T18:36:13.878Z"),
  "lastHeartbeatRecv" : ISODate("2022-12-19T18:36:12.861Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "localhost:26051",
  "syncSourceHost" : "localhost:26051",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1
},

```

Figura 50 - Config Server – localhost:26052

```
{
  "_id" : 2,
  "name" : "localhost:26053",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 84,
  "optime" : {
    "ts" : Timestamp(1671474962, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1671474962, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2022-12-19T18:36:02Z"),
  "optimeDurableDate" : ISODate("2022-12-19T18:36:02Z"),
  "lastHeartbeat" : ISODate("2022-12-19T18:36:13.879Z"),
  "lastHeartbeatRecv" : ISODate("2022-12-19T18:36:12.873Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "localhost:26051",
  "syncSourceHost" : "localhost:26051",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 1
}
```

Figura 51 - Config Server – localhost:26053

## 2) Ativação dos processos do Shard Servers.

```
mkdir -p /home/ubuntu/Shards/s1
mkdir -p /home/ubuntu/Shards/s2
mkdir -p /home/ubuntu/Shards/s3
mkdir -p /home/ubuntu/Shards/s4
```

```
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shards/s1
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shards/s2
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shards/s3
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shards/s4
ubuntu@ubuntu2004:~$
```

Figura 52 - Criação dos diretórios de particionamentos para Shards

```
mongod --fork --shardsvr --logpath
/home/ubuntu/log/mongodb_shs_ss1.log --
dbpath=/home/ubuntu/Shards/s1 --port 27051
mongod --fork --shardsvr --logpath
/home/ubuntu/log/mongodb_shs_ss2.log --
dbpath=/home/ubuntu/Shards/s2 --port 27052
```

```

mongod --fork --shardsvr --logpath
/home/ubuntu/log/mongodb_shs_ss3.log --
dbpath=/home/ubuntu/Shards/s3 --port 27053
mongod --fork --shardsvr --logpath
/home/ubuntu/log/mongodb_shs_ss4.log --
dbpath=/home/ubuntu/Shards/s4 --port 27054

```



```

ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu    3431      1  2 13:24 ?        00:00:36 mongod --fork --replSet rscfg -
-configsvr --logpath /home/ubuntu/log/mongodb_sh_cfgs1.log --dbpath=/home/ubuntu/Sh
ards/cfgs1 --port 26051
ubuntu    3464      1  2 13:24 ?        00:00:38 mongod --fork --replSet rscfg -
-configsvr --logpath /home/ubuntu/log/mongodb_sh_cfgs2.log --dbpath=/home/ubuntu/Sh
ards/cfgs2 --port 26052
ubuntu    3498      1  2 13:26 ?        00:00:37 mongod --fork --replSet rscfg -
-configsvr --logpath /home/ubuntu/log/mongodb_sh_cfgs3.log --dbpath=/home/ubuntu/Sh
ards/cfgs3 --port 26053
ubuntu    3766      1  2 13:48 ?        00:00:03 mongod --fork --shardsvr --logp
ath /home/ubuntu/log/mongodb_shs_ss1.log --dbpath=/home/ubuntu/Shards/s1 --port 270
51
ubuntu    3794      1  2 13:48 ?        00:00:03 mongod --fork --shardsvr --logp
ath /home/ubuntu/log/mongodb_shs_ss2.log --dbpath=/home/ubuntu/Shards/s2 --port 270
52
ubuntu    3820      1  2 13:49 ?        00:00:03 mongod --fork --shardsvr --logp
ath /home/ubuntu/log/mongodb_shs_ss3.log --dbpath=/home/ubuntu/Shards/s3 --port 270
53
ubuntu    3845      1  2 13:49 ?        00:00:03 mongod --fork --shardsvr --logp
ath /home/ubuntu/log/mongodb_shs_ss4.log --dbpath=/home/ubuntu/Shards/s4 --port 270
54
ubuntu@ubuntu2004:~$

```

Figura 53 - Ativação do MongoDB com o parâmetro Shard Server

### 3. Conectar ao mongos

```

mongos --fork --configdb
"rscfg/localhost:26051,localhost:26052,localhost:26053" --
logpath /home/ubuntu/log/mongos_sh.log --port 28000

mongo localhost:28000

use config
sh.addShard("localhost:27051")
sh.addShard("localhost:27052")
sh.addShard("localhost:27053")
sh.addShard("localhost:27054")

sh.status()

```

```

mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("63a0aed7c689090865bc008c")
  }
  shards:
    { "_id" : "shard0000", "host" : "localhost:27051", "state" : 1 }
    { "_id" : "shard0001", "host" : "localhost:27052", "state" : 1 }
    { "_id" : "shard0002", "host" : "localhost:27053", "state" : 1 }
    { "_id" : "shard0003", "host" : "localhost:27054", "state" : 1 }
  active mongoses:
    "3.6.8" : 1
  autosplit:
    Currently enabled: yes
  balancer:
    Currently enabled: yes
    Currently running: no

```

Figura 54 - Status do shard – mongos

```

mongos> db.shards.find()
{ "_id" : "shard0000", "host" : "localhost:27051", "state" : 1 }
{ "_id" : "shard0001", "host" : "localhost:27052", "state" : 1 }
{ "_id" : "shard0002", "host" : "localhost:27053", "state" : 1 }
{ "_id" : "shard0003", "host" : "localhost:27054", "state" : 1 }
mongos>

```

Figura 55 - Coleção de shards – mongos

a. Ativação do particionamento para o database 'itens\_partic'.

```

use itens_partic
sh.enableSharding ("itens_partic")

```

```

mongos> use itens_partic
switched to db itens_partic
mongos> db
itens_partic
mongos> sh.enableSharding ("itens_partic")
{
  "ok" : 1,
  "operationTime" : Timestamp(1671482394, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1671482394, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

Figura 56 - Particionamento do database itens\_partic

## b. Particionamento da coleção 'produtos'.

```
sh.shardCollection ("itens_partic.produtos", {_id:1}, true)
```

```
mongos> sh.shardCollection ("itens_partic.produtos", {_id:1}, true)
{
  "collectionsharded" : "itens_partic.produtos",
  "collectionUUID" : UUID("7332d8ed-7ba2-4a3e-be0e-b75120971f92"),
  "ok" : 1,
  "operationTime" : Timestamp(1671482639, 7),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1671482639, 7),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>
```

Figura 57 - Particionamento da coleção produtos no database itens\_partic

## c. Inserção de 1.000 documentos para essa coleção através do comando [for].

```
for (var i=0; i < 1000; i++ ) { db.produtos.insert ( { item : i,
cod: i + 11, qtde : i * 5 } ) }
db.produtos.find()
```

```
mongos> for (var i=0; i < 1000; i++ ) { db.produtos.insert ( { item : i, cod: i + 11, qtde : i * 5 } ) }
WriteResult({ "nInserted" : 1 })
mongos> db.produtos.find()
{ "_id" : ObjectId("63a0ce7776ba64186245db56"), "item" : 0, "cod" : 11, "qtde" : 0 }
{ "_id" : ObjectId("63a0ce7776ba64186245db57"), "item" : 1, "cod" : 12, "qtde" : 5 }
{ "_id" : ObjectId("63a0ce7776ba64186245db58"), "item" : 2, "cod" : 13, "qtde" : 10 }
{ "_id" : ObjectId("63a0ce7776ba64186245db59"), "item" : 3, "cod" : 14, "qtde" : 15 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5a"), "item" : 4, "cod" : 15, "qtde" : 20 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5b"), "item" : 5, "cod" : 16, "qtde" : 25 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5c"), "item" : 6, "cod" : 17, "qtde" : 30 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5d"), "item" : 7, "cod" : 18, "qtde" : 35 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5e"), "item" : 8, "cod" : 19, "qtde" : 40 }
{ "_id" : ObjectId("63a0ce7776ba64186245db5f"), "item" : 9, "cod" : 20, "qtde" : 45 }
{ "_id" : ObjectId("63a0ce7776ba64186245db60"), "item" : 10, "cod" : 21, "qtde" : 50 }
{ "_id" : ObjectId("63a0ce7776ba64186245db61"), "item" : 11, "cod" : 22, "qtde" : 55 }
{ "_id" : ObjectId("63a0ce7776ba64186245db62"), "item" : 12, "cod" : 23, "qtde" : 60 }
{ "_id" : ObjectId("63a0ce7776ba64186245db63"), "item" : 13, "cod" : 24, "qtde" : 65 }
{ "_id" : ObjectId("63a0ce7776ba64186245db64"), "item" : 14, "cod" : 25, "qtde" : 70 }
{ "_id" : ObjectId("63a0ce7776ba64186245db65"), "item" : 15, "cod" : 26, "qtde" : 75 }
{ "_id" : ObjectId("63a0ce7776ba64186245db66"), "item" : 16, "cod" : 27, "qtde" : 80 }
{ "_id" : ObjectId("63a0ce7776ba64186245db67"), "item" : 17, "cod" : 28, "qtde" : 85 }
{ "_id" : ObjectId("63a0ce7776ba64186245db68"), "item" : 18, "cod" : 29, "qtde" : 90 }
{ "_id" : ObjectId("63a0ce7776ba64186245db69"), "item" : 19, "cod" : 30, "qtde" : 95 }
Type "it" for more
mongos>
```

Figura 58 - Inserção de mil documentos na coleção 'produtos'

## d. Mostrando a distribuição da coleção criada.

```
db.produtos.getShardDistribution()
```

```

mongos> db.produtos.getShardDistribution()

Shard shard0003 at localhost:27054
  data : 61KiB docs : 1000 chunks : 1
  estimated data per chunk : 61KiB
  estimated docs per chunk : 1000

Totals
  data : 61KiB docs : 1000 chunks : 1
  Shard shard0003 contains 100% data, 100% docs in cluster, avg obj size on shard : 63B

mongos> █

```

Figura 59 - Distribuição da coleção 'produtos'

## F. Storage Engines

### 1. Criação de uma instância do MongoDB que usa o storage engine mmapv1.

#### a. Conectar a essa instância.

```

mkdir /home/ubuntu/ste_mmap
mongod --fork --storageEngine mmapv1 --logpath
/home/ubuntu/log/mongodb_ste_mmap.log --
dbpath=/home/ubuntu/ste_mmap --port 28000
ps -ef | grep -i mongo | grep -iv grep

```

```

ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      5168      1 10 18:17 ?        00:00:19 mongod --fork --storageEngine mmapv1 --logpath /home/ubuntu/log/mongodb_ste_mmap.log --dbpa
th=/home/ubuntu/ste_mmap --port 28000
ubuntu@ubuntu2004:~$ █

```

Figura 60 - Instância do MongoDB com storage engine mmapv1

#### b. Verificação do storage engine corrente.

```

ls -l /home/ubuntu/ste_mmap
mongo localhost:28000/ste_mmap
db.serverStatus().storageEngine

```

```

> db.serverStatus().storageEngine
{
  "name" : "mmapv1",
  "supportsCommittedReads" : false,
  "readOnly" : false,
  "persistent" : true
}
> █

```

Figura 61 - Verificação do storage engine corrente

#### c. Criação da coleção "FO\_produtos" e inserção de 4 documentos.

```

for (var i = 1; i < 5; i++) (db.FO_produtos.insert( { x : i * 3 ,
y: "Teste", cor: "cor "+i} ))

```

```
db.FO_produtos.find()
```

```
> for (var i = 1; i < 5; i++) (db.FO_produtos.insert( { x : i * 3 , y: "Teste", cor: "cor "+i} ))
WriteResult({ "nInserted" : 1 })
> db.FO_produtos.find()
{ "_id" : ObjectId("63a0f4a6a0bfc4035966da4e"), "x" : 3, "y" : "Teste", "cor" : "cor 1" }
{ "_id" : ObjectId("63a0f4a6a0bfc4035966da4f"), "x" : 6, "y" : "Teste", "cor" : "cor 2" }
{ "_id" : ObjectId("63a0f4a6a0bfc4035966da50"), "x" : 9, "y" : "Teste", "cor" : "cor 3" }
{ "_id" : ObjectId("63a0f4a6a0bfc4035966da51"), "x" : 12, "y" : "Teste", "cor" : "cor 4" }
>
```

Figura 62 - Coleção 'FO\_produtos' com quatro documentos

## 2. Criação de uma instância do MongoDB que usa o storage engine wiredTiger.

### a. Conectar a essa instância.

```
mkdir /home/ubuntu/se_wtiger
mongod --fork --storageEngine wiredTiger --logpath
/home/ubuntu/log/mongodb_se_wtiger.log --
dbpath=/home/ubuntu/se_wtiger --port 28000
ps -ef | grep -i mongo | grep -iv grep
```

```
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      5230      1 19 18:44 ?        00:00:02 mongod --fork --storageEngine wiredTiger --log
--dbpath=/home/ubuntu/se_wtiger --port 28000
ubuntu@ubuntu2004:~$
```

Figura 63 - Instância do MongoDB com storage wiredTiger

### b. Verificação do storage engine corrente.

```
ls -l /home/ubuntu/se_wtiger
mongo localhost:28000/se_wtiger
db.serverStatus().storageEngine
```

```
> db.serverStatus().storageEngine
{
  "name" : "wiredTiger",
  "supportsCommittedReads" : true,
  "readOnly" : false,
  "persistent" : true
}
>
```

Figura 64 - Verificação do storage engine corrente



c. Criação da coleção “FO\_lugares” e inserção de 4 documentos.

```
for (var i = 1; i <5; i++) (db.FO_lugares.insert( { cidade :
"Cidade "+i, estado : "Estado "+i } ))
db.FO_lugares.find()
```

```
> for (var i = 1; i <5; i++) (db.FO_lugares.insert( { cidade : "Cidade "+i, estado : "Estado "+i } ))
WriteResult({ "nInserted" : 1 })
> db.FO_lugares.find()
{ "_id" : ObjectId("63a0f9844f1b29d8c55aa851"), "cidade" : "Cidade 1", "estado" : "Estado 1" }
{ "_id" : ObjectId("63a0f9844f1b29d8c55aa852"), "cidade" : "Cidade 2", "estado" : "Estado 2" }
{ "_id" : ObjectId("63a0f9844f1b29d8c55aa853"), "cidade" : "Cidade 3", "estado" : "Estado 3" }
{ "_id" : ObjectId("63a0f9844f1b29d8c55aa854"), "cidade" : "Cidade 4", "estado" : "Estado 4" }
>
```

Figura 65 - Coleção ‘FO\_lugares’ com quatro documentos

3. Criação de uma instância do MongoDB que usa segurança (autenticação e autorização).

a. Conectar a essa instância.

```
mkdir /home/ubuntu/seguro
mongod --fork --logpath /home/ubuntu/log/mongodb_seguro.log --
dbpath=/home/ubuntu/seguro --port 30000 --auth
ps -ef | grep -i mongo | grep -iv grep
```

```
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      5280      1  22 19:02 ?        00:00:02 mongod --fork --logpath /home/
port 30000 --auth
ubuntu@ubuntu2004:~$
```

Figura 66 - Instância do MongoDB que usa segurança

b. Criação do usuário dba com a role root.

```
mongo localhost:30000/seguros
use admin
var usuariodba = { user : "dba", pwd : "senhadba", roles : [
"root" ]}
db.createUser (usuariodba)
```

```
> var usuariodba = { user : "dba", pwd : "senhadba", roles : [ "root" ]}
> db.createUser (usuariodba)
Successfully added user: { "user" : "dba", "roles" : [ "root" ] }
>
```

Figura 67 - Criação do usuário dba

c. Conectar com o usuário dba.

```
mongo localhost:30000/admin -u dba -p senhadba
show databases
```

```

ubuntu@ubuntu2004:~$ mongo localhost:30000/admin -u dba -p senhadba
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/admin
Implicit session: session { "id" : UUID("4911a111-d8c8-4616-97cf-e464dc59f47") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-19T19:02:23.250-0500 I STORAGE [initandlisten]
2022-12-19T19:02:23.250-0500 I STORAGE [initandlisten] ** WARNING: Using the
e engine
2022-12-19T19:02:23.250-0500 I STORAGE [initandlisten] ** See http
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten]
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** WARNING: This ser
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** Remote s
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** Start th
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** addresse
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** bind to
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten] ** server w
2022-12-19T19:02:24.990-0500 I CONTROL [initandlisten]
> show databases
admin 0.000GB
config 0.000GB
local 0.000GB
>

```

Figura 68 – Usuário dba conectado

d. Criação do usuário “FO\_desenv” com a role readWrite no database “FO\_rh”.

```

use FO_rh
var usuariodesenv = { user : "FO_desenv", pwd : "senhadesenv1",
roles : [ "readWrite" ] }
db.createUser (usuariodesenv)

```

```

> use FO_rh
switched to db FO_rh
> var usuariodesenv = { user : "FO_desenv", pwd : "senhadesenv1", roles : [ "readWrite" ] }
> db.createUser (usuariodesenv)
Successfully added user: { "user" : "FO_desenv", "roles" : [ "readWrite" ] }
>

```

Figura 69 - Criação do usuário ‘FO\_desenv’

e. Conectar com o usuário “FO\_desenv”.

```

mongo localhost:30000/FO_rh -u FO_desenv -p senhadesenv1

```

```

ubuntu@ubuntu2004:~$ mongo localhost:30000/FO_rh -u FO_desenv -p senhadesenv1
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/FO_rh
Implicit session: session { "id" : UUID("5cc16731-30fc-450b-9008-70514af096c0") }
MongoDB server version: 3.6.8
> db
FO_rh
>

```

Figura 70 – Usuário FO\_desenv conectado

## f. Criação da coleção “FO\_Funcionarios” e inserção de 4 documentos.

```
for (var i = 1; i <5; i++) (db.FO_Funcionarios.insert( { nome :
"nome "+i, cpf : "cpf "+(i*500000) } ))
db.FO_Funcionarios.find()
> for (var i = 1; i <5; i++) (db.FO_Funcionarios.insert( { nome : "nome "+i, cpf : "cpf "+(i*500000) } ))
WriteResult({ "nInserted" : 1 })
> db.FO_Funcionarios.find()
{ "_id" : ObjectId("63a102b7c3b63bac16f6b1a6"), "nome" : "nome 1", "cpf" : "cpf 500000" }
{ "_id" : ObjectId("63a102b7c3b63bac16f6b1a7"), "nome" : "nome 2", "cpf" : "cpf 1000000" }
{ "_id" : ObjectId("63a102b7c3b63bac16f6b1a8"), "nome" : "nome 3", "cpf" : "cpf 1500000" }
{ "_id" : ObjectId("63a102b7c3b63bac16f6b1a9"), "nome" : "nome 4", "cpf" : "cpf 2000000" }
> db.FO_Funcionarios.find()
```

Figura 71 - Coleção ‘FO\_Funcionarios’ com quatro documentos

## G. Depuração, Backup/Restore

### 1. Criação de uma instância do MongoDB.

```
mkdir /home/ubuntu/tunn
mongod --fork --logpath /home/ubuntu/log/mongodb_tunn.log --
dbpath=/home/ubuntu/tunn --port 32000
ps -ef | grep -i mongo | grep -iv grep
```

```
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      2266      1  2 20:34 ?        00:00:00 mongod --fork --logpath /home/ubuntu/log/mongod
godb_tunn.log --dbpath=/home/ubuntu/tunn --port 32000
ubuntu@ubuntu2004:~$
```

Figura 72 - Criação de uma instância no MongoDB

### 2. Conectar a essa instância (acessar o database exerc4b).

```
mongo localhost:32000/exerc4b
db
```

```
ubuntu@ubuntu2004:~$ mongo localhost:32000/exerc4b
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:32000/exerc4b
Implicit session: session { "id" : UUID("42f64735-bc00-4
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-20T20:34:27.849-0500 I STORAGE [initandlisten]
2022-12-20T20:34:27.849-0500 I STORAGE [initandlisten]
strongly recommended with the WiredTiger storage engine
2022-12-20T20:34:27.849-0500 I STORAGE [initandlisten]
g/core/prodnotes-filesystem
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
d for the database.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
a and configuration is unrestricted.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
lhost.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
e to connect to this server.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
ip <address> to specify which IP
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
sponses from, or with --bind_ip_all to
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
his behavior is desired, start the
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
0.1 to disable this warning.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
> db
exerc4b
>
```

Figura 73 - localhost:32000/exerc4b

### 3. Criação da coleção "FO\_Paises" e inserção de 4 documentos.

```
db.FO_Paises.insert({_id : 1, pais : "Japao", continente :
"Oceania"})
db.FO_Paises.insert({_id : 2, pais : "Brasil", continente :
"America do Sul", moeda : "real"})
db.FO_Paises.insert({_id : 3, pais : "Canada", continente :
"America do Norte", moeda : "dolar canadense"})
db.FO_Paises.insert({_id : 4, pais : "Estados Unidos", continente :
"America do Norte", moeda : "dolar"})
db.FO_Paises.find()
```

```
> db.FO_Paises.find()
{ "_id" : 1, "pais" : "Japao", "continente" : "Oceania" }
{ "_id" : 2, "pais" : "Brasil", "continente" : "America do Sul", "moeda" : "real" }
{ "_id" : 3, "pais" : "Canada", "continente" : "America do Norte", "moeda" : "dolar canadense" }
{ "_id" : 4, "pais" : "Estados Unidos", "continente" : "America do Norte", "moeda" : "dolar" }
>
```

Figura 74 - Coleção 'FO\_Paises'

### 4. Executar uma consulta para a coleção "FO\_Paises" com um filtro de sua escolha com o explain.

```
> db.FO_Paises.find({pais : "Brasil"}).explain()
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "exerc4b.FO_Paises",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "pais" : {
        "$eq" : "Brasil"
      }
    },
    "winningPlan" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "pais" : {
          "$eq" : "Brasil"
        }
      },
      "direction" : "forward"
    },
    "rejectedPlans" : [ ]
  },
  "serverInfo" : {
    "host" : "ubuntu2004",
    "port" : 32000,
    "version" : "3.6.8",
    "gitVersion" : "8e540c0b6db93ce994cc548f000900bdc740f80a"
  },
  "ok" : 1
}
```

Figura 75 - Coleção 'FO\_Paises' com filtro – comando explain

## 5. Criação de um índice para um atributo que você fez o filtro no item (1.c)

```
db.FO_Paises.createIndex ( { pais : 1 } )
```

```
> db.FO_Paises.createIndex ( { pais : 1 } )
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Figura 76 – Criação de índice para a Coleção 'FO\_Paises'

## 6. Executar a mesma consulta do item (1.c).

```
db.FO_Paises.find({pais : "Brasil"}).explain()
```

```
> db.FO_Paises.find({pais : "Brasil"}).explain()
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "exerc4b.FO_Paises",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "pais" : {
        "$eq" : "Brasil"
      }
    },
    "winningPlan" : {
      "stage" : "FETCH",
      "inputStage" : {
        "stage" : "IXSCAN",
        "keyPattern" : {
          "pais" : 1
        },
        "indexName" : "pais_1",
        "isMultiKey" : false,
        "multiKeyPaths" : {
          "pais" : [ ]
        },
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 2,
        "direction" : "forward",
        "indexBounds" : {
          "pais" : [
            ["Brasil", "Brasil"]
          ]
        }
      },
      "rejectedPlans" : [ ]
    },
    "serverInfo" : {
      "host" : "ubuntu2004",
      "port" : 32000,
      "version" : "3.6.8",
      "gitVersion" : "8e540c0b6db93ce994cc548f000900bdc740f80a"
    },
    "ok" : 1
  }
}
```

Figura 77 – Mesma consulta após a criação do índice

7. Criação da coleção “FO\_Numero1” e inserção de 50.000 documentos em paralelo; numa outra janela executar o utilitário mongostat.

```
mongostat --port 32000
mongo localhost:32000/exerc4b
for (var i = 1; i <= 50000; i++) { db.FO_Numero1.insert({ x : i }) }
```

```
ubuntu@ubuntu2004: ~
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 63.0M 010 110 159b 60.2k 3 Dec 20 21:08:55.485
insert query update delete getmore command dirty used flushes vsize res qrw arw net_in net_out conn time
*0 *0 *0 *0 0 110 0.0% 0.0% 0 958M 63.0M 010 110 156b 59.1k 3 Dec 20 21:08:56.497
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 63.0M 010 110 158b 60.0k 3 Dec 20 21:08:57.492
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 63.0M 010 110 159b 60.1k 3 Dec 20 21:08:58.486
*0 *0 *0 *0 0 110 0.0% 0.0% 0 958M 64.0M 010 110 156b 59.2k 3 Dec 20 21:08:59.494
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 64.0M 010 110 159b 60.1k 3 Dec 20 21:09:00.487
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 64.0M 010 110 158b 59.9k 3 Dec 20 21:09:01.486
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 64.0M 010 110 158b 59.8k 3 Dec 20 21:09:02.485
*0 *0 *0 *0 0 110 0.0% 0.0% 0 958M 64.0M 010 110 156b 59.3k 3 Dec 20 21:09:03.493
*0 *0 *0 *0 0 210 0.0% 0.0% 0 958M 64.0M 010 110 159b 60.1k 3 Dec 20 21:09:04.486
*0 *0 *0 *0 0 110 0.0% 0.0% 0 958M 64.0M 010 110 157b 59.4k 3 Dec 20 21:09:05.491
insert query update delete getmore command dirty used flushes vsize res qrw arw net_in net_out conn time
*0 *0 *0 *0 0 310 0.0% 0.0% 0 958M 64.0M 010 110 213b 60.1k 3 Dec 20 21:09:06.486
1947 1 *0 *0 0 210 0.1% 0.1% 0 958M 64.0M 010 110 312k 147k 3 Dec 20 21:09:07.485
5412 *0 *0 *0 0 310 0.3% 0.3% 0 958M 66.0M 010 110 866k 304k 3 Dec 20 21:09:08.485
5201 *0 *0 *0 0 210 0.6% 0.6% 0 960M 67.0M 010 110 832k 294k 3 Dec 20 21:09:09.485
4257 *0 *0 *0 0 110 0.8% 0.8% 0 960M 68.0M 010 110 681k 251k 3 Dec 20 21:09:10.486
2918 *0 *0 *0 0 210 0.9% 0.9% 0 961M 69.0M 010 110 467k 191k 3 Dec 20 21:09:11.486
3808 *0 *0 *0 0 210 1.1% 1.1% 0 962M 70.0M 010 110 609k 231k 3 Dec 20 21:09:12.485

ubuntu@ubuntu2004: ~
2022-12-20T20:34:27.849-0500 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, star
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** t the
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **
>
>
>
> for (var i = 1; i <= 50000; i++) { db.FO_Numero1.insert({ x : i }) }
```

Figura 78 – Inserção de 50000 documentos e verificação com mongostat

8. Criação da coleção “FO\_Numero2” e inserção de 50.000 documentos em paralelo; numa outra janela executar o utilitário mongotop.

```
mongotop --port 32000
for (var i = 1; i <= 50000; i++) { db.FO_Numero2.insert({ y : i }) }
```

```

ubuntu@ubuntu2004: ~
config.system.sessions      0ms      0ms      0ms
config.transactions         0ms      0ms      0ms
exerc4b.FO_Numero1         0ms      0ms      0ms
exerc4b.FO_Paises          0ms      0ms      0ms
exerc4b.clientes           0ms      0ms      0ms
local.startup_log          0ms      0ms      0ms
local.system.replset       0ms      0ms      0ms

ns      total      read      write      2022-12-20T21:18:54-05:00
exerc4b.FO_Numero2      85ms      0ms      85ms
admin.system.roles      0ms      0ms      0ms
admin.system.version    0ms      0ms      0ms
config.system.sessions  0ms      0ms      0ms
config.transactions     0ms      0ms      0ms
exerc4b.FO_Numero1      0ms      0ms      0ms
exerc4b.FO_Paises       0ms      0ms      0ms
exerc4b.clientes        0ms      0ms      0ms
local.startup_log       0ms      0ms      0ms
local.system.replset    0ms      0ms      0ms

ubuntu@ubuntu2004: ~
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **      Start the ser
h IP
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **      addresses it
ip_all to
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **      bind to all i
t the
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten] **      server with -
2022-12-20T20:34:28.282-0500 I CONTROL [initandlisten]
>
>
>
> for (var i = 1; i <= 50000; i++) { db.FO_Numero1.insert({ x : i }) }
WriteResult({ "nInserted" : 1 })
>
>
> for (var i = 1; i <= 50000; i++) { db.FO_Numero2.insert({ y : i }) }
WriteResult({ "nInserted" : 1 })
> db.FO_Numero2.drop()
true
> for (var i = 1; i <= 50000; i++) { db.FO_Numero2.insert({ y : i }) }

```

Figura 79 – Inserção de 50000 documentos e verificação com mongotop

## 9. Sair do Mongo Shell e fazer um backup da instância inteira do MongoDB.

```

mkdir /home/ubuntu/backups
mongodump --host 127.0.0.1:32000 --out
/home/ubuntu/backups/bkp_completo

```

```

ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/backups
ubuntu@ubuntu2004:~$ mongodump --host 127.0.0.1:32000 --out /home/ubuntu/backups/bkp_completo
2022-12-20T21:28:29.680-0500   writing admin.system.version to
2022-12-20T21:28:29.681-0500   done dumping admin.system.version (1 document)
2022-12-20T21:28:29.681-0500   writing exerc4b.FO_Número2 to
2022-12-20T21:28:29.682-0500   writing exerc4b.FO_Número1 to
2022-12-20T21:28:29.683-0500   writing exerc4b.FO_Paises to
2022-12-20T21:28:29.683-0500   writing exerc4b.clientes to
2022-12-20T21:28:29.684-0500   done dumping exerc4b.clientes (0 documents)
2022-12-20T21:28:29.685-0500   done dumping exerc4b.FO_Paises (4 documents)
2022-12-20T21:28:29.833-0500   done dumping exerc4b.FO_Número1 (50000 documents)
2022-12-20T21:28:29.837-0500   done dumping exerc4b.FO_Número2 (50000 documents)

```

Figura 80 – Criação do diretório de arquivos de backup e comando para backup da instância inteira do MongoDB

```
ls -lR /home/ubuntu/backups/bkp_completo
```

```

ubuntu@ubuntu2004:~$ ls -lR /home/ubuntu/backups/bkp_completo
/home/ubuntu/backups/bkp_completo:
total 8
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 20 21:28 admin
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 20 21:28 exerc4b

/home/ubuntu/backups/bkp_completo/admin:
total 8
-rw-rw-r-- 1 ubuntu ubuntu  59 Dec 20 21:28 system.version.bson
-rw-rw-r-- 1 ubuntu ubuntu 134 Dec 20 21:28 system.version.metadata.json

/home/ubuntu/backups/bkp_completo/exerc4b:
total 3244
-rw-rw-r-- 1 ubuntu ubuntu    0 Dec 20 21:28 clientes.bson
-rw-rw-r-- 1 ubuntu ubuntu  197 Dec 20 21:28 clientes.metadata.json
-rw-rw-r-- 1 ubuntu ubuntu 1650000 Dec 20 21:28 FO_Número1.bson
-rw-rw-r-- 1 ubuntu ubuntu  132 Dec 20 21:28 FO_Número1.metadata.json
-rw-rw-r-- 1 ubuntu ubuntu 1650000 Dec 20 21:28 FO_Número2.bson
-rw-rw-r-- 1 ubuntu ubuntu  132 Dec 20 21:28 FO_Número2.metadata.json
-rw-rw-r-- 1 ubuntu ubuntu   328 Dec 20 21:28 FO_Paises.bson
-rw-rw-r-- 1 ubuntu ubuntu   199 Dec 20 21:28 FO_Paises.metadata.json
ubuntu@ubuntu2004:~$

```

Figura 81 – Arquivos e diretórios criados após o backup

## 10. Matar o processo do MongoDB.

```
ps -ef | grep -i mongo | grep -iv grep
```

```
kill -p 2266
```

```
ps -ef | grep -i mongo | grep -iv grep
```

```

ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu    2266      1   1 20:34 ?        00:00:41 mongod --fork --logpath /home/
--port 32000
ubuntu@ubuntu2004:~$ kill -9 2266
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu@ubuntu2004:~$

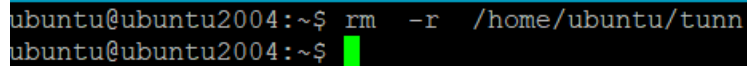
```

Figura 82 – Matando o processo no MongoDB



## 11. Apagar o diretório de dados do MongoDB.

```
rm -r /home/ubuntu/tunn
```

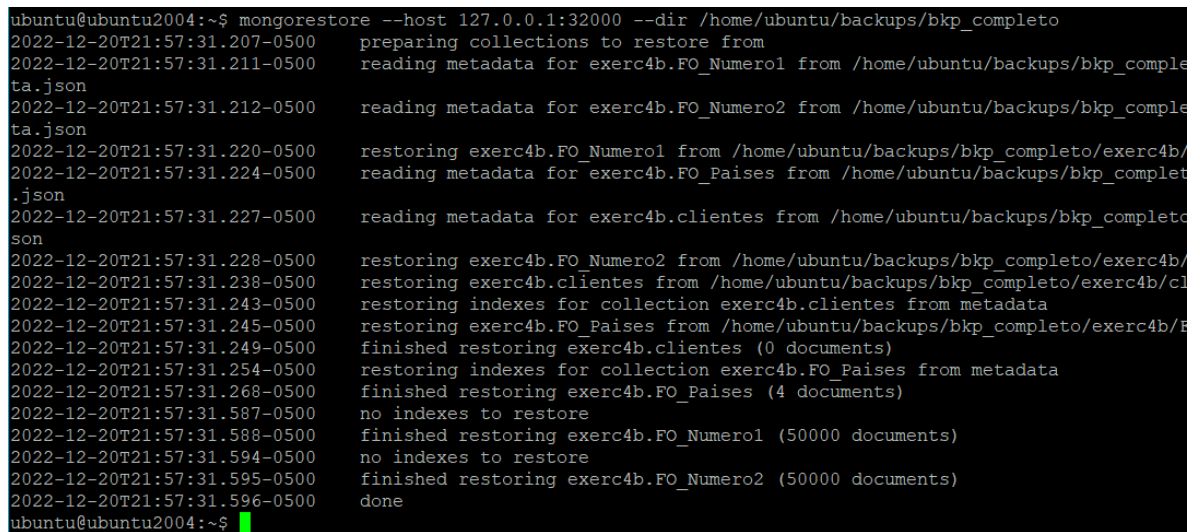


```
ubuntu@ubuntu2004:~$ rm -r /home/ubuntu/tunn
ubuntu@ubuntu2004:~$
```

Figura 83 – Apagando diretório de dados

## 12. Fazer o restore do MongoDB (o caminho tem que ser criado anteriormente).

```
mkdir /home/ubuntu/tunn
mongod --fork --logpath /home/ubuntu/log/mongodb_tunn.log --
dbpath=/home/ubuntu/tunn --port 32000
mongorestore --host 127.0.0.1:32000 --dir
/home/ubuntu/backups/bkp_completo
```

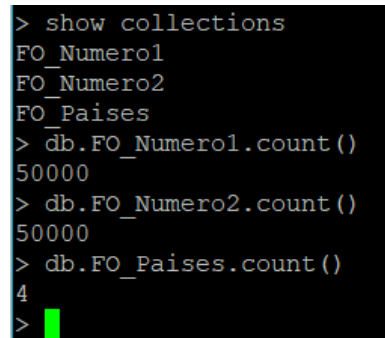


```
ubuntu@ubuntu2004:~$ mongorestore --host 127.0.0.1:32000 --dir /home/ubuntu/backups/bkp_completo
2022-12-20T21:57:31.207-0500   preparing collections to restore from
2022-12-20T21:57:31.211-0500   reading metadata for exerc4b.FO_Numero1 from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Numero1.ta.json
2022-12-20T21:57:31.212-0500   reading metadata for exerc4b.FO_Numero2 from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Numero2.ta.json
2022-12-20T21:57:31.220-0500   restoring exerc4b.FO_Numero1 from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Numero1.ta.json
2022-12-20T21:57:31.224-0500   reading metadata for exerc4b.FO_Paises from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Paises.ta.json
2022-12-20T21:57:31.227-0500   reading metadata for exerc4b.clientes from /home/ubuntu/backups/bkp_completo/exerc4b.clientes.ta.json
2022-12-20T21:57:31.228-0500   restoring exerc4b.FO_Numero2 from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Numero2.ta.json
2022-12-20T21:57:31.238-0500   restoring exerc4b.clientes from /home/ubuntu/backups/bkp_completo/exerc4b.clientes.ta.json
2022-12-20T21:57:31.243-0500   restoring indexes for collection exerc4b.clientes from metadata
2022-12-20T21:57:31.245-0500   restoring exerc4b.FO_Paises from /home/ubuntu/backups/bkp_completo/exerc4b.FO_Paises.ta.json
2022-12-20T21:57:31.249-0500   finished restoring exerc4b.clientes (0 documents)
2022-12-20T21:57:31.254-0500   restoring indexes for collection exerc4b.FO_Paises from metadata
2022-12-20T21:57:31.268-0500   finished restoring exerc4b.FO_Paises (4 documents)
2022-12-20T21:57:31.587-0500   no indexes to restore
2022-12-20T21:57:31.588-0500   finished restoring exerc4b.FO_Numero1 (50000 documents)
2022-12-20T21:57:31.594-0500   no indexes to restore
2022-12-20T21:57:31.595-0500   finished restoring exerc4b.FO_Numero2 (50000 documents)
2022-12-20T21:57:31.596-0500   done
ubuntu@ubuntu2004:~$
```

Figura 84 – Restore do MongoDB

13. Acessar a instância e ir para o database exerc4b. Verificar o número de documentos por coleção.

```
mongo localhost:32000/exerc4b  
show collections  
db.FO_Numero1.count()  
db.FO_Numero2.count()  
db.FO_Paises.count()
```

A terminal window with a black background and white text. It shows the execution of several MongoDB commands. The first command is 'show collections', which lists three collections: 'FO\_Numero1', 'FO\_Numero2', and 'FO\_Paises'. The next three commands are 'db.FO\_Numero1.count()', 'db.FO\_Numero2.count()', and 'db.FO\_Paises.count()', which return the document counts for each collection: 50000, 50000, and 4 respectively. The prompt '>' is visible at the end of each line.

```
> show collections  
FO_Numero1  
FO_Numero2  
FO_Paises  
> db.FO_Numero1.count()  
50000  
> db.FO_Numero2.count()  
50000  
> db.FO_Paises.count()  
4  
>
```

Figura 85 – Número de documentos por coleção após Restore do MongoDB.