Projeto Final da Disciplina Infraestrutura MongoDB – Pós Graduação MIT em Engenharia de dados: Big Data

Prof.: Carlos Eduardo Gertners

Conceitos

1.

Diferenças:

Modelo de dados

SQL: Utiliza um modelo de dados relacional, onde os dados são organizados em tabelas com esquemas predefinidos.

NoSQL: Pode ter modelos de dados variados, incluindo documentos, chave-valor, grafos, famílias de colunas e etc.

Escalabilidade

No SQL a escalabilidade é vertical, ou seja, para aumentar o desempenho você tem que melhorar o processamento, memória e etc, daquela máquina.

No noSQL para o aumento do desempenho, basta vc adionar mais n[os, ou seja, adicionar outra máquina na mesma configuração da outra.

Consistência vs. Desempenho:

SQL: Oferece garantia de consistência forte devido ao modelo ACID.

NoSQL: Muitas vezes sacrifica a consistência em prol do desempenho, seguindo o modelo CAP (Consistência, Disponibilidade, Tolerância a Partições).

Semelhanças:

Armazenamento de Dados:

Ambos os tipos de bases de dados são projetados para armazenar dados de forma persistente. A principal diferença está na maneira como esses dados são modelados e acessados.

Manipulação de Dados:

Tanto em SQL quanto em NoSQL, você pode realizar operações CRUD (Create, Read, Update, Delete) para manipular os dados armazenados.

Índices:

Ambos suportam índices para melhorar o desempenho de consultas. Os índices são usados para acelerar a recuperação de dados em consulta.

2.

Não existe um modelo melhor ou pior, e sim irá depender do tipo do negócio para analisarmos qual das bases fará mais sentido.

Se for algo com diferentes tipos de dados, o noSQL se encaixa melhor; se for algo que requer uma consistência alta, o modelo SQL entregará mais.

3.

NoSQL

Pra extração de dados oriundo de streaming, rede social, pela diversidade de dado optaria por um modelo noSQL.

SQL

Para criação por exemplo de algum sistema bancário, usaria o SQL por sua forte característica de consistência.

4.

Existe uma série de tipos de banco de dados noSQL atualmente mercado, mas as principais categorias, são:

- Documentos → MongoDB
- Colunar → Cassandra
- Chave-valor → Redis
- Grafos → Neo4j

Documentos

Os dados são documentos, flexíveis e semiestruturados, permitindo alteração conforme a necessidade. Os documentos são estruturas de chave e valor.

Colunar

Faz o armazenamento dos dados em linhas e permite consulta com base em colunas específicas.

Chave-valor

Os dados são armazenados na forma de chave valor como tabelas hash. Não há tabelas e não há uma maneira definida para relacionar os dados.

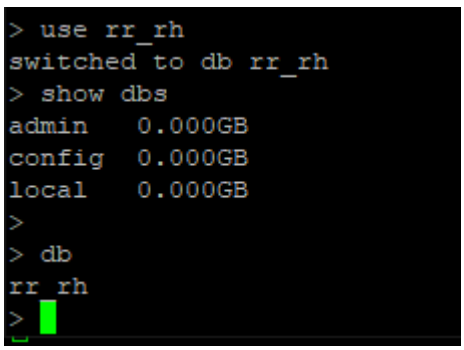
Grafos

O grafo é uma estrutura composta por uma série de nós que podem ser conectados por arestas. Redes sociais e mapas de cidades, utilizam grafo.

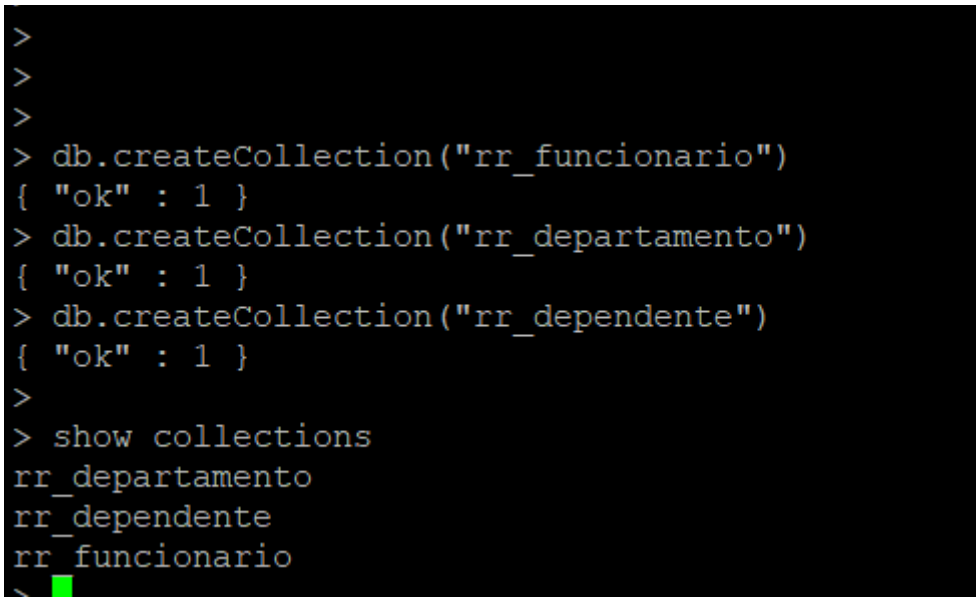
Fonte: Curso de Big Data IFMG 2022 e XP Educação.

Prática

Criação do Data Base.



Criação das collections



Inserindo documentos da collection rr_departamento.

```
> db.rr_departamento.insertMany([
...     {
...         cod: 1,
...         nome: "Pesquisa",
...         localizacao: "São Paulo"
...     },
...     {
...         cod: 2,
...         nome: "Engenharia",
...         localizacao: "Rio de Janeiro",
...         bairro: "Centro"
...     },
...     {
...         cod: 3,
...         nome: "Admininstracao",
...         localizacao: "Belo Horizonte"
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("658112bd76df72522fa4ec03"),
                ObjectId("658112bd76df72522fa4ec04"),
                ObjectId("658112bd76df72522fa4ec05")
        ]
}
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rr_rh   0.000GB
> db.rr_departamento.find()
{ "_id" : ObjectId("658112bd76df72522fa4ec03"), "cod" : 1, "nome" : "Pesquisa", "localizacao" : "São Paulo" }
{ "_id" : ObjectId("658112bd76df72522fa4ec04"), "cod" : 2, "nome" : "Engenharia", "localizacao" : "Rio de Janeiro", "bairro" : "Centro" }
{ "_id" : ObjectId("658112bd76df72522fa4ec05"), "cod" : 3, "nome" : "Admininstracao", "localizacao" : "Belo Horizonte" }
>
```

Inserindo documentos da collection rr_funcionário.

```
> db['rr_funcionario'].insertMany([
...     {
...         "cod": 101,
...         "nome": "Rafael",
...         "cargo": "Engenheiro",
...         "salario": 10000
...     },
...     {
...         cod: 102,
...         nome: "Fernanda",
...         cargo: "Analista",
...         salario: 8000,
...         Endereco: " Rua João Gomes 10"
...     },
...     {
...         cod: 103,
...         nome: "Roberta",
...         cargo: "Administradora",
...         salario: 7500
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6581170476df72522fa4ec09"),
                ObjectId("6581170476df72522fa4ec0a"),
                ObjectId("6581170476df72522fa4ec0b")
        ]
}
```

```
> db.rr_funcionario.find()
{ "_id" : ObjectId("6581170476df72522fa4ec09"), "cod" : 101, "nome" : "Rafael", "cargo" : "Engenheiro", "salario" : 10000 }
{ "_id" : ObjectId("6581170476df72522fa4ec0a"), "cod" : 102, "nome" : "Fernanda", "cargo" : "Analista", "salario" : 8000, "Endereco" : " Rua João Gomes 10" }
{ "_id" : ObjectId("6581170476df72522fa4ec0b"), "cod" : 103, "nome" : "Roberta", "cargo" : "Administradora", "salario" : 7500 }
>
```

Inserindo documentos na collection rr_dependente.

```
> db.rr_dependente.insertMany([
...     {
...         cod: 201,
...         nome: "Ana",
...         relacao: "Filha"
...     },
...     {
...         cod: 202,
...         nome: "Pedro",
...         relacao: "Filho",
...         maior_idade: "Nao"
...     },
...     {
...         cod: 203,
...         nome: "Mariana",
...         relacao: "Filha"
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65811aae76df72522fa4ec0f"),
                ObjectId("65811aae76df72522fa4ec10"),
                ObjectId("65811aae76df72522fa4ec11")
        ]
}
> db.rr_dependente.find()
{ "_id" : ObjectId("65811aae76df72522fa4ec0f"), "cod" : 201, "nome" : "Ana", "relacao" : "Filha" }
{ "_id" : ObjectId("65811aae76df72522fa4ec10"), "cod" : 202, "nome" : "Pedro", "relacao" : "Filho", "maior_idade" : "Nao" }
{ "_id" : ObjectId("65811aae76df72522fa4ec11"), "cod" : 203, "nome" : "Mariana", "relacao" : "Filha" }
>
```

Fazendo um filtro pela descrição da collection rr_departamento.

```
>
> db.rr_departamento.find({
...     nome: "Engenharia"
... })
{ "_id" : ObjectId("6581195276df72522fa4ec0d"), "cod" : 2, "nome" : "Engenharia", "localizacao" : "Rio de Janeiro", "bairro" : "Centro" }
>
>
```

Fazendo um filtro na collection rr_funcionario que recebem mais que 7900.

```
>
> db['rr_funcionario'].find({
...     "salario": { $gt: 7900 }
... })
{ "_id" : ObjectId("6581170476df72522fa4ec09"), "cod" : 101, "nome" : "Rafael", "cargo" : "Engenheiro", "salario" : 10000 }
{ "_id" : ObjectId("6581170476df72522fa4ec0a"), "cod" : 102, "nome" : "Fernanda", "cargo" : "Analista", "salario" : 8000, "Endereco" : " Rua João Gomes 10" }
>
>
>
>
```

Aplicando o distinct na collection rr_dependente para retornar todos os valores diferentes da chave “nome”.

```
>
>
>
> db.rr_dependente.distinct("nome")
[ "Ana", "Pedro", "Mariana" ]
>
>
>
```

Update na coleção rr_departamento

```
> db.rr_departamento.update(
...     { cod: 2 },
...     {
...       $set: {
...         localizacao: "Santa Catarina",
...         extra_atributo: "Florianopolis"
...       }
...     }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.rr_departamento.find()
{ "_id" : ObjectId("65822f3230f4e827b24141d2"), "cod" : 1, "nome" : "Pesquisa", "localiza
cao" : "São Paulo" }
{ "_id" : ObjectId("65822f3230f4e827b24141d3"), "cod" : 2, "nome" : "Engenharia", "locali
zacao" : "Santa Catarina", "bairro" : "Centro", "extra_atributo" : "Florianopolis" }
{ "_id" : ObjectId("65822f3230f4e827b24141d4"), "cod" : 3, "nome" : "Admininstracao", "lo
calizacao" : "Belo Horizonte" }
>
```

Remoção da coleção rr_dependente

```
>
> db.rr_dependente.drop()
true
>
>
> show collections
rr_departamento
rr_funcionario
>
>
```

Modelagem

Cardinalidade 1-N

```
> db
rr_modelo
>
>
> db.rr_colla.insertMany([
...     {
...         id: 1,
...         livro: "Mindset",
...         autora: "Carol Dweck"
...     },
...     {
...         id: 2,
...         livro: "Trading in the Zone",
...         autor: "Mark Douglas"
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65823c3ee835f0f91d3986c9"),
                ObjectId("65823c3ee835f0f91d3986ca")
        ]
}
```

```
> db.rr_col2a.insertMany([
...     {
...         id: 104,
...         colla_id: 1,
...         detalhe: "Para Psicologos"
...     },
...     {
...         id: 105,
...         colla_id: 1,
...         detalhe: "Para Pscicologos"
...     },
...     {
...         cod: 106,
...         colla_id: 2,
...         detalhe: "Para Tarders"
...     },
...     {
...         cod: 107,
...         colla_id: 2,
...         detalhe: "Para Traders"
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65823d4ae835f0f91d3986cf"),
                ObjectId("65823d4ae835f0f91d3986d0"),
                ObjectId("65823d4ae835f0f91d3986d1"),
                ObjectId("65823d4ae835f0f91d3986d2")
        ]
}
```

Cardinalidade N-N por referência

Criação da collection cliente

```
> db.rr_clientes.insertMany([
...     {
...         id: 1,
...         nome: "João",
...         endereco: "Rua 1920"
...     },
...     {
...         id: 2,
...         nome: "Rafael",
...         endereco: "Rua 2028"
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65824776e835f0f91d3986d6"),
                ObjectId("65824776e835f0f91d3986d7")
        ]
}
```

Criação da collection pedidos

```
> db.rr_pedidos.insertMany([
...     {
...         cod_prod: 101,
...         cliente_id: 1,
...             },
...     {
...         id: 102,
...         cliente_id: 2,
...            }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65824788e835f0f91d3986d8"),
                ObjectId("65824788e835f0f91d3986d9")
        ]
}
```

Criação da collection itens

```
> db.rr_itens.insertMany([
...     {
...         id: 1001,
...         descricao: "caneta",
...         preco: 2.99
...     },
...     {
...         id: 1002,
...         descricao: "lapis",
...         preco: 1.99
...     },
...     {
...         id: 1003,
...         descricao: "borracha",
...         preco: 0.99
...     }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("658246d8e835f0f91d3986d3"),
                ObjectId("658246d8e835f0f91d3986d4"),
                ObjectId("658246d8e835f0f91d3986d5")
        ]
}
```

Referência de itens nos pedidos dos clientes.

```
> db.rr_pedidos.find()
{ "_id" : ObjectId("65824f3ee835f0f91d3986dc"), "id" : 101, "cliente_id" : 1
 }
{ "_id" : ObjectId("65824f3ee835f0f91d3986dd"), "id" : 102, "cliente_id" : 2
 }
>
>
> db.rr_pedidos.update(
...     { id: 101 },
...     { $addToSet: { "itens": { $each: [1001, 1002] } } }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
>
>
> db.rr_pedidos.find()
{ "_id" : ObjectId("65824f3ee835f0f91d3986dc"), "id" : 101, "cliente_id" : 1
, "itens" : [ 1001, 1002 ] }
{ "_id" : ObjectId("65824f3ee835f0f91d3986dd"), "id" : 102, "cliente_id" : 2
 }
>
```

```
> db.rr_pedidos.update(
...     { id: 102 },
...     { $addToSet: { "itens": { $each: [1002, 1003] } } }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
>
> db.rr_pedidos.find()
{ "_id" : ObjectId("65824f3ee835f0f91d3986dc"), "id" : 101, "cliente_id" : 1
, "itens" : [ 1001, 1002 ] }
{ "_id" : ObjectId("65824f3ee835f0f91d3986dd"), "id" : 102, "cliente_id" : 2
, "itens" : [ 1002, 1003 ] }
>
>
```

Índice

ubuntu@ubuntu2004: ~

```
> db.rr_indexar1.createIndex({ "atributo1": 1 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.rr_indexar1.createIndex({ "atributo2": -1 }, { unique: true })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
> db.rr_indexar1.createIndex({ "atributo3": 1, "atributo4": -1 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 3,
        "numIndexesAfter" : 4,
        "ok" : 1
}
> db.rr_indexar1.createIndex({ "atributo3": 1 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 4,
        "numIndexesAfter" : 5,
        "ok" : 1
}
> db.rr_indexar1.createIndex({ "atributo5": 1 }, { sparse: true })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 5,
        "numIndexesAfter" : 6,
        "ok" : 1
}
> db.rr_indexar1.createIndex({ "atributo6.subatributo": 1 }, { expireAfterSecon
ds: 20 })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 6,
        "numIndexesAfter" : 7,
        "ok" : 1
}
```

```
> db.rr_indexar1.insertMany([
...     {
...       "atributo1": "valor1",
...       "atributo2": 42,
...       "atributo3": ["array1", "array2"],
...       "atributo4": true,
...       "atributo5": "texto",
...       "atributo6": 3.14,
...       "atributo7": {
...         "subatributo": "subvalor"
...       }
...     },
...     {
...       "atributo1": "valor2",
...       "atributo2": 25,
...       "atributo3": ["array3", "array4"],
...       "atributo4": false,
...       "atributo5": "outro texto",
...       "atributo6": 2.718,
...       "atributo7": {
...         "subatributo": "outro subvalor"
...       }
...     },
...     {
...       "atributo1": "valor3",
...       "atributo2": 30,
...       "atributo3": ["array5", "array6"],
...       "atributo4": true,
...       "atributo5": "mais um texto",
...       "atributo6": 1.618,
...       "atributo7": {
...         "subatributo": "mais um subvalor"
...       }
...     },
...     {
...       "atributo1": "valor4",
...       "atributo2": 18,
...       "atributo3": ["array7", "array8"],
...       "atributo4": false,
...       "atributo5": "último texto",
...       "atributo6": 2.0,
...       "atributo7": {
...         "subatributo": "último subvalor"
...       }
```

```
...       atributo7 : {
...         "subatributo": "último subvalor"
...       }
...     }
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6583b150879cb0afd582f7e2"),
                ObjectId("6583b150879cb0afd582f7e3"),
                ObjectId("6583b150879cb0afd582f7e4"),
                ObjectId("6583b150879cb0afd582f7e5")
        ]
}
```

Índice Textual

```
> db.createCollection("rr_indexar2")
{ "ok" : 1 }
>
>
> db.rr_indexar2.createIndex({ atributo1: "text", atributo2: "text", atri
buto3: "text", atributo4: "text" })
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

```
> db.rr_indexar2.insertMany([
...   {
...     atributo1: "Um dia ficarei fera em MongoDB",
...     atributo2: "Está sendo um baita desafio essa Pós",
...     atributo3: "Mas se Deus quiser chegarei no meu objetivo",
...     atributo4: "Fazer minha transição para a área de Engenahria de Da
dos"
...   },
...   {
...     atributo1: "MongoDB",
...     atributo2: "Cassandra",
...     atributo3: "Spark",
...     atributo4: "Hadoop"
...   },
...   {
...     atributo1: "Data Lake",
...     atributo2: "Warehouse",
...     atributo3: "Banco Relaciona",
...     atributo4: "Banco nao Relacional"
...   },
...   {
...     atributo1: "VAMO QUE VAMO",
...     atributo2: "FÉ",
...     atributo3: "Esforço",
...     atributo4: "Não desistir"
...   }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6583b51d879cb0afd582f7e6"),
                ObjectId("6583b51d879cb0afd582f7e7"),
                ObjectId("6583b51d879cb0afd582f7e8"),
                ObjectId("6583b51d879cb0afd582f7e9")
        ]
```

```
> db.rr_Venda.aggregate([
...   {
...     $group: {
...       _id: "$UF_Venda",
...       totalDoc: { $sum: 1 }
...     }
...   }
... ])
{ "_id" : "PR", "totalDoc" : 4 }
{ "_id" : "SC", "totalDoc" : 4 }
{ "_id" : "SP", "totalDoc" : 4 }
{ "_id" : "RJ", "totalDoc" : 4 }
>
>
```

```
> db.rr_Venda.aggregate([
...   {
...     $group: {
...       _id: "$UF_Venda",
...       Total_Venda: { $sum: "$Valor_Venda" }
...     }
...   }
... ])
{ "_id" : "PR", "Total_Venda" : 2270 }
{ "_id" : "SC", "Total_Venda" : 2350 }
{ "_id" : "SP", "Total_Venda" : 2310 }
{ "_id" : "RJ", "Total_Venda" : 1950 }
>
```

```
> db.rr_Venda.aggregate([
...   {
...     $group: {
...       _id: "$UF_Venda",
...       mediaVendas: { $avg: "$Valor_Venda" }
...     }
...   }])
{ "_id" : "PR", "mediaVendas" : 567.5 }
{ "_id" : "SC", "mediaVendas" : 587.5 }
{ "_id" : "SP", "mediaVendas" : 577.5 }
{ "_id" : "RJ", "mediaVendas" : 487.5 }
>
```

```
> db.rr_Venda.aggregate([
...   {
...     $group: {
...       _id: "$UF_Venda",
...       MaiorValorVenda: { $max: "$Valor_Venda" }
...     }
...   }])
{ "_id" : "PR", "MaiorValorVenda" : 900 }
{ "_id" : "SC", "MaiorValorVenda" : 1300 }
{ "_id" : "SP", "MaiorValorVenda" : 1500 }
{ "_id" : "RJ", "MaiorValorVenda" : 1000 }
>
```

```
> db.rr_Venda.aggregate([
...    {
...      $group: {
...        _id: "$UF_Venda",
...        MenorValorVenda: { $min: "$Valor_Venda" }
...      }
...    }])
{ "_id" : "PR", "MenorValorVenda" : 180 }
{ "_id" : "SC", "MenorValorVenda" : 140 }
{ "_id" : "SP", "MenorValorVenda" : 120 }
{ "_id" : "RJ", "MenorValorVenda" : 150 }
>
>
```

Replicação

Verificando se as instâncias estão rodando.

```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu     75375   75272  0 18:25 pts/2    00:00:13 mongod --replSet rscad --logpath /home/ubuntu/log/mongodb_1.log --dbpath=/home/ubuntu/replica/node1 --port 40001
ubuntu     75482   75456  1 19:00 pts/1    00:00:02 mongod --replSet rscad --logpath /home/ubuntu/log/mongodb_2.log --dbpath=/home/ubuntu/replica/node2 --port 40002
ubuntu     75599   75585  1 19:02 pts/3    00:00:01 mongod --replSet rscad --logpath /home/ubuntu/log/mongodb_3.log --dbpath=/home/ubuntu/replica/node3 --port 40003
ubuntu     75693   75680  3 19:03 pts/4    00:00:01 mongod --replSet rscad --logpath /home/ubuntu/log/mongodb_4.log --dbpath=/home/ubuntu/replica/arbiter --port 40004
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
```

Conectando ao nó 1 para, criando o db e iniciando o replica set.

```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongo localhost:40001
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:40001/test
Implicit session: session { "id" : UUID("bee9d098-9d04-45f5-8787-f88e44a696bd") }
MongoDB server version: 3.6.8
Server has startup warnings:
2023-12-21T18:25:13.230-0500 I STORAGE  [initandlisten]
2023-12-21T18:25:13.230-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2023-12-21T18:25:13.230-0500 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten]
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten]
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2023-12-21T18:25:14.082-0500 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2023-12-21T18:25:14.083-0500 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2023-12-21T18:25:14.083-0500 I CONTROL  [initandlisten]
>
> use rr_rsposmit
switched to db rr_rsposmit
>
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "localhost:40001",
        "ok" : 1,
        "operationTime" : Timestamp(1703203792, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703203792, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rscad:OTHER>
rscad:PRIMARY>
```

.

Adicionando os nós ao replica set.

```
                        "keyId" : NumberLong(0)
                }
        }
}
rscad:PRIMARY> rs.add('localhost:40002')
{
        "ok" : 1,
        "operationTime" : Timestamp(1703204059, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703204059, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rscad:PRIMARY> rs.add('localhost:40003')
{
        "ok" : 1,
        "operationTime" : Timestamp(1703204078, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703204078, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rscad:PRIMARY> rs.addArb('localhost:40004')
{
        "ok" : 1,
        "operationTime" : Timestamp(1703204099, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703204099, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rscad:PRIMARY>
```

Nó primário

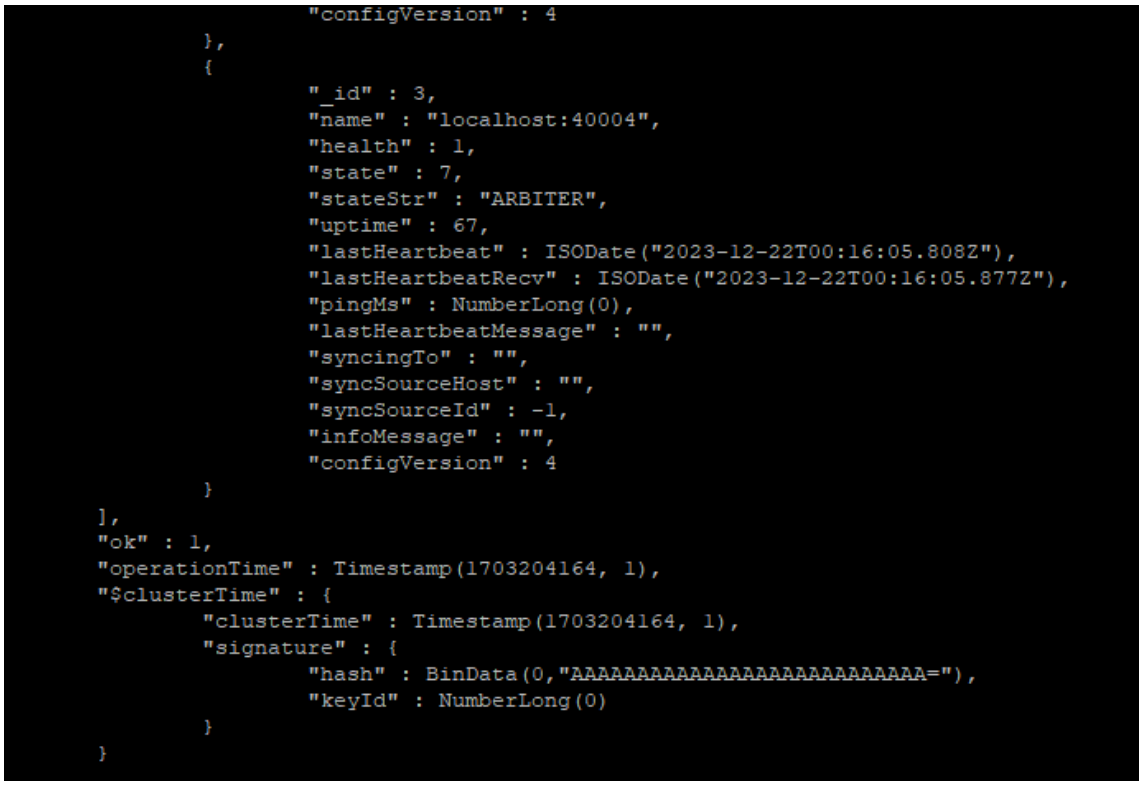
```
        }
}
rscad:PRIMARY>
rscad:PRIMARY> rs.status()
{
        "set" : "rscad",
        "date" : ISODate("2023-12-22T00:16:06.893Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:40001",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 3053,
                        "optime" : {
                                "ts" : Timestamp(1703204164, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2023-12-22T00:16:04Z"),
```

Nós secundários

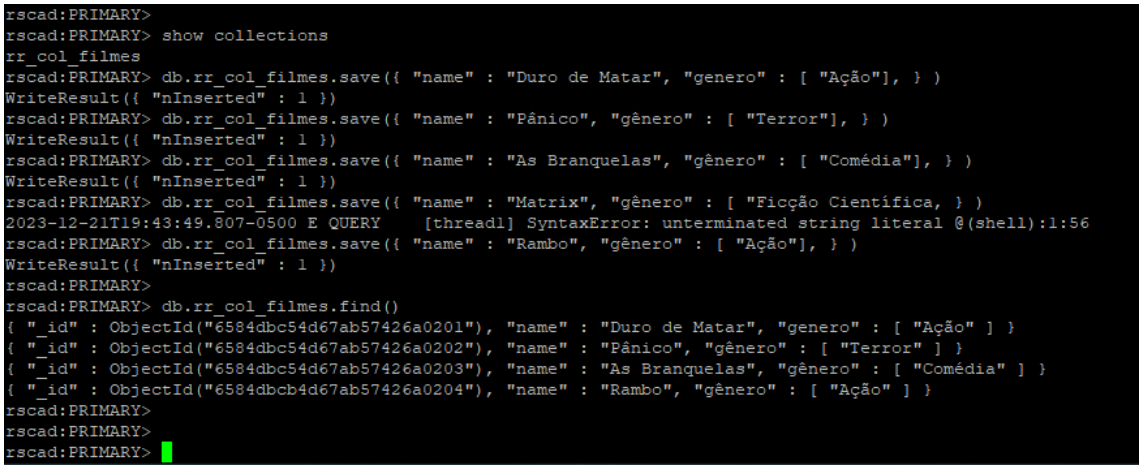
ubuntu@ubuntu2004: ~


```
        },
        {
                "_id" : 1,
                "name" : "localhost:40002",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 107,
                "optime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "optimeDate" : ISODate("2023-12-22T00:16:04Z"),
                "optimeDurableDate" : ISODate("2023-12-22T00:16:04Z"),
                "lastHeartbeat" : ISODate("2023-12-22T00:16:05.811Z"),
                "lastHeartbeatRecv" : ISODate("2023-12-22T00:16:04.926Z"),
                "pingMs" : NumberLong(0),
                "lastHeartbeatMessage" : "",
                "syncingTo" : "localhost:40001",
                "syncSourceHost" : "localhost:40001",
                "syncSourceId" : 0,
                "infoMessage" : "",
                "configVersion" : 4
        },
        {
                "_id" : 2,
                "name" : "localhost:40003",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 88,
                "optime" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1703204164, 1),
                        "t" : NumberLong(1)
                },
                "optimeDate" : ISODate("2023-12-22T00:16:04Z"),
```

Nó Árbitro



Inserindo 5 documentos na coleção rr_col_filmes no nó primário.



Acessando o nó secundário, ativando-o como slave do replica set e consultando a coleção rr_col_filmes.

```
rscad:SECONDARY> use rr_rsposmit
switched to db rr_rsposmit
rscad:SECONDARY>
rscad:SECONDARY> rs.slaveOk()
rscad:SECONDARY>
rscad:SECONDARY> db.rr_col_filmes.find()
{ "_id" : ObjectId("6584dbc54d67ab57426a0201"), "name" : "Duro de Matar", "genero" : [ "Ação" ] }
{ "_id" : ObjectId("6584dbc54d67ab57426a0202"), "name" : "Pânico", "gênero" : [ "Terror" ] }
{ "_id" : ObjectId("6584dbc54d67ab57426a0203"), "name" : "As Branquelas", "gênero" : [ "Comédia" ] }
{ "_id" : ObjectId("6584dbcb4d67ab57426a0204"), "name" : "Rambo", "gênero" : [ "Ação" ] }
rscad:SECONDARY>
rscad:SECONDARY>
rscad:SECONDARY>
```

Particionamento

Criando o ecossistema com os ConfigServers, os Shards e o MongoS.

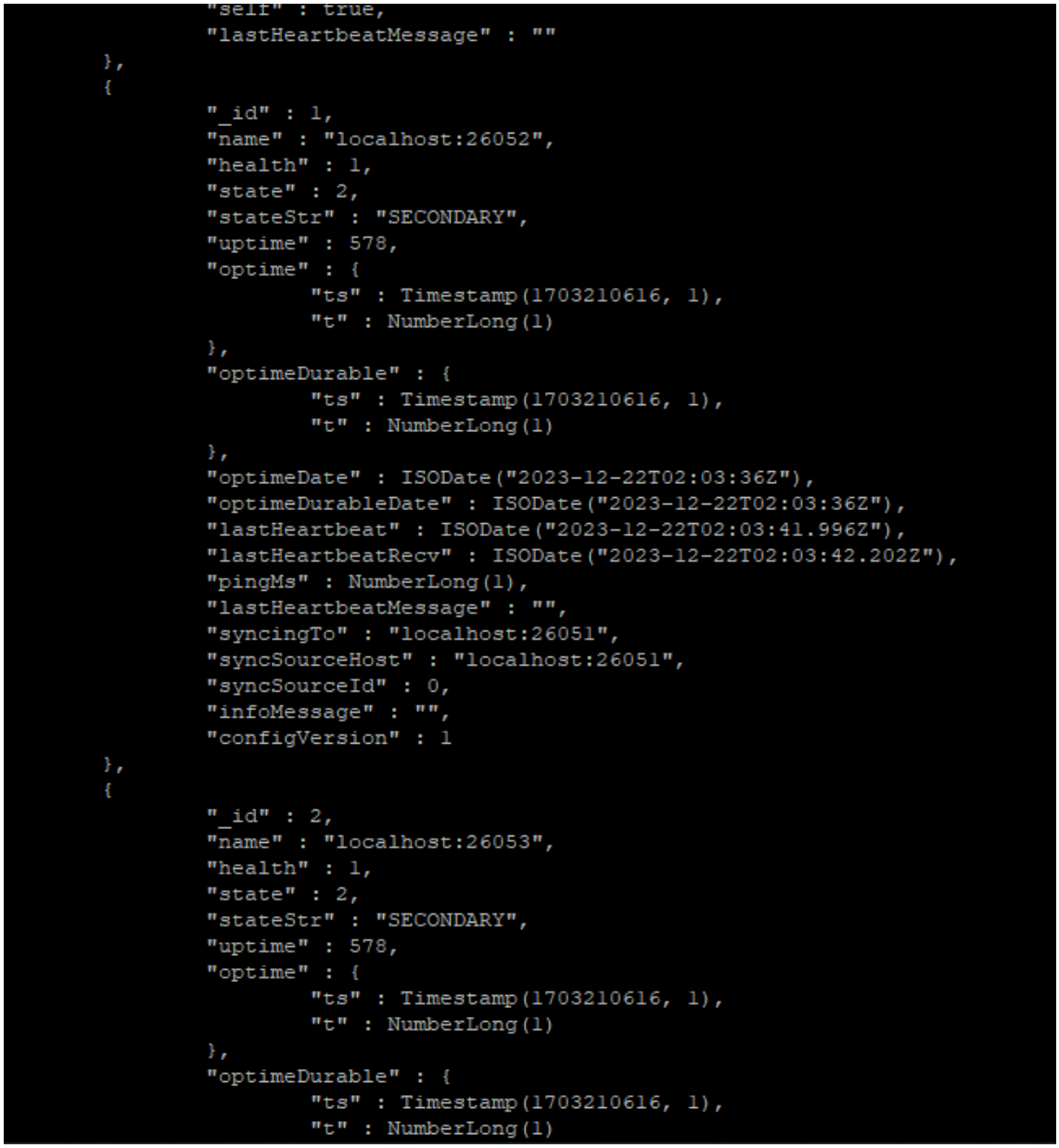
Criando os ConfigServers

```
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --dbpath=/home/ubuntu/Shard/cfg1 --port 26051
about to fork child process, waiting until server is ready for connections.
forked process: 76423
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg2.log --dbpath=/home/ubuntu/Shard/cfg2 --port 26052
about to fork child process, waiting until server is ready for connections.
forked process: 76456
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg3.log --dbpath=/home/ubuntu/Shard/cfg3 --port 26053
about to fork child process, waiting until server is ready for connections.
forked process: 76489
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      76423       1  2 20:49 ?        00:00:02 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --dbpath=/home/ubuntu/Shard
/cfg1 --port 26051
ubuntu      76456       1  3 20:50 ?        00:00:01 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg2.log --dbpath=/home/ubuntu/Shard
/cfg2 --port 26052
ubuntu      76489       1 18 20:51 ?        00:00:01 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg3.log --dbpath=/home/ubuntu/Shard
/cfg3 --port 26053
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
```

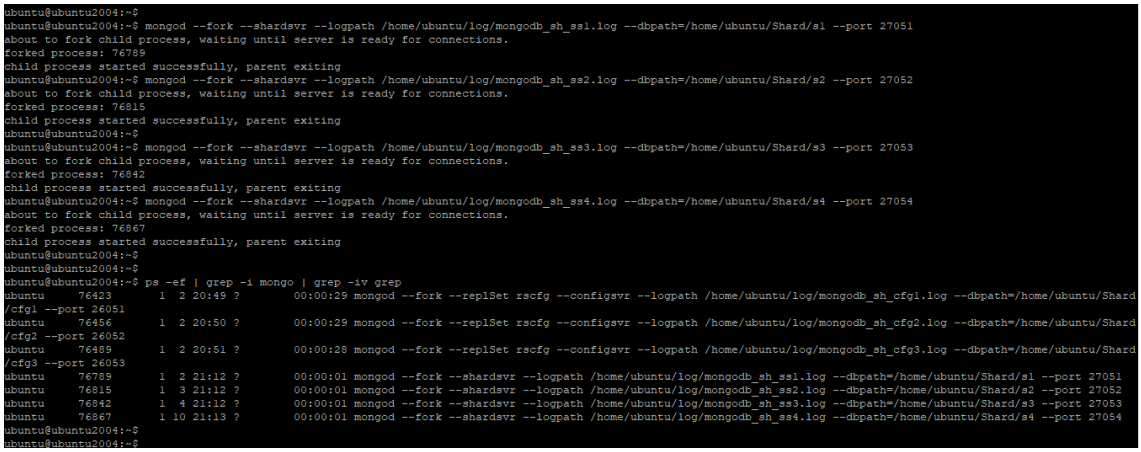
```
ubuntu@ubuntu2004:~$ mongo --port 26051
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:26051/
Implicit session: session { "id" : UUID("0e725b5f-fa06-4e92-b408-670bc682c6c2") }
MongoDB server version: 3.6.8
Server has startup warnings:
2023-12-21T20:49:53.188-0500 I STORAGE  [initandlisten]
2023-12-21T20:49:53.188-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2023-12-21T20:49:53.188-0500 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten]
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten]
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2023-12-21T20:49:54.068-0500 I CONTROL  [initandlisten]
>
>
> rs.initiate(
...   {
...     _id: "rscfg",
...     configsvr: true,
...     members: [
...       { _id : 0, host : "localhost:26051" },
...       { _id : 1, host : "localhost:26052" },
...       { _id : 2, host : "localhost:26053" }
...     ]
...   }
... )
{
        "ok" : 1,
        "operationTime" : Timestamp(1703210044, 1),
        "$gleStats" : {
                "lastOpTime" : Timestamp(1703210044, 1),
                "electionId" : ObjectId("000000000000000000000000")
        },
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703210044, 1),
                "signature" : {
```

Verificando o status dos ConfigServers e seus respectivos nós primário e os secundários.

```
rscfg:PRIMARY>
rscfg:PRIMARY> rs.status()
{
        "set" : "rscfg",
        "date" : ISODate("2023-12-22T02:03:42.629Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "configsvr" : true,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1703210616, 1),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1703210616, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1703210616, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1703210616, 1),
                        "t" : NumberLong(1)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:26051",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 829,
                        "optime" : {
                                "ts" : Timestamp(1703210616, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2023-12-22T02:03:36Z"),
```



Aqui subimos quatro Shards e posteriormente podemos verificar os processos em aberto dos três ConfigServers e dos quatro Shards.



Subindo o MongoS e posteriormente verificando os processos ativos.

```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongos --fork --configdb "rscfg/localhost:26051,localhost:26052,localhost:26053" --logpath /home/ubuntu/log/mongos_sh.log --port 28000
about to fork child process, waiting until server is ready for connections.
forked process: 76958
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu      76423       1  1 20:49 ?        00:00:45 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --dbpath=/home/ubuntu/Shard
/cfg1 --port 26051
ubuntu      76456       1  2 20:50 ?        00:00:45 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg2.log --dbpath=/home/ubuntu/Shard
/cfg2 --port 26052
ubuntu      76489       1  2 20:51 ?        00:00:44 mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg3.log --dbpath=/home/ubuntu/Shard
/cfg3 --port 26053
ubuntu      76789       1  1 21:12 ?        00:00:11 mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss1.log --dbpath=/home/ubuntu/Shard/s1 --port 27051
ubuntu      76815       1  1 21:12 ?        00:00:10 mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss2.log --dbpath=/home/ubuntu/Shard/s2 --port 27052
ubuntu      76842       1  1 21:12 ?        00:00:10 mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss3.log --dbpath=/home/ubuntu/Shard/s3 --port 27053
ubuntu      76867       1  1 21:13 ?        00:00:10 mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss4.log --dbpath=/home/ubuntu/Shard/s4 --port 27054
ubuntu      76958       1  1 21:27 ?        00:00:00 mongos --fork --configdb rscfg/localhost:26051,localhost:26052,localhost:26053 --logpath /home/ubuntu/log/mongos_sh.
log --port 28000
ubuntu@ubuntu2004:~$
```

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("6584ec48932d212f73c59949")
  }
  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
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard0000       1
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : shard0000 Timestamp(1, 0)
```

Habilitando o particionamento no banco de dados test.

```
mongos>
mongos>
mongos> sh.enableSharding ("test")
{
        "ok" : 1,
        "operationTime" : Timestamp(1703216319, 5),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703216319, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos>
```

Conectando ao banco de dados test e inserindo a coleção clientes.

```
mongos> use test
switched to db test
mongos>
mongos>
mongos> show collections
mongos>
mongos>
mongos> for (var i=0; i < 1000; i++ ) { db.clientes.insert ( { x:i, y:3, uf :"RJ" } ) }
WriteResult({ "nInserted" : 1 })
mongos>
mongos> show collections
clientes
mongos>
```

Particionando a coleção clientes do banco de dados test.

```
mongos>
mongos> sh.shardCollection ("test.clientes", {_id:1}, true)
{
        "collectionsharded" : "test.clientes",
        "collectionUUID" : UUID("2361a87a-d32c-46f8-b5b6-1449f1be1178"),
        "ok" : 1,
        "operationTime" : Timestamp(1703216384, 10),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1703216384, 10),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos>
mongos>
```

Abaixo podemos verificar os estados dos shards, que o balanceamento está ativo e que os dados estão todos no shard01.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("6584ec48932d212f73c59949")
  }
  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
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shard0000       1
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : shard0000 Timestamp(1, 0)
        {  "_id" : "test",  "primary" : "shard0001",  "partitioned" : true }
                test.clientes
                        shard key: { "_id" : 1 }
                        unique: true
                        balancing: true
                        chunks:
                                shard0001       1
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : shard0001 Timestamp(1, 0)

mongos>
```

Abaixo temos o número de chunks (faixa de valores), apenas um.

E podemos perceber pela distribuição que o shard01 contém 100% dos dados.

```
mongos>
mongos> db.clientes.stats().nchunks
1
mongos> db.clientes.getShardDistribution()

Shard shard0001 at localhost:27052
 data : 53KiB docs : 1000 chunks : 1
 estimated data per chunk : 53KiB
 estimated docs per chunk : 1000

Totals
 data : 53KiB docs : 1000 chunks : 1
 Shard shard0001 contains 100% data, 100% docs in cluster, avg obj size on shard : 55B
```

Storage Engine

```
ubuntu@ubuntu2004:~$ mongod --fork --storageEngine mmapv1 --logpath /home/ubuntu/log/mongodb_se_mmap.log --dbpath=/home/ubuntu/se_mmap --port 28000
about to fork child process, waiting until server is ready for connections.
forked process: 79377
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongo localhost:28000/se_mmap
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:28000/se_mmap
Implicit session: session { "id" : UUID("42e8a0d1-2f66-46ba-b54b-bfe152fa3059") }
MongoDB server version: 3.6.8
Server has startup warnings:
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten]
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten]
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2023-12-22T19:56:37.354-0500 I CONTROL  [initandlisten]
>
>
> db.serverStatus().storageEngine
{
        "name" : "mmapv1",
        "supportsCommittedReads" : false,
        "readOnly" : false,
        "persistent" : true
}
```

```
> db.createCollection("rr_produtos")
{ "ok" : 1 }
>
>
> show dbs
admin    0.078GB
config   0.078GB
local    0.078GB
se_mmap  0.078GB
>
```

```
>
> db.rr_produtos.insert({ produto: "notebook", valor: 4235.00})
WriteResult({ "nInserted" : 1 })
>         db.rr_produtos.insert({ produto: "placa de video", valor: 850.00})
WriteResult({ "nInserted" : 1 })
>         db.rr_produtos.insert({ produto: "roteador", valor: 2135.00})
WriteResult({ "nInserted" : 1 })
>         db.rr_produtos.insert({ produto: "switch", valor: 1135.00})
WriteResult({ "nInserted" : 1 })
>
>
> db.rr_produtos.find()
{ "_id" : ObjectId("65863659ababe0d207e3faef"), "produto" : "notebook", "valor" : 4235 }
{ "_id" : ObjectId("65863659ababe0d207e3faf0"), "produto" : "placa de video", "valor" : 850 }
{ "_id" : ObjectId("65863659ababe0d207e3faf1"), "produto" : "roteador", "valor" : 2135 }
{ "_id" : ObjectId("6586365dababe0d207e3faf2"), "produto" : "switch", "valor" : 1135 }
>
>
```

Criamos a instância de segurança, conectamos na mesma, vimos que pelo erro apresentado no database segurança, que nós não temos privilégio para executar o comando show dbs.

Criamos o database admin, conectamos, criamos através da variável 'userdba', para posteriormente criarmos o usuário dba com a senha e seus devidos privilégios de acesso.

```
ubuntu     79749       1  3 21:01 ?        00:00:02 mongod --fork --logpath /home/ubuntu/log/mongodb_segur.log --dbpath=/home/ubuntu/segur --port 30000 --auth
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongo localhost:30000/seguranca
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/seguranca
Implicit session: session { "id" : UUID("b3a9f579-4f31-4a82-9eae-1be8b3af94fe") }
MongoDB server version: 3.6.8
>
>
> db
seguranca
> show dbs
2023-12-22T21:03:27.532-0500 E QUERY    [thread1] Error: listDatabases failed:{
        "ok" : 0,
        "errmsg" : "not authorized on admin to execute command { listDatabases: 1.0, lsid: { id: UUID(\"b3a9f579-4f31-4a82-9eae-1be8b3af94fe\") }, $db: \"admin\" }",
        "code" : 13,
        "codeName" : "Unauthorized"
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:67:1
shellHelper.show@src/mongo/shell/utils.js:860:19
shellHelper@src/mongo/shell/utils.js:750:15
@(shellhelp2):1:1
>
>
> use admin
switched to db admin
>
> db
admin
>
> var usuariodba = { user : "dba", pwd : "senhadba", roles : [ "userAdminAnyDatabase", "readWriteAnyDatabase" ]}
>
> db.createUser (usuariodba)
Successfully added user: {
        "user" : "dba",
        "roles" : [
                "userAdminAnyDatabase",
                "readWriteAnyDatabase"
        ]
}
```

Criando o usuário rr_rafael, com role readWrite, através do usuário dba.

```
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("fd6d542d-ce28-402c-bacb-cd7a54193400") }
MongoDB server version: 3.6.8
>
> use rr_rh
switched to db rr_rh
>
> var usuariodesenv = { user : "rr_rafael", pwd : "rafadev", roles : [ "readWrite" ]}
> db.createUser (usuariodesenv)
Successfully added user: { "user" : "rr_rafael", "roles" : [ "readWrite" ] }
>
```

Acessando com as credenciais do usuário rr_rafael.

```
ubuntu@ubuntu2004:~$ mongo localhost:30000/rr_rh -u rr_rafael -p rafadev
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/rr_rh
Implicit session: session { "id" : UUID("cfdc6a5b-492f-4c9c-94f2-32676ffa0fde") }
MongoDB server version: 3.6.8
>
>
> db
rr_rh
```

Inserindo uma coleção através do usuário rr_rafael.

```
> db.rr_funcionarios.insert({ nome: "Roberta", setor: "Financeiro"})
        db.rr_funcionarios.insert({ nome: "Rafael", setor: "Engenharia"})
        db.rr_funcionarios.insert({ nome: "Juliana", setor: "Pesquisa"})
        db.rr_funcionarios.insert({ nome: "Lucas", setor: "Desenvolvimento"})WriteResult({ "nInserted" : 1 })
>         db.rr_funcionarios.insert({ nome: "Rafael", setor: "Engenharia"})
WriteResult({ "nInserted" : 1 })
>         db.rr_funcionarios.insert({ nome: "Juliana", setor: "Pesquisa"})
WriteResult({ "nInserted" : 1 })
>         db.rr_funcionarios.insert({ nome: "Lucas", setor: "Desenvolvimento"})
WriteResult({ "nInserted" : 1 })
>
>
> show collections
rr_funcionarios
>
> db.rr_funcionarios.find()
{ "_id" : ObjectId("65865491f404abebb9cfc5ad"), "nome" : "Roberta", "setor" : "Financeiro" }
{ "_id" : ObjectId("65865491f404abebb9cfc5ae"), "nome" : "Rafael", "setor" : "Engenharia" }
{ "_id" : ObjectId("65865491f404abebb9cfc5af"), "nome" : "Juliana", "setor" : "Pesquisa" }
{ "_id" : ObjectId("65865492f404abebb9cfc5b0"), "nome" : "Lucas", "setor" : "Desenvolvimento" }
>
```

Depuração, Backup/Restore

Abaixo nós criamos o diretório tuning e a instância.

```
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/tuning
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_tuning.log
 --dbpath=/home/ubuntu/tuning --port 32000
about to fork child process, waiting until server is ready for connections.
forked process: 80375
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu     80375       1  9 09:58 ?        00:00:01 mongod --fork --logpath /hom
e/ubuntu/log/mongodb_tuning.log --dbpath=/home/ubuntu/tuning --port 32000
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
```

Conectamos localmente via porta 32000 no db_tunning, verificamos o db corrente e fomos para o db rr_ideagreg.

```
ubuntu@ubuntu2004:~$ mongo localhost:32000/db_tuning
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:32000/db_tuning
Implicit session: session { "id" : UUID("fdf58e25-2f71-4082-adfa-70242e21ab2d") }
MongoDB server version: 3.6.8
Server has startup warnings:
2023-12-23T09:58:19.107-0500 I STORAGE  [initandlisten]
2023-12-23T09:58:19.107-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS f
ilesystem is strongly recommended with the WiredTiger storage engine
2023-12-23T09:58:19.107-0500 I STORAGE  [initandlisten] **          See http://doch
ub.mongodb.org/core/prodnotes-filesystem
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten]
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten] ** WARNING: Access control
is not enabled for the database.
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten] **          Read and write
access to data and configuration is unrestricted.
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten]
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten] ** WARNING: This server is
bound to localhost.
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten] **          Remote systems
will be unable to connect to this server.
2023-12-23T09:58:20.021-0500 I CONTROL  [initandlisten] **          Start the serve
r with --bind_ip <address> to specify which IP
2023-12-23T09:58:20.022-0500 I CONTROL  [initandlisten] **          addresses it sh
ould serve responses from, or with --bind_ip_all to
2023-12-23T09:58:20.022-0500 I CONTROL  [initandlisten] **          bind to all int
erfaces. If this behavior is desired, start the
2023-12-23T09:58:20.022-0500 I CONTROL  [initandlisten] **          server with --b
ind_ip 127.0.0.1 to disable this warning.
2023-12-23T09:58:20.022-0500 I CONTROL  [initandlisten]
>
> db
db_tuning
>
>
>
> use rr_ideagreg
switched to db rr_ideagreg
>
```

Criamos a collection rr_Paises e demos um insert de 4 documentos.

```
>
> db.createCollection("rr_Paises")
{ "ok" : 1 }
>
>
> db.rr_Paises.insert({ cod_ddi: "+55", nome: "Brasil",  continente:"Sul Americano"})
WriteResult({ "nInserted" : 1 })
> db.rr_Paises.insert({ cod_ddi: "+49" , nome: "Alemanha", continente:"Europeu"})
WriteResult({ "nInserted" : 1 })
> db.rr_Paises.insert({ cod_ddi: "+81", nome: "Japão", continente:"Asiático"})
WriteResult({ "nInserted" : 1 })
> db.rr_Paises.insert({ cod_ddi: "+20", nome: "Egito", continente:"Africano"})
WriteResult({ "nInserted" : 1 })
>
> db.rr_Paises.find()
{ "_id" : ObjectId("65870061e81cfbc2541c5d21"), "cod_ddi" : "+55", "nome" : "Brasil", "continente" : "Sul Americano" }
{ "_id" : ObjectId("65870061e81cfbc2541c5d22"), "cod_ddi" : "+49", "nome" : "Alemanha", "continente" : "Europeu" }
{ "_id" : ObjectId("65870061e81cfbc2541c5d23"), "cod_ddi" : "+81", "nome" : "Japão", "continente" : "Asiático" }
{ "_id" : ObjectId("65870062e81cfbc2541c5d24"), "cod_ddi" : "+20", "nome" : "Egito", "continente" : "Africano" }
>
>
```

Aqui fizemos lima consulta filtrando pelo continente e usando o método explain.

Podemos verificar em winningPlan que foi feito um COLLSCAN, ou seja, para retornar a consulta foi necessário verificar linha por linha da coluna continente, isso porque que não há índice nenhum criado na mesma.

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

Criação do índice na coluna continente como 1, ou seja, ascendente.

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

Aplicando novamente o mesmo comando anteriormente, podemos verificar dessa vez em winningPlan, que o foi realizado um IXSCAN, ou seja, foi utilizado nosso índice criado, para que nossa consulta tenha um desempenho significativamente melhor.

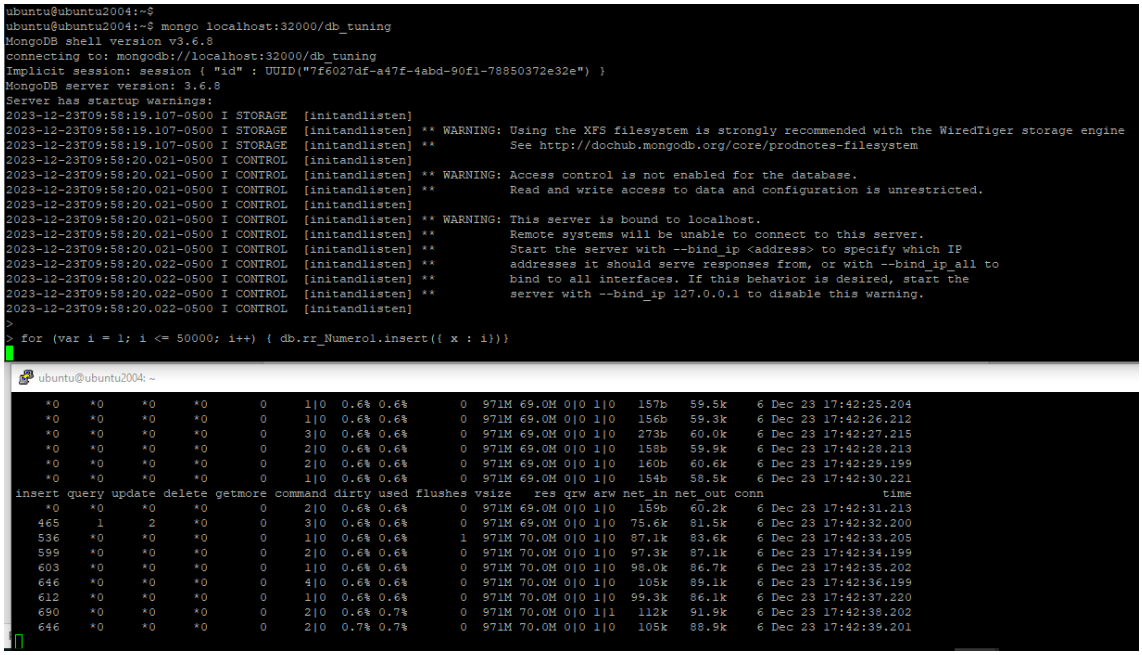
```
db.rr_Paises.find({continente:"Sul Americano"}).explain()

        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "rr_ideagreg.rr_Paises",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "continente" : {
                                "$eq" : "Sul Americano"
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "continente" : 1
                                },
                                "indexName" : "continente_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "continente" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "continente" : [
                                                "[\"Sul Americano\", \"Sul Americano\"]"
                                        ]
                                }
                        }
```

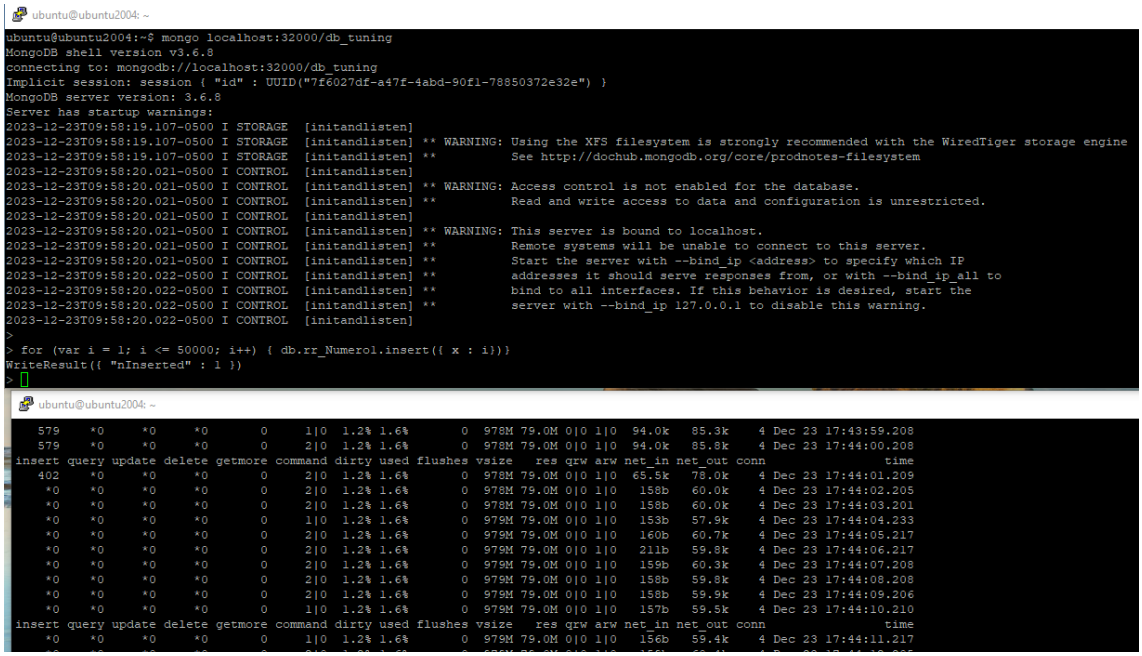
Mongostat

Na parte superior do print, conectamos localmente pela porta 32000 no db_tunning e na tela de baixo, chamamos o mongostat que é uma ferramenta do MongoDB para acompanhamento em tempo real de gravações, informações de CPU, memória e etc.

Inserimos a coleção rr_Numero1, com 50000 documentos e podemos notar pelo mongostat na coluna net_in o salto de dados de bytes para kbytes.



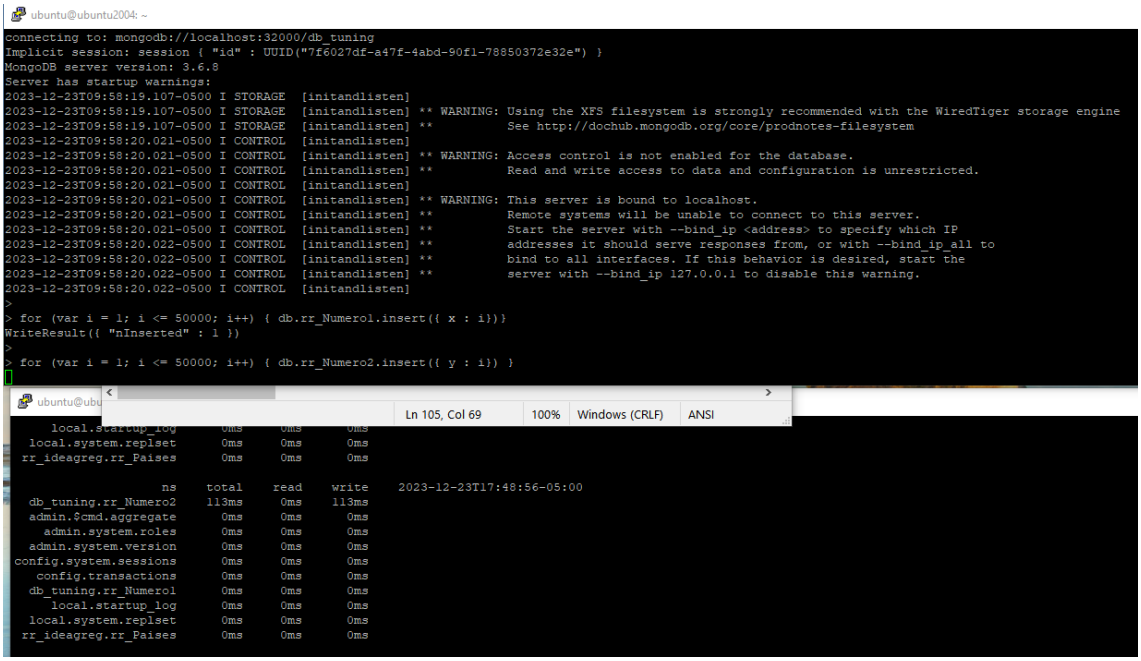
Aqui podemos reparar após o término da criação da coleção, que o mongostat informa que o tráfego de dados diminuiu voltando para a “casa” dos bytes.



Mongotop

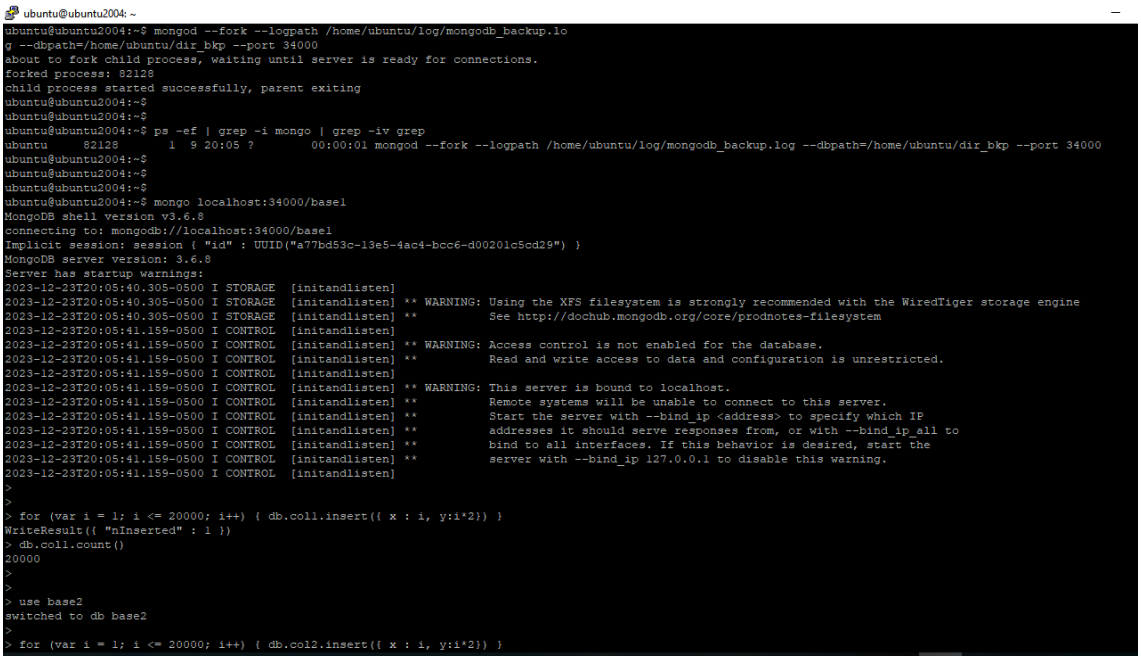
Aqui utilizamos o mongotop, onde nos diz qual coleção está sendo mais utilizadas.

Fizemos a criação da coleção rr_Numero2 com 50000 documentos, e podemos verificar na parte inferior do print o tempo de escrita que ele está contabilizando até o momento.



Backup e Restore

Criando a instância e conectando no database base1 e base2, para criarmos as collections.



Fazendo o backup através do mongodump e especificando o caminho para o backup.

```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongodump --host 127.0.0.1:34000 --out /home/ubuntu/backup/bkp_completo
2023-12-23T20:08:24.414-0500    writing admin.system.version to
2023-12-23T20:08:24.419-0500    done dumping admin.system.version (1 document)
2023-12-23T20:08:24.419-0500    writing base1.coll1 to
2023-12-23T20:08:24.420-0500    writing base2.col2 to
2023-12-23T20:08:24.654-0500    done dumping base2.col2 (20000 documents)
2023-12-23T20:08:24.677-0500    done dumping base1.coll1 (20000 documents)
ubuntu@ubuntu2004:~$ ls -lR /home/ubuntu/backup/bkp_completo
/home/ubuntu/backup/bkp_completo:
total 12
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 23 20:08 admin
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 23 20:08 base1
drwxrwxr-x 2 ubuntu ubuntu 4096 Dec 23 20:08 base2

/home/ubuntu/backup/bkp_completo/admin:
total 8
-rw-rw-r-- 1 ubuntu ubuntu  59 Dec 23 20:08 system.version.bson
-rw-rw-r-- 1 ubuntu ubuntu 134 Dec 23 20:08 system.version.metadata.json

/home/ubuntu/backup/bkp_completo/base1:
total 864
-rw-rw-r-- 1 ubuntu ubuntu 880000 Dec 23 20:08 coll1.bson
-rw-rw-r-- 1 ubuntu ubuntu    124 Dec 23 20:08 coll1.metadata.json

/home/ubuntu/backup/bkp_completo/base2:
total 864
-rw-rw-r-- 1 ubuntu ubuntu 880000 Dec 23 20:08 col2.bson
-rw-rw-r-- 1 ubuntu ubuntu    124 Dec 23 20:08 col2.metadata.json
ubuntu@ubuntu2004:~$
```

Matando o processo do mongo server

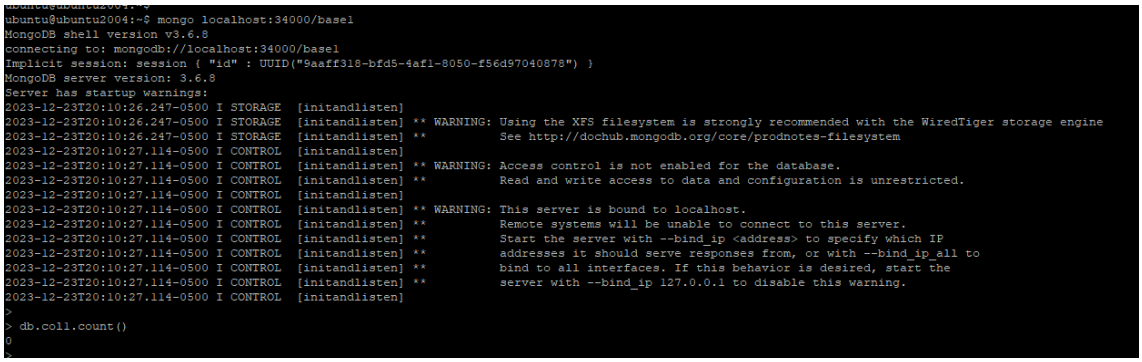
```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep
ubuntu     82128       1 21 20:05 ?        00:00:39 mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --dbpath=/home/ubuntu/dir_bkp --port 34000
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ kill -9 82128
ubuntu@ubuntu2004:~$
```

Excluindo o caminho /home/ubuntu/dir_bkp

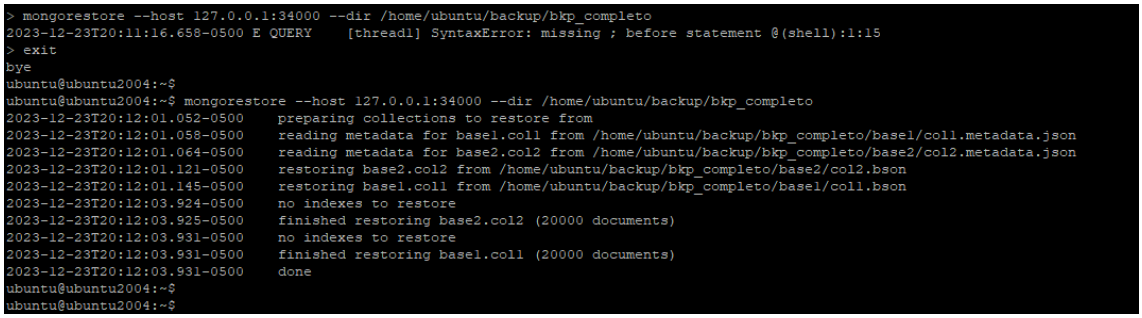
ubuntu@ubuntu2004: ~

```
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ls -l /home/ubuntu/dir_bkp
total 1572
-rw------- 1 ubuntu ubuntu  16384 Dec 23 20:06 collection-0--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu  16384 Dec 23 20:06 collection-2--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu   4096 Dec 23 20:05 collection-4--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu 458752 Dec 23 20:07 collection-7--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu 462848 Dec 23 20:08 collection-9--1291073343524828663.wt
drwx------ 2 ubuntu ubuntu   4096 Dec 23 20:09 diagnostic.data
-rw------- 1 ubuntu ubuntu 221184 Dec 23 20:08 index-10--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu  16384 Dec 23 20:06 index-1--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu  16384 Dec 23 20:06 index-3--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu   4096 Dec 23 20:05 index-5--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu   4096 Dec 23 20:06 index-6--1291073343524828663.wt
-rw------- 1 ubuntu ubuntu 217088 Dec 23 20:07 index-8--1291073343524828663.wt
drwx------ 2 ubuntu ubuntu   4096 Dec 23 20:05 journal
-rw------- 1 ubuntu ubuntu  32768 Dec 23 20:07 _mdb_catalog.wt
-rw------- 1 ubuntu ubuntu      6 Dec 23 20:05 mongod.lock
-rw------- 1 ubuntu ubuntu  36864 Dec 23 20:08 sizeStorer.wt
-rw------- 1 ubuntu ubuntu    114 Dec 23 20:05 storage.bson
-rw------- 1 ubuntu ubuntu     46 Dec 23 20:05 WiredTiger
-rw------- 1 ubuntu ubuntu   4096 Dec 23 20:05 WiredTigerLAS.wt
-rw------- 1 ubuntu ubuntu     21 Dec 23 20:05 WiredTiger.lock
-rw------- 1 ubuntu ubuntu   1082 Dec 23 20:08 WiredTiger.turtle
-rw------- 1 ubuntu ubuntu  69632 Dec 23 20:08 WiredTiger.wt
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ rm  -r  /home/ubuntu/dir_bkp
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ ls -l /home/ubuntu/dir_bkp
ls: cannot access '/home/ubuntu/dir_bkp': No such file or directory
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/dir_bkp
ubuntu@ubuntu2004:~$
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --dbpath=/home/ubuntu/dir_bkp --port 34000
about to fork child process, waiting until server is ready for connections.
forked process: 82193
child process started successfully, parent exiting
```

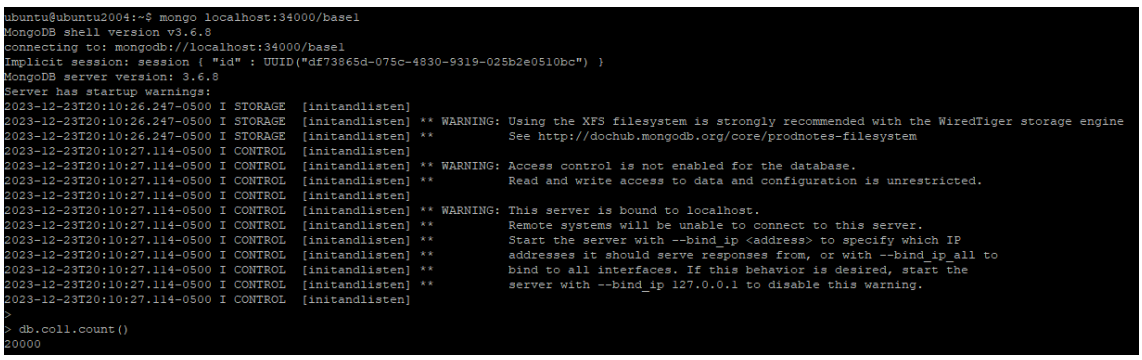
Após subir a instância novamente, conectando pelo localhost, verificamos que os dados no database base1 não existem mais.



Aqui aplicamos a restauração através do mongorestore, informando o caminho desejado.



Conectando novamente na base1, verificamos que os dados retoranaram.



Acessamos a instância para irmos ao database rr_ideagreg para fazermos um count na collection rr_Paises.

```
ubuntu@ubuntu2004:~$ mongo localhost:32000/db_tuning;
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:32000/db_tuning
Implicit session: session { "id" : UUID("c3b3cc06-b022-4994-a8c1-2532f816dff6") }
MongoDB server version: 3.6.8
Server has startup warnings:
2023-12-23T20:30:39.920-0500 I STORAGE  [initandlisten]
2023-12-23T20:30:39.920-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2023-12-23T20:30:39.920-0500 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten]
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten]
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2023-12-23T20:30:41.954-0500 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2023-12-23T20:30:41.955-0500 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2023-12-23T20:30:41.955-0500 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2023-12-23T20:30:41.955-0500 I CONTROL  [initandlisten]
>
>
> db
db_tuning
>
> show dbs
admin        0.000GB
config       0.000GB
db_tuning    0.003GB
local        0.000GB
rr_ideagreg  0.000GB
>
> use rr_ideagreg
switched to db rr_ideagreg
>
> show collections
rr_Paises
>
>
> db.rr_Paises.count()
4
```