

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE  
SÃO PAULO**

**WESLEY SCHUAB VIEIRA**

**MongoDB Teoria e Prática**

**CAMPOS DO JORDÃO**

**2024**

## RESUMO

O MongoDB, um banco de dados NoSQL orientado a documentos, oferece uma alternativa ágil e escalável aos tradicionais bancos de dados relacionais. Sua arquitetura distribuída e o modelo de dados flexível, baseado em coleções e documentos JSON, permitem armazenar e consultar dados de forma eficiente e intuitiva. Com

Features como replicação, sharding e índices, o MongoDB garante alta disponibilidade e desempenho, mesmo em ambientes de alta carga. A linguagem de consulta rica e expressiva, baseada em JSON, facilita a integração com aplicações desenvolvidas em diversas linguagens, como Java, Node.js e Python.

**Palavras-Chave:** MongoDB, NoSQL, Java, JSON.

## **ABSTRACT**

MongoDB, a document-oriented NoSQL database, offers an agile and scalable alternative to traditional relational databases. Its distributed architecture and flexible data model, based on collections and JSON documents, allow you to store and query data efficiently and intuitively. With features such as replication, sharding and indexes, MongoDB ensures high availability and performance, even in high-load environments. The rich and expressive query language, based on JSON, facilitates integration with applications developed in various languages, such as Java, Node.js and Python.

**Keywords:** MongoDB, NoSQL, Java, JSON.

<b>1 INTRODUÇÃO</b>	<b>4</b>
1.1 Objetivos	5
<b>2. FUNDAMENTAÇÃO TEÓRICA</b>	<b>5</b>
2.1 Surgimento do Bancos Não Relacionais	6
2.2 NoSQL	6
2.3 MongoDB	7
<b>3. METODOLOGIA</b>	<b>7</b>
3.1 Funcionamento do MongoDB	8
3.2 Instalação e Configuração	10
<b>4. CONCLUSÃO:</b>	<b>19</b>

## 1 INTRODUÇÃO

*Segundo Tiwari (2011) a quantidade de dados gerados por essa nova geração tiveram aumento significativo. As bases de dados tradicionais passaram a ter alguns desafios para lidar com que foi chamado de bigdat. Para resolver essas questões surgiu uma nova classe de banco de dados: o NoSQL é um termo abrangente que se refere a bancos de dados que não seguem o modelo relacional tradicional (baseado em tabelas, linhas e colunas). Essa flexibilidade permite que eles sejam adaptados a uma variedade de tipos de dados e cargas de trabalho, tornando-os ideais para aplicações modernas, como bigdata, internet das coisas (IoT) e aplicativos móveis. Uma das características desse modelo é que em vez de armazenar dados em linhas (como em bancos relacionais), os dados são organizados em colunas. Isso é particularmente eficiente quando você precisa analisar grandes volumes de dados com base em um subconjunto de colunas, temos também o conceito de “ Bancos de Dados Chave-Valor “, Os dados são armazenados como pares de chave-valor, onde a chave é um identificador único e o valor pode ser qualquer tipo de dado. Perfeito para aplicações que exigem acesso rápido a dados individuais, como caches, sessões de usuário e armazenamento de configurações, e podemos citar os “Bancos de Dados Documentais”, Os dados são armazenados em documentos, que são*

*estruturas de dados flexíveis semelhantes a JSON. Cada documento pode ter diferentes campos e níveis de aninhamento. Para analisarmos os bancos de dados não relacionais, escolhi o MongoDB. O objetivo do trabalho é mostrar que os bancos de dados NoSQL, oferecem uma alternativa viável e eficiente para o armazenamento e manipulação de dados, especialmente em cenários onde a flexibilidade e a escalabilidade são importantes.*

## **1.1 Objetivos**

O objetivo deste trabalho é explorar e demonstrar como os bancos de dados NoSQL, com ênfase no MongoDB, podem oferecer uma alternativa viável e eficiente para o armazenamento e a manipulação de dados em cenários que demandam alta flexibilidade e escalabilidade. Por meio de uma análise do modelo não relacional, será evidenciada sua capacidade de atender às necessidades de aplicações modernas, como bigdata, Internet das Coisas (IoT) e aplicativos móveis. Além disso, o trabalho busca destacar as vantagens do MongoDB em termos de organização de dados, desempenho e adaptação a diferentes tipos de cargas de trabalho, validando sua aplicabilidade em ambientes de grandes volumes de dados e requisitos dinâmicos.

## **2. FUNDAMENTAÇÃO TEÓRICA**

Esse trabalho tem como base outros artigos que realizam uma análise e comparação entre os clássicos banco de dados relacionais que utilizam a linguagem SQL e novos conceitos que surgiram para lidar com ambientes que precisam de mais flexibilidade e lidam com grandes volumes de dados conhecidos como NoSQL. Abordaremos as sintaxes e demonstraremos exemplos de aplicações assim e também vamos divagar sobre as estruturas de cada tipo de banco.

## **2.1 Surgimento do Bancos Não Relacionais**

Com o avanço exponencial na geração de dados, impulsionado por tecnologias como Internet das Coisas (IoT), bigdata e aplicativos móveis, os bancos de dados tradicionais enfrentam limitações em termos de flexibilidade, escalabilidade e desempenho. Essas demandas criaram a necessidade de soluções que possam lidar com volumes massivos de dados não estruturados ou semiestruturados, além de oferecer tempos de resposta rápidos e uma estrutura adaptável.

Nesse contexto, os bancos de dados NoSQL surgem como uma alternativa promissora, proporcionando modelos de armazenamento mais flexíveis, que se ajustam às necessidades específicas de diferentes tipos de aplicações. Entre eles, o MongoDB destaca-se por sua estrutura documental, que combina facilidade de uso, alto desempenho e escalabilidade horizontal, atendendo aos desafios das aplicações modernas.

## **2.2 NoSQL**

Not Only SQL é o termo utilizado para o leque que engloba todos os bancos de dados e sistemas de armazenamento que não seguem o tradicional modelo normalizado RDBMS. Segundo Tiwari, Shashank 2011, NoSQL representam uma classe de produtos e coleções de diversos tipos, e às vezes relacionada com conceitos de armazenamento e manipulação.

Segundo Banker(2012) que, o MongoDB nasceu de um projeto muito mais ambicioso, por volta de 2007 uma empresa chamada 10gen iniciou seu trabalho com software do tipo plataforma como serviço, composto por uma aplicação servidor e uma base de dados, que iriam hospedar aplicações web e escalá-las quando necessário.

## **2.3 MongoDB**

O MongoDB é um banco de dados NoSQL que armazena dados em documentos flexíveis, proporcionando uma alternativa aos tradicionais bancos de dados relacionais. Ao invés de tabelas rígidas, O MongoDB foi criado com o intuito de trabalhar com documentos ao invés de linhas, com o objetivo de ser extremamente rápido, amplamente escalável e fácil de usar. Uma grande vantagem que o MongoDB apresenta é a de não precisar fazer com que os seus dados encaixem-se em uma tabela, o que de fato nem sempre é possível. Isto significa que o MongoDB é indicado para armazenar dados complexos (HOWS; MEMBREY; PLUGGE, 2019). Essa característica o torna ideal para aplicações que lidam com dados semiestruturados ou não estruturados, como logs, dados de geolocalização e informações de perfil de usuário. Uma das principais vantagens do MongoDB é sua escalabilidade. Ele permite que os dados sejam distribuídos em vários servidores, proporcionando um alto desempenho e disponibilidade. Além disso, o MongoDB é conhecido por sua facilidade de uso e rápida implementação, o que o torna uma escolha popular para o desenvolvimento de aplicações modernas.

## **3. METODOLOGIA**

MongoDB O MongoDB é um banco de dados NoSQL de código aberto sob a licença GNU AGPL v3.0, escrito em C++, orientado a documentos e livre de schemas. Seu nome é derivado da expressão em inglês humongous, que pode ser traduzido como “enorme” ou “monstruoso”.

O MongoDB inicialmente foi desenvolvido como um componente de serviço pela empresa 10gen em outubro de 2007, passando a ser um software open source em 2009. Atualmente, o MongoDB é um dos mais populares banco de dados NoSQL (senão o mais popular) e está na versão 2.6. O projeto ainda é mantido pela 10gen que oferece suporte comercial e demais serviços.

O principal diferencial para que o MongoDB tenha se tornado mais popular que outros Sistemas de Gerenciamento de Banco de Dados NoSQL, talvez seja a facilidade na instalação.

### **3.1 Funcionamento do MongoDB**

O MongoDB é composto por dois serviços: mongod e mongos. O mongos é usado para autosharding. Sharding é a distribuição física de dados em nós distintos, utilizado quando é necessário o balanceamento de carga.

O mongod é o servidor de banco de dados. É altamente recomendado que o MongoDB seja executado em um sistema operacional 64 bits, uma vez que o tamanho máximo para um banco de dados numa arquitetura de 32 bits é de 2 gigabytes.

Além do servidor, o MongoDB possui um serviço cliente shell padrão, o mongos, que é utilizado para a conexão com o servidor através da porta 27017, mas além deste cliente, estão disponíveis diversos drivers oficiais para linguagens como PHP, JAVA, C, C++, entre outros.

O servidor de MongoDB pode armazenar um ou mais banco de dados distintos, onde cada banco de dados é constituído de uma ou mais coleções e cada coleção pode ter um ou mais documentos. Estes documentos são estruturados como documentos JSON e armazenados no formato BSON (Binary JSON).

JSON (JavaScript Object Notation), é uma estrutura de dados criada em javascript, consistindo no padrão atributo/valor, parecido com arrays. O JSON pode ser uma alternativa ao XML, onde por vezes se sobressai na questão da velocidade de leitura dos dados nele armazenados.

Exemplo de estrutura JSON:

```
{“nome”: “joao”, “idade” : “20”}, {“nome”: “maria”, “idade” : “22”}.
```

Podemos observar como funciona este tipo de query utilizado pelo MongoDB no seguinte exemplo, onde é necessário retornar o nome de todos os usuários que tenham idade acima de 28 anos.

Tabela de comparativo entre MySQL e MongoDB.



<b>SQL :</b> <b>SELECT nome FROM usuarios WHERE idade &gt; 28</b>
<b>MongoDB Query:</b> <b>db.usuarios.find( { ' nome ', ' idade ' : { ' \$gt ' : 28 } } );</b>

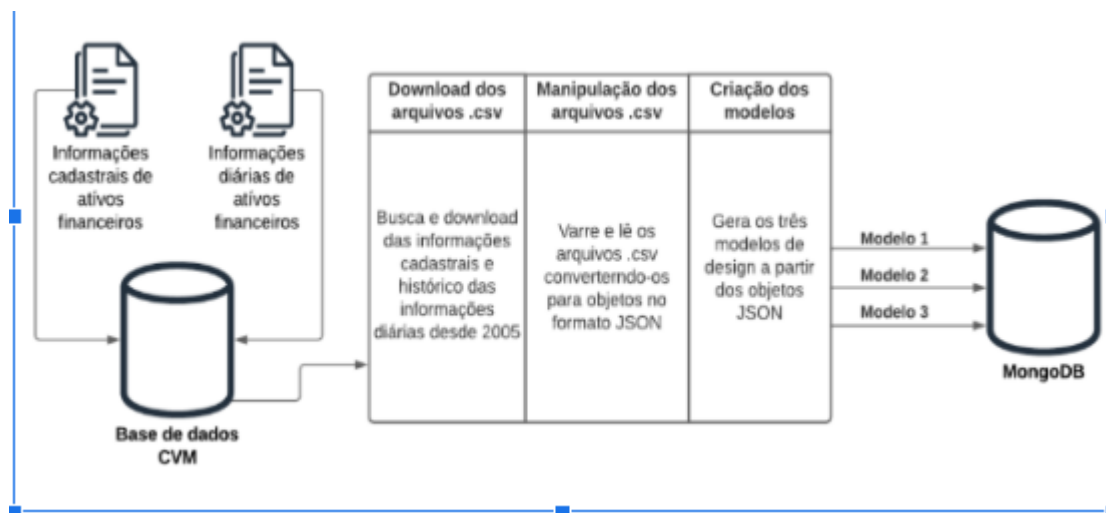
**Figura 1 – Select -MongoDB**

Em uma comparação com o modelo relacional, coleções são equivalentes a tabelas e documentos equivalentes a tuplas(ou linhas) que são armazenados na tabela.

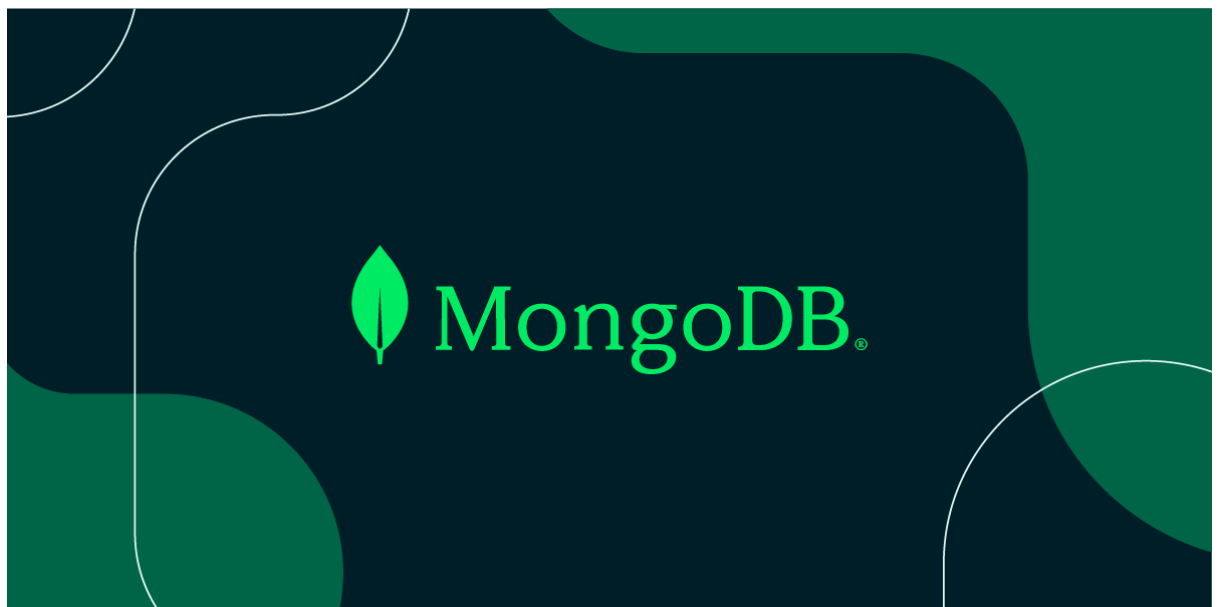
Tabela 1 – Termos utilizados (In: <http://www.mongodb.org>, 2012).

MySQL term	Mongo term/concept
database	database
table	collection
index	index
row	BSON documment
column	BSON field
join	Embedding and linking
primary key	_id field
group by	aggregation

As consultas aos dados são feitas a partir da sintaxe JSON e enviadas para o servidor através de drivers no formato BSON, onde neste modelo, é possível buscar informações em todos os documentos de uma coleção.



**Figura 2** – Esquemático do funcionamento da aplicação.



**Figura 3** – Logo do MongoDB

No MongoDB, não existem recursos para transações e também joins, que são muito utilizados em banco de dados relacionais, ficando por responsabilidade dos desenvolvedores implementar na aplicação caso seja realmente necessário.

### 3.2 Instalação e Configuração

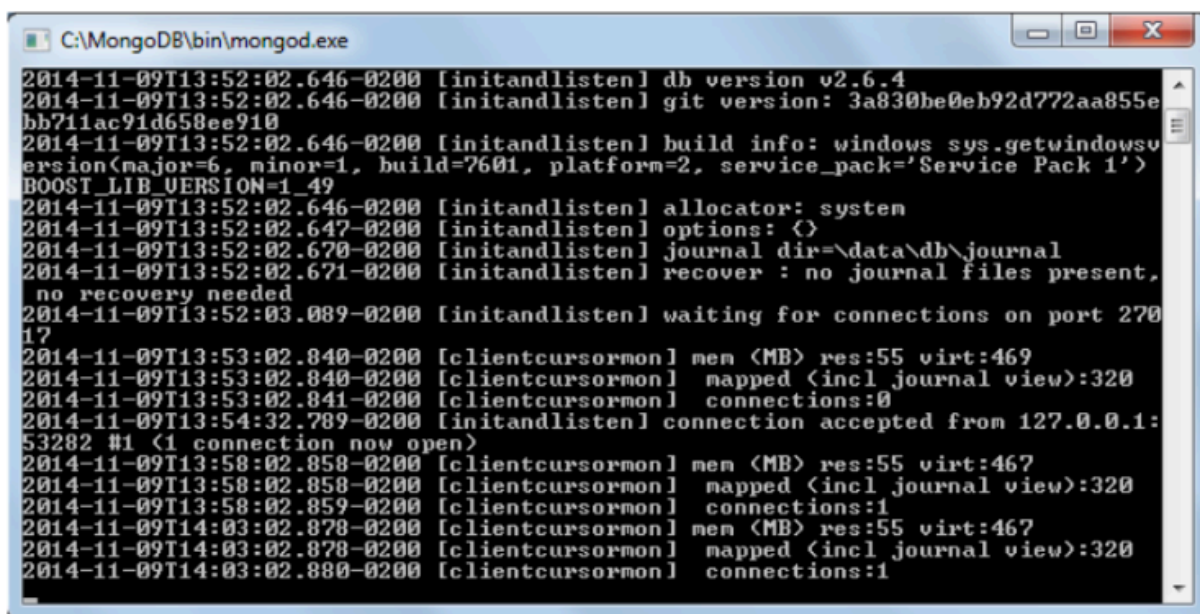
Após o download do arquivo, no caso MongoDB pra Windows, basta executar o arquivo recebido para que a instalação seja concluída. Os arquivos do MongoDB

ficarão em um diretório em “C:”, caso seu sistema tenha sido instalado em uma partição com rótulo diferente a letra pode ser diferente como D ou E por exemplo.

Por padrão, os bancos, documentos e coleções serão armazenados no diretório C:\data\db, portanto é necessário criar estes diretórios caso não existam. Porém, é possível configurar para que os dados sejam armazenados em outro diretório de preferência. Para isso, é necessário sempre inicializar o servidor via prompt de comando do Windows digitando o comando a seguir:

```
C:\mongodb\bin\mongod.exe --dbpath c:\novodir\db\data
```

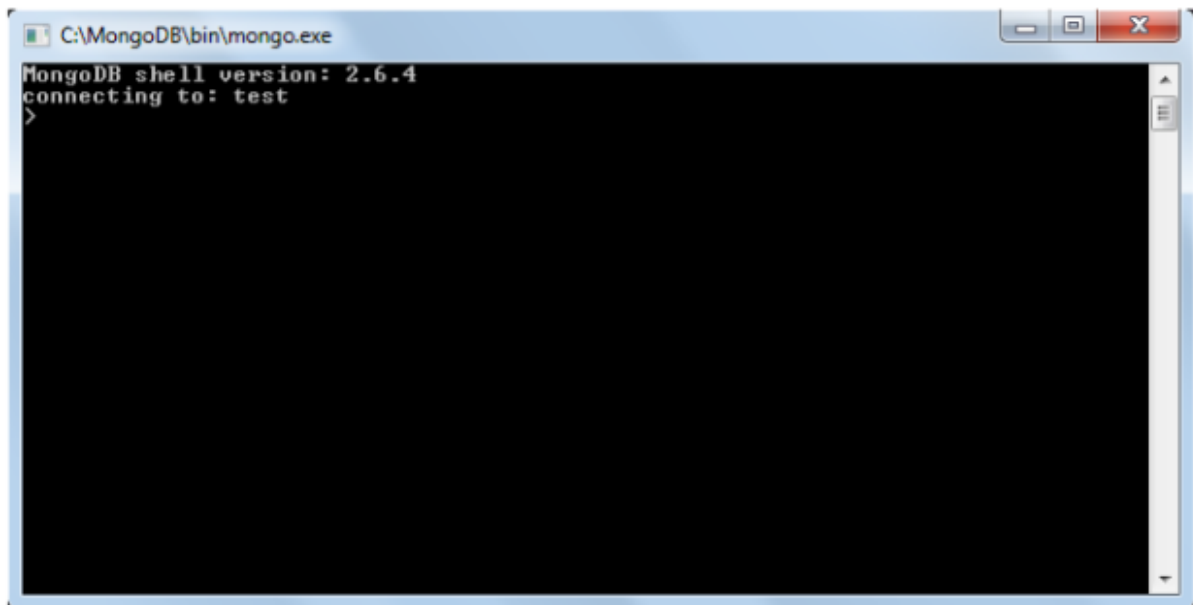
É necessário sempre executar o executável mongod.exe para que o servidor fique online, ao fecha-lo o servidor ficará off-line automaticamente.



```
C:\MongoDB\bin\mongod.exe
2014-11-09T13:52:02.646-0200 [initandlisten] db version v2.6.4
2014-11-09T13:52:02.646-0200 [initandlisten] git version: 3a830be0eb92d772aa855e
bb711ac91d658ee910
2014-11-09T13:52:02.646-0200 [initandlisten] build info: windows sys.getwindowsv
ersion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1')
BOOST_LIB_VERSION=1_49
2014-11-09T13:52:02.646-0200 [initandlisten] allocator: system
2014-11-09T13:52:02.647-0200 [initandlisten] options: {}
2014-11-09T13:52:02.670-0200 [initandlisten] journal dir=\data\db\journal
2014-11-09T13:52:02.671-0200 [initandlisten] recover : no journal files present,
no recovery needed
2014-11-09T13:52:03.089-0200 [initandlisten] waiting for connections on port 270
17
2014-11-09T13:53:02.840-0200 [clientcursormon] mem <MB> res:55 virt:469
2014-11-09T13:53:02.840-0200 [clientcursormon] mapped <incl journal view>:320
2014-11-09T13:53:02.841-0200 [clientcursormon] connections:0
2014-11-09T13:54:32.789-0200 [initandlisten] connection accepted from 127.0.0.1:
53282 #1 <1 connection now open>
2014-11-09T13:58:02.858-0200 [clientcursormon] mem <MB> res:55 virt:467
2014-11-09T13:58:02.858-0200 [clientcursormon] mapped <incl journal view>:320
2014-11-09T13:58:02.859-0200 [clientcursormon] connections:1
2014-11-09T14:03:02.878-0200 [clientcursormon] mem <MB> res:55 virt:467
2014-11-09T14:03:02.878-0200 [clientcursormon] mapped <incl journal view>:320
2014-11-09T14:03:02.880-0200 [clientcursormon] connections:1
```

**Figura 4 - Tela Servidor MongoDB**

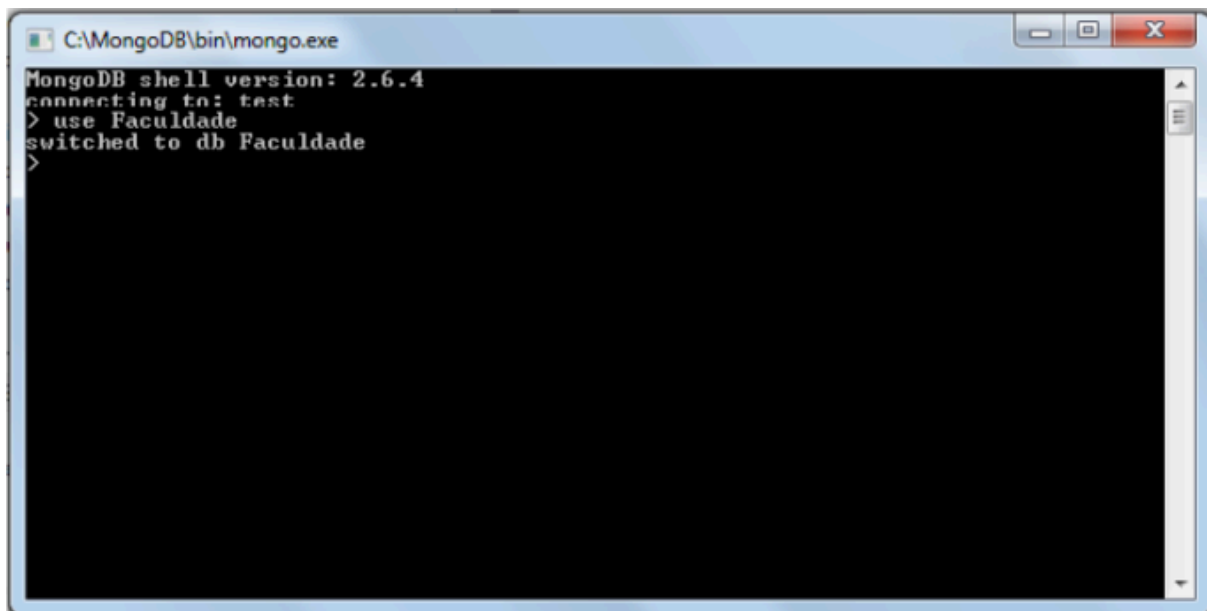
Na figura 4, podemos ver a tela exibida ao executar o arquivo mongod.exe, que inicializa o servidor. Manipulação de dados Para abrir o cliente do MongoDB, basta executar o arquivo mongo.exe encontrado no diretório “bin”.



**Figura 5** - Cliente MongoDB

Na imagem 5 vemos a tela exibida ao executar o cliente mongo.exe. Pode-se notar que somos automaticamente conectados ao banco de dados padrão nomeado "test".

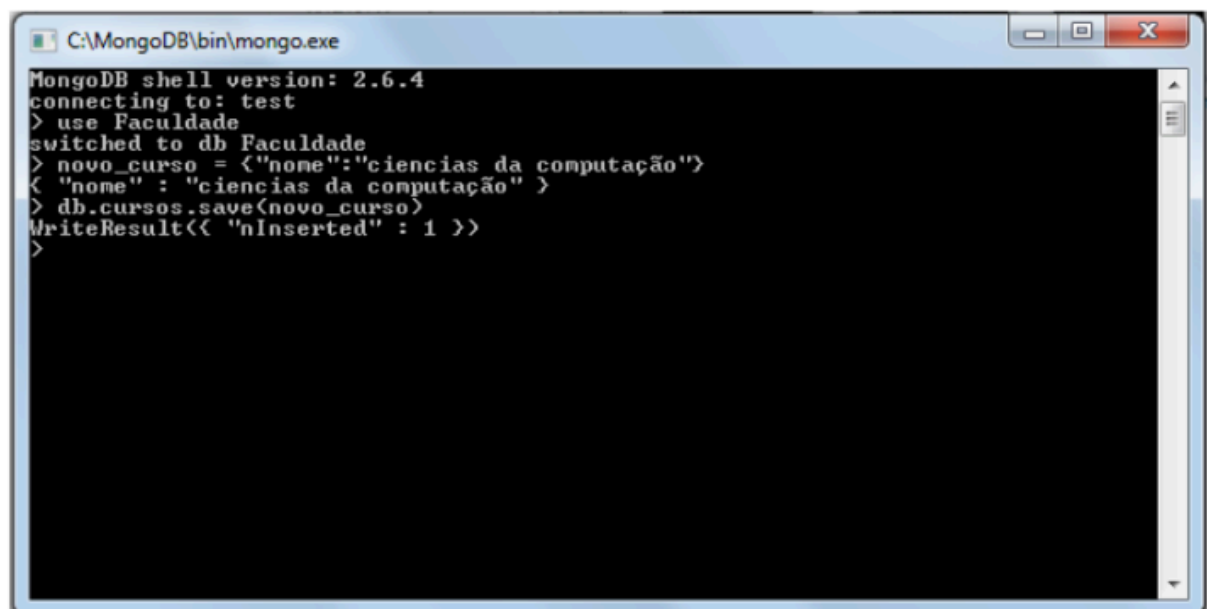
No MongoDB, não é necessário criar uma coleção para começarmos a inserir documentos e informações. Isto é feito automaticamente ao inserir o primeiro registro. Supondo que vamos criar o banco de dados "Faculdade", e a coleção "Cursos" (comparando com um banco de dados relacional, "Cursos" é o equivalente a uma tabela), ao invés de executar um comando para criar a tabela, já podemos executar o comando de inserção de registros e o documento é criado automaticamente:



```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
>
```

**Figura 6 – Criando um Banco de Dados**

O comando “use Faculdade” acessa o banco de dados “Faculdade” ou cria um banco de dados caso não exista nenhum outro banco com este nome, indicando também que os próximos comandos são referentes a esta base de dados.

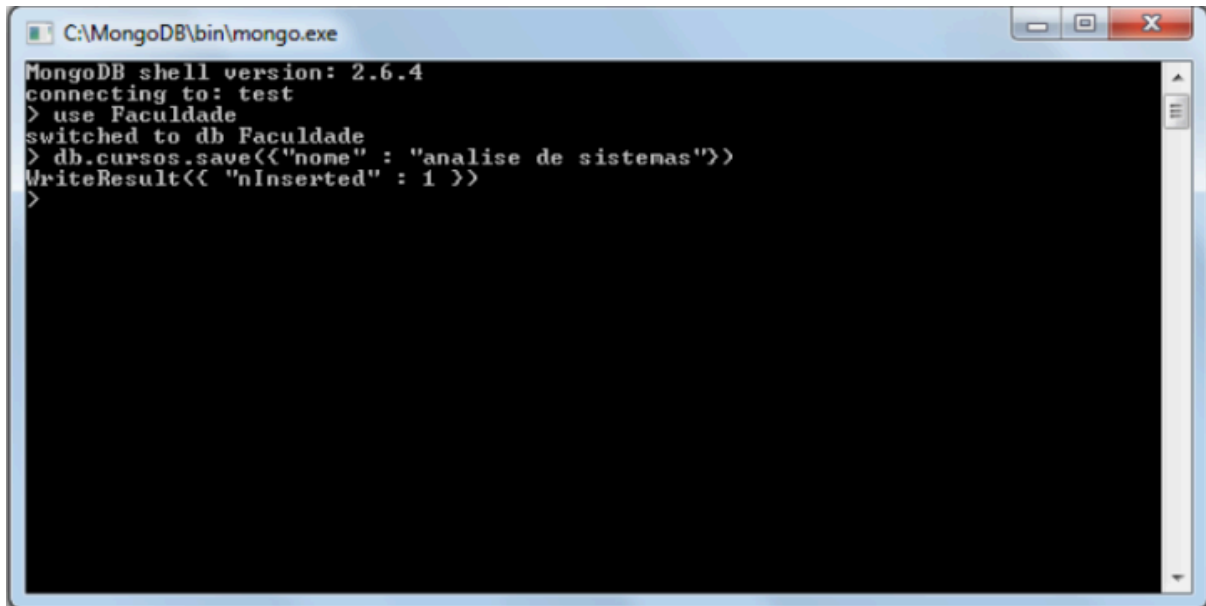


```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> novo_curso = {"nome":"ciencias da computação"}
> {"nome" : "ciencias da computação" }
> db.cursos.save(novo_curso)
WriteResult(< "Inserted" : 1 >)
>
```

**Figura 7 – Comando Inserir**

A figura 7 mostra os próximos comandos para inserir um novo registro na coleção “Cursos”. No caso, inserimos o curso “Ciências da Computação”. O comando foi

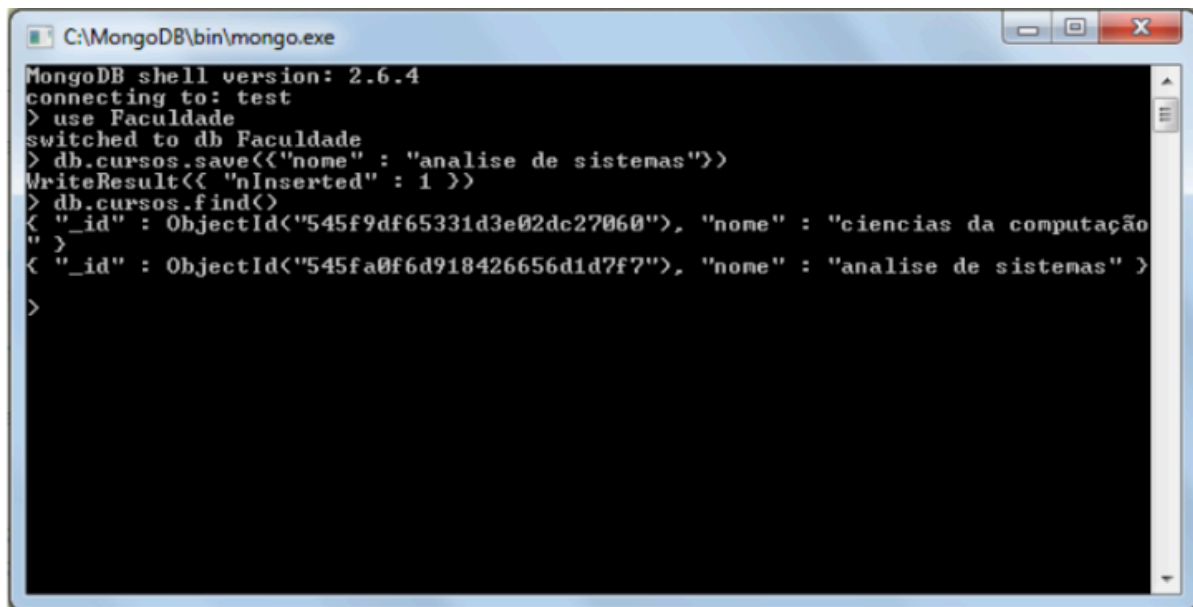
montado com um array chamado “novo\_curso” utilizando a sintaxe em JSON (onde o índice “nome” é equivalente a uma coluna de um banco de dados relacional), seguido do comando “db.cursos.save(novo\_curso)”, onde “cursos” será o nome da coleção e o parâmetro é a variável que armazena o registro a ser inserido. Nota-se que não foi feito nenhum comando para criar a coleção, apenas foi executado o comando de inserção.

A screenshot of a Windows command prompt window titled "C:\MongoDB\bin\mongo.exe". The window displays the MongoDB shell interface. The text shown is: "MongoDB shell version: 2.6.4", "connecting to: test", "> use Faculdade", "switched to db Faculdade", "> db.cursos.save(<<\"nome\" : \"analise de sistemas\">>)", "WriteResult(<< \"nInserted\" : 1 >>)", and ">". The background of the terminal is black, and the text is white. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> db.cursos.save(<<"nome" : "analise de sistemas">>)
WriteResult(<< "nInserted" : 1 >>)
>
```

**Figura 8 - Insert Direto**

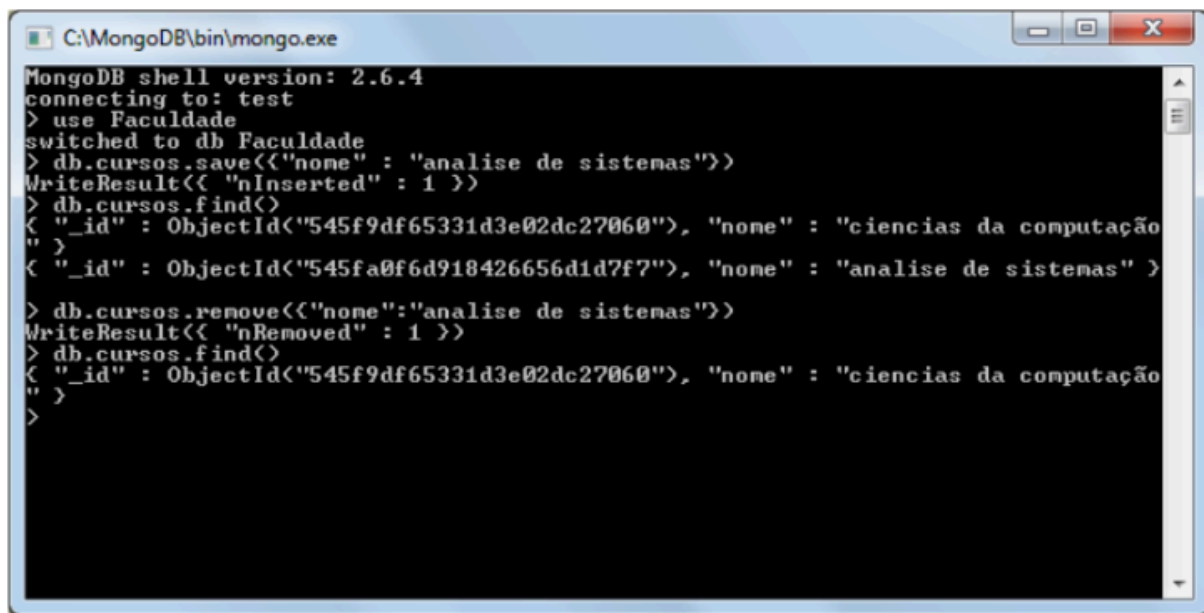
A figura 8 demonstra outra maneira de fazer a inserção, sem utilizar a variável de apoio “novo\_curso”, passando como parâmetro para a função “save()” diretamente o array JSON de dados a ser inserido.



```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> db.cursos.save(<"nome" : "analise de sistemas">)
WriteResult<< "nInserted" : 1 >>
> db.cursos.find()
< "_id" : ObjectId<"545f9df65331d3e02dc27060">, "nome" : "ciencias da computação" >
< "_id" : ObjectId<"545fa0f6d918426656d1d7f7">, "nome" : "analise de sistemas" >
>
```

**Figura 9 - Consultar Registros**

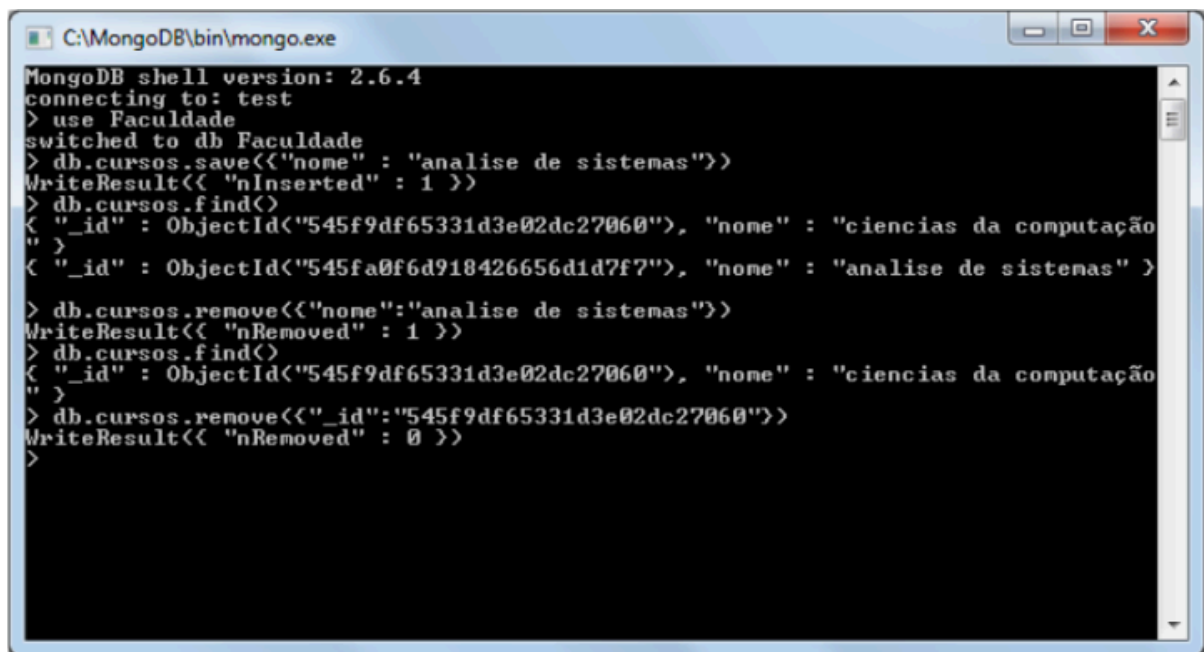
Na figura 9, é possível ver o resultado do comando “db.cursos.find()”, que consulta todos os documentos (ou registros) da coleção “cursos”. Note que é criado automaticamente um atributo chamado “\_id”. Este atributo cria um índice único para os registros, para que seja possível a edição ou exclusão de um registro específico. É basicamente um campo que armazena a hora da criação do registro e codifica tal dado em BSON. Para decodificar este dado, o MongoDB disponibiliza uma função chamada “getTimestamp()”, que retornará um objeto do tipo data como no exemplo: “ISODate(“2012-10-17T20:46:22Z”)”.

A screenshot of a Windows command prompt window titled "C:\MongoDB\bin\mongo.exe". The text inside shows the MongoDB shell version 2.6.4 connecting to a test database. The user switches to the 'Faculdade' database and saves a document with 'nome' as 'analise de sistemas'. Then, they use 'db.cursos.remove()' to delete all documents with that name. A subsequent 'find()' command shows only one document remains with 'nome' as 'ciencias da computação'.

```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> db.cursos.save(<<"nome" : "analise de sistemas">>)
WriteResult<< "nInserted" : 1 >>
> db.cursos.find()
{ "_id" : ObjectId("545f9df65331d3e02dc27060"), "nome" : "ciencias da computação" }
{ "_id" : ObjectId("545fa0f6d918426656d1d7f7"), "nome" : "analise de sistemas" }
> db.cursos.remove(<<"nome":"analise de sistemas">>)
WriteResult<< "nRemoved" : 1 >>
> db.cursos.find()
{ "_id" : ObjectId("545f9df65331d3e02dc27060"), "nome" : "ciencias da computação" }
>
```

**Figura 10 - Remover**

Na figura 10, vemos o comando de exclusão “db.cursos.remove()”. Passamos por parâmetro, o que seria o atributo where da linguagem SQL, neste exemplo estamos excluindo todos os registros cujo atributo nome seja igual a “análise de sistemas”.

A screenshot of a Windows command prompt window titled "C:\MongoDB\bin\mongo.exe". The text inside shows the MongoDB shell version 2.6.4 connecting to a test database. The user switches to the 'Faculdade' database and saves a document with 'nome' as 'analise de sistemas'. Then, they use 'db.cursos.remove()' to delete all documents with that name. A subsequent 'find()' command shows only one document remains with 'nome' as 'ciencias da computação'. Finally, they use 'db.cursos.remove()' with the specific '\_id' of the remaining document. The final 'find()' command shows an empty result set, and the 'remove()' command returns 'nRemoved' as 0.

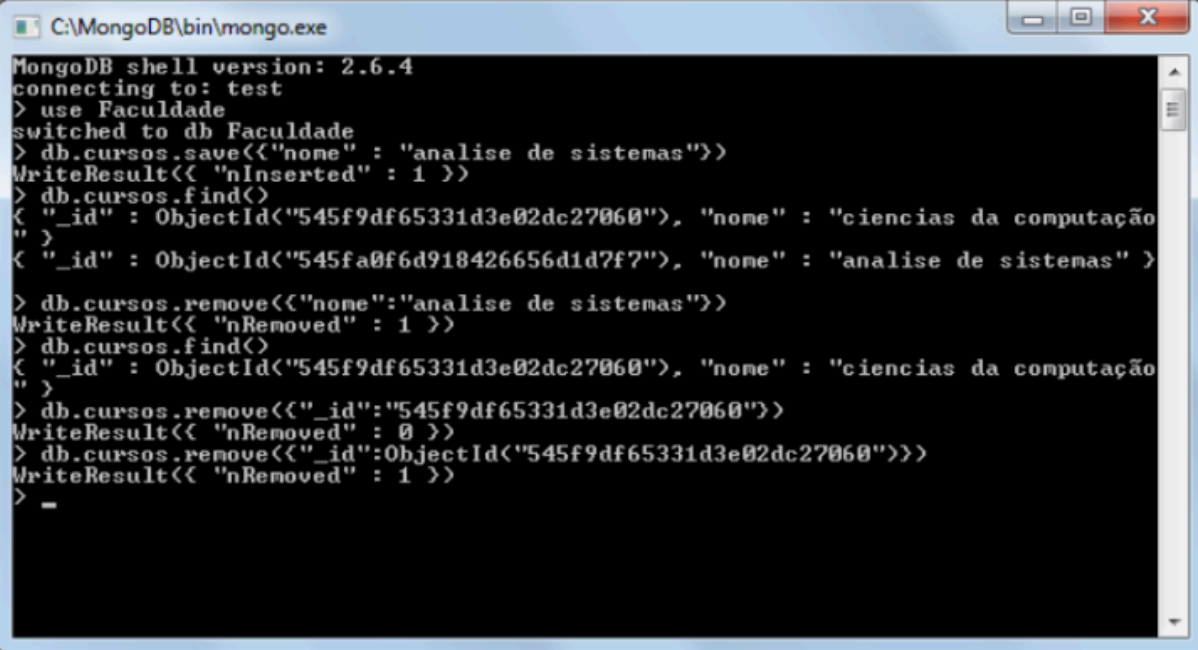
```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> db.cursos.save(<<"nome" : "analise de sistemas">>)
WriteResult<< "nInserted" : 1 >>
> db.cursos.find()
{ "_id" : ObjectId("545f9df65331d3e02dc27060"), "nome" : "ciencias da computação" }
{ "_id" : ObjectId("545fa0f6d918426656d1d7f7"), "nome" : "analise de sistemas" }
> db.cursos.remove(<<"nome":"analise de sistemas">>)
WriteResult<< "nRemoved" : 1 >>
> db.cursos.find()
{ "_id" : ObjectId("545f9df65331d3e02dc27060"), "nome" : "ciencias da computação" }
> db.cursos.remove(<<"_id":"545f9df65331d3e02dc27060">>)
WriteResult<< "nRemoved" : 0 >>
>
```

**Figura 11 - Remover documento pelo atributo ID**

Na figura 11, tentamos excluir um documento pelo seu “\_id”. Podemos observar o retorno “{“nRemoved” : 0}”, que indica que nenhum item foi excluído. Isto porque, como foi dito anteriormente, o índice “\_id” armazena um objeto contendo a



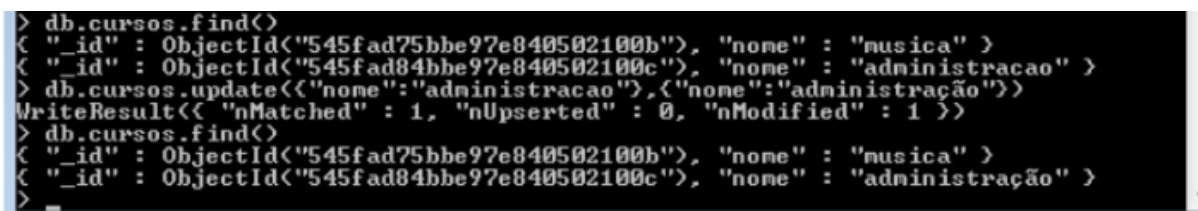
data que o registro foi inserido, portanto para excluir o registro equivalente, temos que passar um objeto como parâmetro, como demonstrado na figura 12.



```
C:\MongoDB\bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use Faculdade
switched to db Faculdade
> db.cursos.save(<<"nome" : "analise de sistemas">>)
WriteResult<< "nInserted" : 1 >>
> db.cursos.find()
< "_id" : ObjectId<"545f9df65331d3e02dc27060">, "nome" : "ciencias da computação" >
< "_id" : ObjectId<"545fa0f6d918426656d1d7f7">, "nome" : "analise de sistemas" >
> db.cursos.remove(<<"nome": "analise de sistemas">>)
WriteResult<< "nRemoved" : 1 >>
> db.cursos.find()
< "_id" : ObjectId<"545f9df65331d3e02dc27060">, "nome" : "ciencias da computação" >
> db.cursos.remove(<<"_id": "545f9df65331d3e02dc27060">>)
WriteResult<< "nRemoved" : 0 >>
> db.cursos.remove(<<"_id": ObjectId<"545f9df65331d3e02dc27060">>>)
WriteResult<< "nRemoved" : 1 >>
> =
```

**Figura 12 - Remover Atributo por “Id” - Exemplo 2**

Na figura 12 passamos um “ObjectId” como parâmetro e o retorno apontou para 1 registro excluído. Para apagar todos os registros de uma coleção, usamos o comando “db.cursos.drop()”;



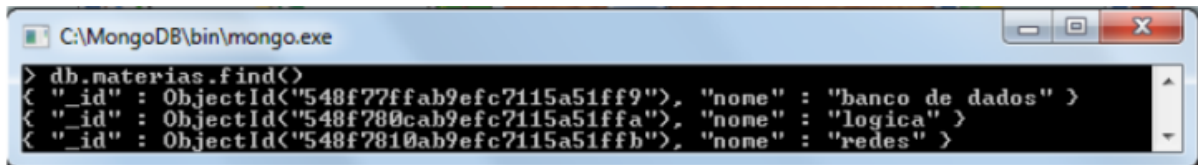
```
> db.cursos.find()
< "_id" : ObjectId<"545fad75bbe97e840502100b">, "nome" : "musica" >
< "_id" : ObjectId<"545fad84bbe97e840502100c">, "nome" : "administracao" >
> db.cursos.update(<<"nome": "administracao">>, <<"nome": "administração">>)
WriteResult<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>
> db.cursos.find()
< "_id" : ObjectId<"545fad75bbe97e840502100b">, "nome" : "musica" >
< "_id" : ObjectId<"545fad84bbe97e840502100c">, "nome" : "administração" >
> =
```

**Figura 13 - Exemplo de Atualização**

Na figura 13, inserimos dois cursos, “Música” e “Administração”. Em seguida exibimos na tela. O comando “db.cursos.update()” atualizou o registro que continha o nome “administracao” para “administração”. Nos bancos de dados NoSQL não existem relacionamentos e caso necessário fazer uma relação entre os documentos (o que sairia um pouco do contexto de estar utilizando um banco de dados não relacional), o usuário pode definir por exemplo, na coleção matéria, o índice “curso\_id” e armazenar o valor do índice “\_id” da coleção cursos, porém a consistência entre as tabelas terá de ser controlada através de programação. Em

relações “muitos para muitos” ou “many to many”, é necessário criar um novo índice em ambas as coleções que irão se relacionar, contendo um array com os valores relacionados, como mostra o exemplo a seguir.

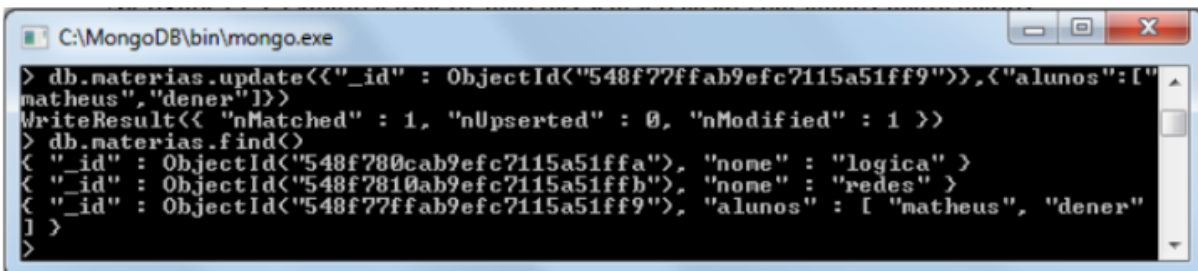
Na figura 14, é exibido a lista de matérias sem relação com alunos matriculados.



```
C:\MongoDB\bin\mongo.exe
> db.materias.find()
{ "_id" : ObjectId("548f77ffab9efc7115a51ff9"), "nome" : "banco de dados" }
{ "_id" : ObjectId("548f780cab9efc7115a51ffa"), "nome" : "logica" }
{ "_id" : ObjectId("548f7810ab9efc7115a51ffb"), "nome" : "redes" }
```

Figura 14 - Listando documentos salvos na coleção matéria

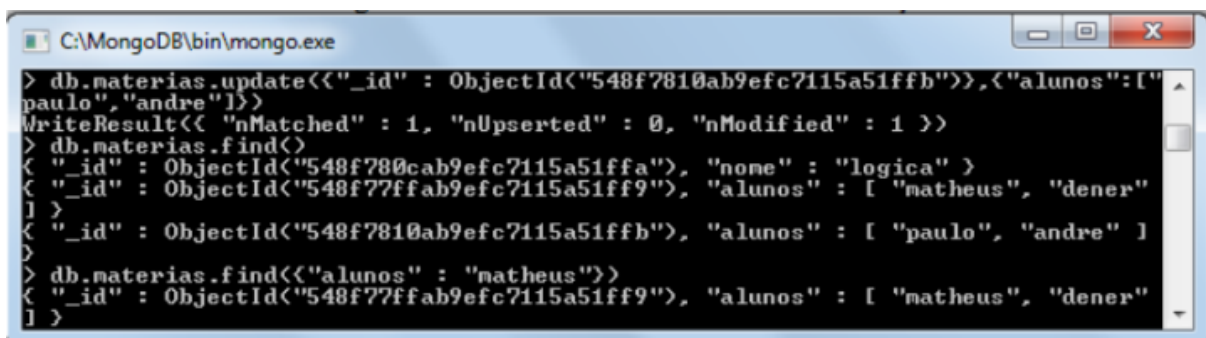
Será feito um update no documento desejado, adicionando um índice com os alunos matriculados na devida disciplina, como veremos na figura 15.



```
C:\MongoDB\bin\mongo.exe
> db.materias.update(<<"_id" : ObjectId("548f77ffab9efc7115a51ff9")>>,<<"alunos":["matheus","dener"]>>)
WriteResult(<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>)
> db.materias.find()
{ "_id" : ObjectId("548f780cab9efc7115a51ffa"), "nome" : "logica" }
{ "_id" : ObjectId("548f7810ab9efc7115a51ffb"), "nome" : "redes" }
{ "_id" : ObjectId("548f77ffab9efc7115a51ff9"), "alunos" : [ "matheus", "dener" ] }
```

Figura 15 - Criando "relação" de matérias com alunos

Na figura 16, temos um exemplo de consulta neste índice que contém os alunos relacionados.



```
C:\MongoDB\bin\mongo.exe
> db.materias.update(<<"_id" : ObjectId("548f7810ab9efc7115a51ffb")>>,<<"alunos":["paulo","andre"]>>)
WriteResult(<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>)
> db.materias.find()
{ "_id" : ObjectId("548f780cab9efc7115a51ffa"), "nome" : "logica" }
{ "_id" : ObjectId("548f77ffab9efc7115a51ff9"), "alunos" : [ "matheus", "dener" ] }
{ "_id" : ObjectId("548f7810ab9efc7115a51ffb"), "alunos" : [ "paulo", "andre" ] }
> db.materias.find(<<"alunos" : "matheus">>)
{ "_id" : ObjectId("548f77ffab9efc7115a51ff9"), "alunos" : [ "matheus", "dener" ] }
```

Figura 16 - Consulta no índice formado por um array

No exemplo acima, foi realizado um update em outro documento da coleção de matérias e exibido os dados logo a seguir com o comando “db.materias.find()”. Para consultar, por exemplo, as matérias em que o aluno “Matheus” está

matriculado, foi usado o comando `“db.materias.find( {“alunos” : “matheus”} )”`.

Nos exemplos da figura 15 e da figura 16, fizemos consultas na coleção de matérias. Para que as mesmas consultas pudessem ser executadas na coleção dos alunos, seria necessário criar o índice “matérias” na coleção dos alunos, assim como fizemos na coleção de matérias para relacionar os alunos.

Existem muitos outros comandos para consultas parecidos com comandos utilizados em SQL e podem ser encontrados no manual on-line do MongoDB.

#### **4. CONCLUSÃO:**

A crescente complexidade e os elevados volumes de dados gerados por aplicações modernas, como big data, Internet das Coisas (IoT) e aplicativos móveis, têm exposto as limitações dos bancos de dados relacionais tradicionais, especialmente no que diz respeito à flexibilidade, escalabilidade e desempenho.

Para compreender como essa nova era gerou um volume tão grande de dados basta pensar em como as redes sociais estão presentes na vidas das pessoas hoje em dia nas milhares de fotos que são tiradas diariamente nas que são postadas e nas que não são mas ainda assim são dados que são gerenciados muitas vezes em nuvem em aplicações como o google fotos.

Nesse contexto, os bancos de dados NoSQL, com destaque para o MongoDB, apresentam-se como uma alternativa viável e eficiente para atender às demandas de armazenamento e manipulação de dados em cenários dinâmicos.

O MongoDB, como um dos principais representantes do modelo documental, se destaca por sua estrutura orientada a documentos, suporte à escalabilidade horizontal e integração com diversas linguagens de programação por meio de drivers oficiais. Suas capacidades, como a utilização de JSON/BSON para armazenamento de dados, e recursos avançados como sharding, o tornam uma solução robusta para aplicações que exigem alta performance e adaptabilidade a cargas de trabalho variáveis. A sua fácil instalação e integração com diversas aplicações o tornaram um dos sistemas mais populares atualmente.

Ao longo do trabalho, foi possível demonstrar como o MongoDB supera desafios encontrados em ambientes que lidam com dados não estruturados ou semiestruturados, oferecendo respostas rápidas e organização eficiente. Conclui-se, portanto, que o MongoDB não apenas complementa, mas em muitos casos, substitui com eficácia os bancos relacionais, principalmente em contextos que demandam maior flexibilidade, como aplicações voltadas a IoT, big data e mobilidade.

O estudo reforça que a escolha entre bancos relacionais e NoSQL deve considerar os requisitos específicos do sistema, pois, embora os bancos NoSQL como o MongoDB ofereçam diversas vantagens, os bancos relacionais ainda possuem sua relevância em aplicações com dados altamente estruturados e que exigem transações complexas. Dessa forma, a adoção do MongoDB em projetos futuros pode ser uma decisão estratégica para organizações que buscam inovação e desempenho em suas soluções de tecnologia.

## REFERÊNCIAS

Banker, Kyle. **MongoDB in Action**. Shelter Island: Manning, 2012.

Chodorow, Kristina; Michael Dirolf. **MongoDB The Definitive Guide**, First Edition. Gravenstein Highway North: O'Reilly, 2010.

**Documentação MongoDB e Tutorial para implementação em linguagem Java.** Disponível em: <<http://www.mongodb.org/display/DOCS/Java+Language+Center>>. Acesso em: 25 de novembro de 2024.

BARASUOL. **MongoDB Uma base de Dados Orientadas as Documentada que Utiliza Orientação a Objetos**: Trabalho de Conclusão de Curso. 101 f. Fundação Educacional de Assis, Campos José Santil Sobrinho.

MongoDB (2024). MongoDB: CRUD Operations [on-line]. Disponível em “<http://docs.mongodb.org/master/MongoDB-crud-guide.pdf>”.

Tiwari, Shashank. **Professional NoSQL**. Indianapolis: John Wiley & Sons, 2011.