

Mongodb

Big Data - Sistemas de Informação

Alan Victor Matos Cerqueira - 01536516
Amanda Gabriela Bernardo da Costa - 01566793
Breno Antônio Alexandrino da Silva - 01280628
Cibele Benício Lourenço - 01535470
Clóvis Vitor dos Santos Tavares - 01564866
Maria Christina Santos Barbosa - 01565538

Recife, Setembro de 2024.

Sumário

Introdução.....	2
Banco de dados.....	2
Os principais tipos de bancos de dados NoSQL:.....	2
O MongoDB.....	3
Ferramentas de MongoDB.....	3
1. RoboMongo (Robo 3T).....	3
2. Mongo Express.....	4
3. Atlas MongoDB.....	4
Mongo Atlas.....	5
1. Criar uma conta no MongoDB Atlas.....	5
2. Criar um cluster.....	5
3. Configurar a segurança.....	6
4. Conectar-se ao cluster.....	8
O que é CRUD.....	8
Fazendo um CRUD no MongoDB.....	9
Ferramentas utilizadas.....	9
Vantagens dessa Abordagem.....	9
1. Estabelecendo a Conexão com o MongoDB.....	9
2. Escolhendo uma Base de Dados e uma Coleção.....	10
3. Create.....	11
4. Read.....	12
5. Update.....	13
6. Delete.....	14
Conclusão.....	15

Introdução

Banco de dados

Um banco de dados é uma coleção organizada de informações – ou dados – estruturadas, normalmente armazenadas eletronicamente em um sistema de computador.

As aplicações processam um grande volume de dados de fontes diferentes, como mídias sociais, sensores inteligentes e bancos de dados de terceiros. Todos esses dados diferentes não se encaixam perfeitamente no modelo relacional. A aplicação de estruturas de tabela pode levar a problemas de redundância, duplicação de dados e desempenho em grande escala.

Os bancos de dados NoSQL (Not Only SQL) são desenvolvidos especificamente para modelos de dados não relacionais e possuem esquemas flexíveis para a construção de aplicações modernas. Eles são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala.

Existem quatro tipos principais de bancos de dados NoSQL: chave-valor, documento, coluna e grafo.

Os principais tipos de bancos de dados NoSQL:

Chave-valor

Ele armazena dados como um conjunto de pares de chave-valor em que uma chave atua como um identificador exclusivo. Tanto as chaves quanto os valores podem ser qualquer coisa, desde objetos simples até objetos compostos complexos.

Você pode pensar na chave como uma pergunta e no valor como a resposta à pergunta.

Documento

Também conhecido como armazenamento por documentos, os bancos de dados orientados a documentos são um modelo de banco de dados projetado para gerenciar, armazenar e recuperar informações orientadas a documentos, não carecendo de colunas e é um modelo eficiente para o trabalho com dados não estruturados.

O MongoDB

À medida que as aplicações modernas continuam a crescer em complexidade e escala, as pessoas desenvolvedoras enfrentam o desafio de encontrar soluções de banco de dados que ofereçam flexibilidade, desempenho e escalabilidade sem comprometer a integridade dos dados.

Com o objetivo de solucionar esse desafio, surge o *MongoDB* desenvolvido pela empresa MongoDB Inc.

Afinal, o que “ele” faz?

O MongoDB é um sistema de gerenciamento de banco de dados NoSQL (Not Only SQL) de código aberto e orientado a documentos.

Em vez de seguir o modelo de banco de dados relacional tradicional, que organiza os dados em tabelas com esquemas predefinidos, o MongoDB adota um modelo de dados flexível baseado em documentos.

Assim, o MongoDB armazena os dados no formato BSON (Binary JSON), que permite estruturas de dados hierárquicos e não requer um esquema fixo.

Ferramentas de MongoDB

As ferramentas **RoboMongo**, **Mongo Express** e **MongoDB Atlas** são amplamente utilizadas para interagir com o MongoDB, cada uma possuindo propósitos e funcionalidades distintas:

1. RoboMongo (Robo 3T)

O **RoboMongo**, atualmente conhecido como **Robo 3T**, facilita a administração de bases de dados MongoDB oferecendo uma interface gráfica, ela substitui a necessidade de uso do Shell do Linux ou CMD do Windows, permitindo realizar operações CRUD e visualizar dados em formato JSON. Ele é utilizado em ambientes locais (desktop), sendo instalado no computador do usuário, e possibilita o gerenciamento de bases de dados MongoDB tanto locais quanto remotas. O software simplifica a visualização e manipulação dos dados sem a necessidade de comandos no terminal.

2. Mongo Express

O **Mongo Express** é uma ferramenta baseada na **web**, ela atua como um painel de controle para o MongoDB, proporcionando assim uma interface acessível pelo navegador para a gestão dos dados, diretamente do servidor. A ferramenta oferece transações CRUD, possibilitando a criação e exclusão de coleções, edição de documentos e execução de consultas, de maneira leve e simples para administra-lo através da web. Quando o MongoDB está rodando em um servidor remoto, como um servidor web, o Mongo Express pode ser utilizado para gerenciar o banco de dados diretamente pelo navegador, eliminando a necessidade de acessar o servidor via terminal.

3. Atlas MongoDB

O **MongoDB Atlas** é o serviço de MongoDB gerenciado na nuvem, oferecido pela empresa MongoDB. Ele oferece facilidade de uso, segurança, escalabilidade e alta disponibilidade sem a necessidade de configurar infraestrutura. Além disso, automatiza tarefas complexas como backups e segurança, permitindo a criação e gerenciamento de bases de dados na nuvem, acessíveis de qualquer lugar. É ideal para aplicações que demandam armazenamento de grandes volumes de dados, sem a necessidade de instalação ou manutenção de servidores.

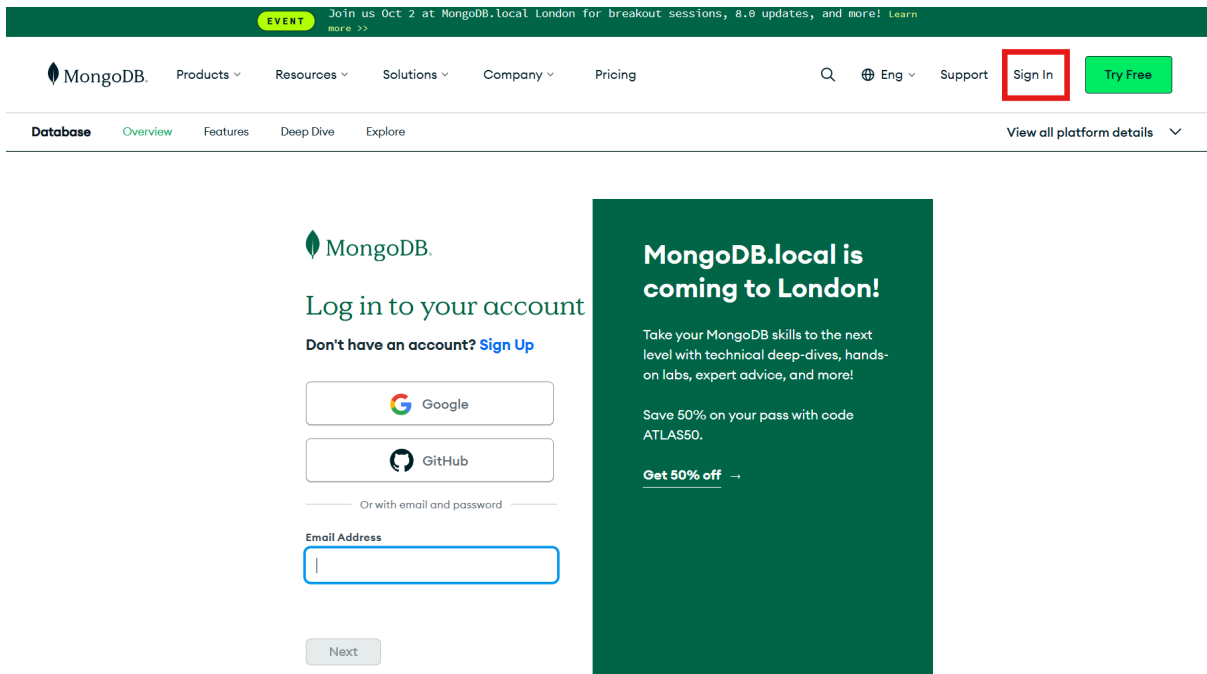
Em resumo, enquanto o RoboMongo e o Mongo Express são soluções mais externas para gerenciamento local e web do MongoDB, respectivamente, o MongoDB Atlas se destaca por ser uma solução completa e gerenciada na nuvem, ideal para quem busca praticidade e escalabilidade.

Mongo Atlas

Com o MongoDB como banco de dados e o Mongo Atlas escolhido como nossa ferramenta, o primeiro passo é instalar e configurar.

1. Criar uma conta no MongoDB Atlas

- Acesse o site oficial [MongoDB Atlas](https://www.mongodb.com/atlas).
- Clique em "Iniciar gratuitamente" ou "Sign up" e crie sua conta com seu e-mail ou usando o login do Google ou GitHub.



2. Criar um cluster

- Após criar a conta e fazer login, clique em "Build a Cluster".
- Escolha um provedor de nuvem (AWS, Google Cloud, ou Azure), selecione a região do servidor mais próxima de você ou de seu usuário.
- Escolha a configuração gratuita (Shared Cluster) ou um plano pago, dependendo de suas necessidades.
- Clique em "Create Cluster". O processo pode levar alguns minutos.

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

☐ **M10** **\$0.12/hour**
For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

☐ **Serverless**
For application development and testing, or workloads with variable traffic.

STORAGE	RAM	vCPU
Up to 1 TB	Auto-scale	Auto-scale

☒ **M0** **Free**
For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

✔ **Free forever!** Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name

You cannot change the name once the cluster is created.

I'll do this later

Go to Advanced Configuration

Create Deployment

Name

You cannot change the name once the cluster is created.

☐ Preload sample dataset ⓘ

Provider



Region

Sao Paulo (sa-east-1) ★

★ Recommended ⓘ

Low carbon emissions ⓘ

Tag (optional)

Create your first tag to categorize and label your resources; more tags can be added later. [Learn more.](#)

3. Configurar a segurança

- **Criar um usuário do banco de dados:** Em "Database Access", adicione um novo usuário com um nome de usuário e senha. Defina as permissões adequadas (admin, read-only, etc.).
- **Configurar IP Whitelisting:** Vá para "Network Access" e adicione o IP do seu computador ou de outros sistemas que terão acesso ao banco de dados. Você pode permitir acesso de qualquer lugar com "Allow Access From Anywhere", mas para mais segurança, especifique apenas os IPs confiáveis.

Connect to Cluster0



You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

✓ Your current IP address (190.89.232.202) has been added to enable local connectivity. Add another later in [Network Access](#).

2. Create a database user

This first user will have [atlasAdmin](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

❗ You'll need your database user's credentials in the next step. Copy the database user password.

Username

ex. dbUser

Password


ex. dbUserPassword

HIDE

Copy

Create Database User

Connect to Cluster0



You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

✓ Your current IP address (190.89.232.202) has been added to enable local connectivity. Add another later in [Network Access](#).

2. Create a database user


✓ A database user has been added to this project. Create another user later in [Database Access](#).

You'll need your database user's credentials in the next step.

Close

Choose a connection method

Connect to Cluster0



Connect to your application

Drivers

Access your Atlas data using MongoDB's native drivers (e.g., Node.js, Go, etc.)

Access your data through tools

Compass

Explore, modify, and visualize your data with MongoDB's GUI

Shell

Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code

Work with your data in MongoDB directly from your VS Code environment

Atlas SQL

Easily connect SQL tools to Atlas for data analysis and visualization

Go Back

Close

4. Conectar-se ao cluster

- Quando o cluster estiver pronto, clique em "Connect".
 - O Atlas oferece várias opções de conexão:
 - **MongoDB Compass:** Se quiser usar uma interface gráfica, você pode baixar o MongoDB Compass e conectar ao cluster com o URI fornecido.
 - **Aplicações:** Para conectar sua aplicação, escolha a linguagem (Node.js, Python, etc.) e copie o URI de conexão fornecido, substituindo `<password>` pela senha criada.
 - **Shell:** Se preferir usar o terminal, siga as instruções para conectar-se usando o MongoDB Shell (comando `mongo`).
-

O que é CRUD

CRUD é uma sigla que representa as quatro operações mais importantes que podemos fazer com dados em um banco de dados:

- **Criar (Criar)** : Adicionar um novo dado.
 - **Leia (Ler)** : Consultar os dados que estão armazenados.
 - **Atualizar (Atualizar)** : Modificar informações que já estão no banco de dados.
 - **Excluir (Excluir)** : Remove informações.
-

Fazendo um CRUD no MongoDB

Ferramentas utilizadas

Utilizamos o **Google Colab** e o **MongoDB Atlas** que oferecem uma solução prática e acessível para lidar com dados em nuvem sem sobrecarregar a máquina local, sendo ideal para quem quer evitar instalações locais e ainda trabalhar de forma profissional com dados.

Vantagens dessa Abordagem

- **Não precisa de instalação local:** Usando o Colab, você evita a necessidade de instalar Python, MongoDB ou qualquer outro software localmente.
- **Acesso de qualquer lugar:** Como o MongoDB Atlas está na nuvem e o Colab também, você pode acessar e manipular seus dados de qualquer dispositivo com internet.

1. Estabelecendo a Conexão com o MongoDB

O primeiro passo no processo de interação com o MongoDB é estabelecer uma conexão entre o programa e o banco de dados, onde serão armazenadas as informações. Para isso, utilizamos uma biblioteca chamada **PyMongo**, que permite a comunicação entre o código Python e o MongoDB.

```
import pymongo
```

```
# Conectando ao MongoDB com as credenciais fornecidas
```

```
mongodb_client =
```

```
pymongo.MongoClient('mongodb+srv://alanvictor:12345@cluster0.75zye.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')
```

- `import pymongo`: Carrega a biblioteca **PyMongo**, que contém todos os comandos necessários para interagir com o MongoDB.
- `mongodb_client = pymongo.MongoClient(...)`: Essa linha estabelece a conexão entre o programa Python e o servidor MongoDB, localizado no endereço `cluster0.75zye.mongodb.net`. A URL de conexão contém as seguintes informações:
 - **Nome de usuário**: alanvictor
 - **Senha**: 12345
 - **Endereço do servidor**: cluster0.75zye.mongodb.net

2. Escolhendo uma Base de Dados e uma Coleção

Após conectar-se ao MongoDB, o próximo passo é definir em qual base de dados e coleção as informações serão armazenadas. O MongoDB organiza os dados em estruturas chamadas "bases de dados" e "coleções", que podem ser comparadas a pastas e arquivos.

No exemplo abaixo, escolhemos uma base de dados chamada `medicos` e uma coleção também indicada `medicos`, que armazenará as informações de cada médico.

```
# Escolhendo a base de dados 'medicos'  
database = mongodb_client['medicos']
```

```
# Escolhendo a coleção 'medicos'  
collection = database['medicos']
```

- `database = mongodb_client['medicos']`: Aqui, estamos acessando a chamada "pasta" `medicos`, ou seja, uma base de dados que será utilizada.
 - `collection = database['medicos']`: Seleccionamos uma "coleção" dentro da base de dados `medicos`, onde serão armazenadas as informações, como nome, especialidade, CRM, entre outros detalhes dos médicos.
-

3. Create

Com a conexão estabelecida e a coleção selecionada, podemos agora inserir as informações de um novo médico no banco de dados. Criamos um "documento", que representa os dados do médico, e então o salvamos na coleção.

```
# Criando um documento para o médico
medico = {
    'nome': 'Dr. Jose Alirio',
    'especializacao': 'Cardiologia',
    'crm': 'SP123456',
    'anos_experiencia': 10,
    'hospital': 'Hospital das Clínicas',
    'disponibilidade': ['Segunda', 'Quarta', 'Sexta']
}

# Inserindo o documento na coleção
result = collection.insert_one(medico)

# Verificando se o médico foi inserido com sucesso
if result.acknowledged:
    print(f'Médico {medico["nome"]} inserido com sucesso, id {result.inserted_id}')
```

Aqui, o que acontece:

- `medico = {...}`: Criamos um "documento", que é uma estrutura de dados contendo todas as informações do médico, incluindo nome, especialização, CRM, anos de experiência, hospital, e dias de disponibilidade.
- `result = collection.insert_one(medico)`: Este comando insere o documento do médico na coleção `medicos`.
- `result.acknowledged`: Verifica se o MongoDB armazenou o documento com sucesso. Se tudo estiver correto, o programa imprime uma mensagem de sucesso e exibe o ID gerado para o médico.

4. Read

Após inserir médicos na base de dados, pode ser necessário consultar todos os médicos cadastrados. Para isso, usamos o método **find()** , que busca todos os documentos presentes na coleção.

```
def buscar_medicos():  
    try:  
        medicos = collection.find()  
        for medico in medicos:  
            print(medico)  
    except Exception as e:  
        print(f'Um erro ocorreu: {e}')
```

Aqui está o que acontece:

- `collection.find()`: Retorna todos os documentos (médicos) armazenados na coleção `medicos`.
 - `for medico in medicos`: Aqui, percorremos a lista de médicos retornada pelo comando anterior e imprimimos as informações de cada médico.
 - Bloco **try-except**: Serve para capturar qualquer erro que possa acontecer durante a execução e imprimir uma mensagem de erro, se necessário.
-

5. Update

Para atualizar as informações de um médico, por exemplo, sua disponibilidade ou o hospital em que trabalha, use o comando **update_one()** .

```
def atualizar_medico(medico):  
    try:  
        result = collection.update_one({'_id': medico['_id']},  
{'$set': medico})  
        if result.modified_count:  
            print(f'Médico {medico["nome"]} atualizado com  
sucesso')  
        else:  
            print('Falha ao atualizar médico')  
    except Exception as e:  
        print(f'Um erro ocorreu: {e}')
```

- **collection.update_one(..., {'\$set': medico})**: Aqui, estamos dizendo para o MongoDB atualizar o médico com base no identificador único (**_id**) que o MongoDB gera automaticamente. O **'\$set': medico** indica que queremos atualizar as informações do médico com os novos dados fornecidos.
 - **result.modified_count**: Verifica se algum dado foi de fato atualizado.
 - Se o médico for atualizado corretamente, uma mensagem será exibida confirmando a atualização.
-

6. Delete

Se um médico não estiver mais ativo, podemos remover suas informações do banco de dados usando o comando **delete_one()** .

```
def remover_medico(medico):  
    try:  
        result = collection.delete_one({'_id': medico['_id']})  
        if result.deleted_count:  
            print(f'Médico {medico["nome"]} removido com  
sucesso')  
        else:  
            print('Falha ao remover médico')  
    except Exception as e:  
        print(f'Um erro ocorreu: {e}')
```

- **collection.delete_one({'_id': medico['_id']})**: Isso remove o médico do banco de dados usando o identificador único (**_id**) que o MongoDB atribui a cada documento.
 - **result.deleted_count**: Verifica se algum registro foi removido.
 - Se o médico for removido com sucesso, o programa exibirá uma mensagem confirmando a remoção.
-

Conclusão

Os bancos de dados NoSQL, como o MongoDB, desempenham um papel crucial no armazenamento e gerenciamento de grandes volumes de dados não estruturados, atendendo às necessidades das aplicações que requerem flexibilidade, desempenho e escalabilidade. O MongoDB se destaca como uma solução versátil, oferecendo um modelo de dados orientado a documentos que permite armazenar informações complexas sem o estresse dos bancos relacionais.

Além disso, ferramentas como RoboMongo, Mongo Express e MongoDB Atlas facilitam a administração do banco de dados, proporcionando interfaces intuitivas e acessíveis tanto localmente quanto na nuvem. A implementação de operações CRUD no MongoDB exemplifica a simplicidade e eficiência com que dados são criados, consultados, atualizados e excluídos, permitindo que os desenvolvedores otimizem suas aplicações com facilidade.

Portanto, o MongoDB surge como uma solução prática e robusta para o cenário de big data, onde a escalabilidade e o desempenho são essenciais para o sucesso de qualquer aplicação.