



# BANCO DE DADOS NOSQL

AULA 3



Prof. Alex Mateus Porn

## CONVERSA INICIAL

Olá! Nesta aula estudaremos sobre bancos de dados NoSQL orientados a documentos. Nosso objetivo é caracterizar os principais conceitos sobre o modo de armazenamento e gerenciamento de dados NoSQL orientado a documentos e estudar por meio de exemplos práticos uma ferramenta para criação e gerenciamento desses bancos de dados.

Iniciaremos analisando como são estruturados os bancos de dados orientados a documentos, o formato de um documento e como é o processo de modelagem nesse tipo de banco de dados. O objetivo aqui é compreender a estrutura dos documentos desse tipo de banco de dados para facilitar o entendimento dos exemplos práticos que abordaremos na sequência.

Como ferramenta para criação e gerenciamento dos bancos de dados orientados a documentos, estudaremos a ferramenta MongoDB, que atualmente é uma das mais conhecidas e utilizadas pelos profissionais da área. A aula será finalizada com a apresentação das principais aplicações dos bancos de dados NoSQL orientados a documentos. Vejamos os conteúdos que serão trabalhados:



### TEMA 1 – SISTEMAS NOSQL BASEADOS EM DOCUMENTOS

Os bancos de dados NoSQL orientados a documentos podem ser considerados como os mais populares atualmente entre as quatro categorias existentes. Para Hurwitz (2016, p. 91) podem ser encontrados dois tipos de



bancos de dados orientados a documentos, os já conhecidos repositórios para conteúdo em estilo de documento completo, como arquivos editores de texto, planilhas eletrônicas, páginas *web*, entre outros e, as bases de dados para armazenar componentes de documentos ou para um conjunto dinâmico de suas partes, as quais abordaremos nesta aula. Conforme Marquesone (2017, p. 47), esse modelo de banco de dados orientado a documentos é considerado uma extensão do banco de dados NoSQL orientado a chave-valor, oferecendo simplicidade e flexibilidade no gerenciamento dos dados e enriquecendo as possibilidades de consultas.

Para Hurwitz (2016, p. 91), bases de dados documentais são muito úteis quando é necessário produzir relatórios que precisam ser montados dinamicamente a partir de elementos que mudam com frequência. Um exemplo é o preenchimento de um documento da área da saúde, em que a composição do conteúdo varia com base no perfil do paciente, como idade, residência, renda, plano de saúde, elegibilidade para programas de governo.

Ao contrário dos bancos de dados relacionais que aplicamos o processo de normalização de dados, para evitar que os dados tenham valores duplicados nos bancos de dados NoSQL orientados a documentos esse processo é desconsiderado, assim como a criação de *joins* e esquemas. Conforme Marquesone (2017, p. 47) e Hurwitz (2016, p. 92), os documentos de um banco de dados orientado a documentos, podem ser definidos como estruturas flexíveis que podem ser obtidas por meio de dados semiestruturados, como os formatos XML (*eXtensible Markup Language*), JSON (*JavaScript Object Notation*) e BSON (*Binary JSON*).

Para nos aproximarmos um pouco mais da compreensão da estrutura de um banco de dados orientado a documentos, imaginemos um documento como sendo uma linha da tabela, e um conjunto de documentos como sendo a tabela com todos os registros. A diferença é que cada documento pode conter variações em sua estrutura (Marquesone, 2017, p. 47). Para compreender melhor esse formato, vamos analisar o exemplo apresentado na Figura 1, que destaca um documento no formato JSON para armazenamento de informações de um cadastro de clientes.



Figura 1 – Documento de cadastro de clientes no formato JSON

```
1 {  
2   "clientes": [  
3     {  
4       "nome": "Cliente X",  
5       "dataNascimento": "25/03/1985",  
6       "endereco": "Rua das Avenidas, 290",  
7       "telefone": "(42) 3542-9898",  
8       "celular": "(42)-99999-9898"  
9     },  
10    {  
11      "nome": "Cliente Y",  
12      "nascimento": {  
13        "dia": 23,  
14        "mes": 08,  
15        "ano": 1985  
16      },  
17      "endereco": "Avenida das Vielas, 275",  
18      "telefone": "(42) 3542-9999",  
19      "contato": {  
20        "celular": "(42) 98888-8989",  
21        "email": "clientey@xxyy.com"  
22      }  
23    }  
24  ]  
25 }
```

Conforme o exemplo, existem dois registros armazenados no documento “clientes”. Enquanto o cliente de nome “Cliente X” possui o atributo “dataNascimento” para armazenar a data completa, o cliente de nome “Cliente Y” possui o atributo “nascimento”, contendo uma lista dos atributos “dia”, “mes” e “ano”, para que os dados da data de nascimento sejam armazenados separadamente. Ambos os registros apresentados na Figura 1 são referentes aos dados dos clientes, porém é possível que cada registro contenha informações diferentes. A mesma situação ocorre para o atributo “celular” do registro do “Cliente X”, sendo que no registro do “Cliente Y”, esse atributo pertence a uma lista de atributos que constituem o atributo “contato”.

Diferentemente dos bancos de dados NoSQL orientados a chave-valor, que somente permitem que as consultas sejam realizadas pelos campos-chaves, de acordo com Marquesone (2017, p. 48) os bancos de dados NoSQL orientados a documentos permitem a criação de consultas e filtros sobre os valores armazenados. Assim, ainda seguindo o exemplo apresentado na Figura 1, podemos criar várias consultas, como por exemplo, listar os clientes por nome, endereço ou data de nascimento.

Desse modo, Marquesone (2017, p. 48-49), define que:

Bancos de dados orientados a documentos são ótimas soluções para armazenamento de registros conter todas as informações relevantes para uma consulta, sem necessitar da criação de joins.com atributos variados. Além disso, esses bancos de dados oferecem grande escalabilidade e velocidade de leitura, pois os dados são



armazenados em forma desnormalizada. Por esse motivo, um documento armazenado deve conter todas as informações relevantes para uma consulta, sem necessitar da criação de joins.

Seguindo essa abordagem, Medeiros (2014) menciona que os bancos de dados orientados a documentos têm como características conter todas as informações importantes em um único documento, ser livre de esquemas, possuir identificadores únicos universais, possibilitar a consulta de documentos através de métodos avançados de agrupamento e filtragem (*MapReduce*) e também permitir redundância e inconsistência. Ainda conforme o autor, ao contrário dos bancos de dados relacionais, os bancos de dados orientados a documentos não fornecem relacionamentos entre os documentos, o que mantém seu *design* sem esquemas. Dessa forma, ao invés de armazenar os dados relacionados em uma área de armazenamento separado, os bancos de dados de documentos integram esses dados ao próprio documento. O Quadro 1 apresenta uma breve relação entre o modelo de dados relacional e os bancos de dados orientados a documentos.

Quadro 1 – Relação entre o modelo relacional e orientado a documentos

Modelo Relacional	Orientado a Documentos
• Esquema	• Não Existe
• Tabelas	• Documentos
• Chave Estrangeira	• Não Existe
• Relacionamentos	• Não Existe
• Linhas	• Registros
• Colunas	• Atributos

De acordo com o exemplo do Quadro 1, caso fossemos representar o documento de clientes da Figura 1 em uma tabela no modelo relacional, primeiramente seria necessário especificar um esquema para essa tabela, visto que existem diferenças na representação de alguns atributos para cada registro, como no caso do atributo “dataNascimento”. Supondo que o esquema



definido seja conforme a representação de atributos utilizada para o “Cliente X”, ou seja:

- nome : String
- dataNascimento : Date
- endereco : String
- telefone : String
- celular : String

Havendo a necessidade de acrescentar mais colunas à tabela, torna-se necessária a alteração da própria tabela, pois as novas colunas serão adicionadas para todos os registros existentes. A esses registros serão atribuídos o valor nulo. Esse caso pode ser representado ao ser necessário adicionar o atributo “email”, conforme elencado no registro “Cliente Y” da Figura 1. Assim, ao registro “Cliente X” também será atribuído esse atributo, porém com valor nulo. A Figura 2 apresenta o registro de clientes da Figura 1 no modelo relacional.

Figura 2 – Representação de um documento no modelo relacional

	nome	dataNascimento	endereco	telefone	celular	email
	Cliente X	25/03/85	Rua das Avenidas, 290	(42) 3542-9898	(42) 99999-9898	
	Cliente Y	23/08/85	Avenida das Vielas, 275	(42) 3542-9999	(42) 98888-8989	clientey@xxyy.com

Para Lennon (2011), manter os dados integrados ao próprio documento ao invés de armazená-los em uma área de armazenamento separada, funciona muito bem para aplicativos nos quais os dados ficam autocontidos dentro de um documento pai. Como exemplo podem ser citados os posts e comentários em blogs. Os comentários se aplicam somente a um único post, de modo que não faz sentido separá-los dele. Conforme o exemplo proposto pelo autor, no documento dos “posts de um blog” teria um atributo “nome” para registrar o nome do post e outro atributo “comentario” para armazenar os comentários referentes a cada post. Já em um banco de dados relacional, haveria uma tabela para armazenar os comentários, com uma chave primária, uma tabela de *posts* com uma chave primária e, uma tabela de mapeamento intermediária de *posts* e comentários para definir quais comentários pertencem a quais posts. A Figura 3 apresenta esse exemplo de modo gráfico.



Figura 3 – Representação do modelo de dados de documento e relacional

### Documento de Posts e Comentários

```
1 {  
2   "Post": [  
3     {  
4       "nome": "Postagem X",  
5       "comentarios": ["Comentário 1", "Comentário 2"]  
6     },  
7     {  
8       "nome": "Postagem Y",  
9       "comentarios": ["Comentário 1", "Comentário 2"]  
10    }  
11  ]  
12 }
```

### Tabelas de Posts e Comentários



## 1.1 Modelagem de dados orientada a documentos

De acordo com Monteiro (2019), o objetivo de modelar um banco de dados orientado a documentos é minimizar a quantidade de acessos ao banco de dados, para que a aplicação se torne mais rápida. À medida que a quantidade de dados aumenta podemos ter um conjunto de documentos que dará origem a uma coleção e, desse modo, um conjunto de coleções forma o banco de dados. Portanto, basicamente podemos ter dois formatos de modelagem, com documentos referenciados ou incorporados.

- **Modelagem incorporada (embedded):** a modelagem incorporada refere-se a uma estrutura não normalizada, em que os dados normalmente são acessados juntos, como se fosse um documento dentro do outro (Monteiro, 2019).
- **Modelagem referenciada:** a modelagem referenciada estabelece em ter os documentos separados, mas um dos documentos tem a referência para o outro, de modo a minimizar a quantidade de dados duplicados, aumentar o desempenho nas operações de escrita, porém diminuir o desempenho nas consultas (Monteiro, 2019).

Na modelagem incorporada, a operação principal são as leituras, o que não impede de ser usada em aplicações com um grande volume de escritas.



Para Monteiro (2019), no caso de escritas que envolvem muitas alterações nos dados, é preciso tomar cuidado, porque a informação que está sendo alterada pode estar replicada em vários documentos. Assim, a atualização dessa informação pode precisar ser feita em muitos lugares, sob pena de ter dados inconsistentes. A Figura 4 apresenta um exemplo da modelagem de dados orientada a documentos incorporada.

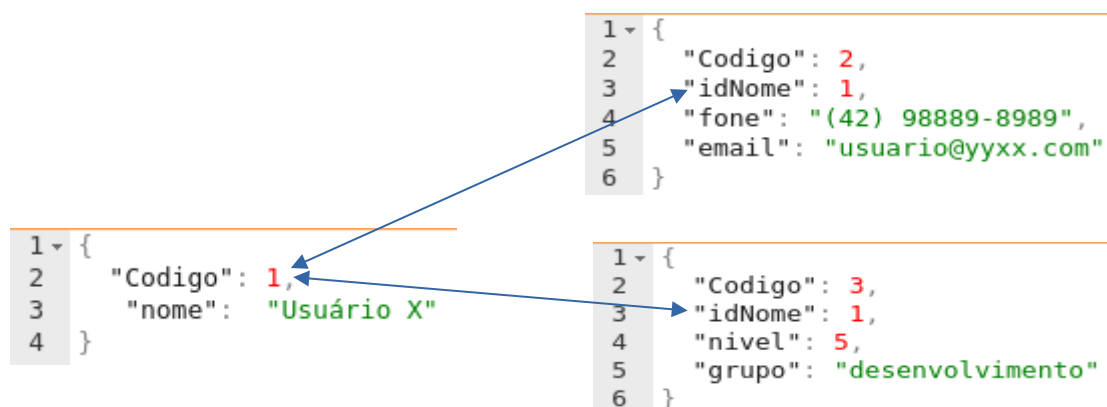
Figura 4 – Modelagem de dados orientada a documentos incorporada



Fonte: Adaptado de Monteiro, 2019.

Já na modelagem referenciada os documentos são armazenados separadamente, fazendo referência à abordagem do modelo relacional. Essa modelagem apresenta maior eficiência quando parte do documento é frequentemente lido/atualizado e a outra parte não, quando os dados não devem ser duplicados e quando um objeto é referenciado em muitos outros. A Figura 5 apresenta um exemplo da modelagem de dados orientada a documentos referenciada.

Figura 5 – Modelagem de dados orientada a documentos referenciada



Fonte: Adaptado de Monteiro, 2019.



## TEMA 2 – MONGODB

O MongoDB, conforme Hows, Membrey e Plugge (2015, p. 16), é um banco de dados que não tem conceitos de tabelas, esquemas, SQL ou linhas. Não há transações, conformidade com as propriedades ACID de bancos de dados relacionais, *joins* ou chaves estrangeiras. Conforme Lennon (2011), trata-se de um banco de dados orientado a documentos e de software livre que armazena dados em coleções de documentos BSON, um formato semelhante ao JSON, ou seja, uma versão binária do formato JSON.

De acordo com Hows, Membrey e Plugge (2015, p. 72), apesar de o MongoDB ser conhecido como um banco de dados sem esquemas, isso não significa que sua estrutura seja completamente desprovida de esquemas, pois são necessárias as definições de coleções e índices durante a modelagem do banco de dados. Contudo, não é preciso predefinir uma estrutura para nenhum dos documentos que serão adicionados, pois o MongoDB é um banco de dados extraordinariamente dinâmico, de modo que diferentes tipos de documentos podem coexistir em uma mesma coleção. A Figura 6 apresenta um exemplo de dois documentos diferentes pertencentes a mesma coleção.

Figura 6 – Documentos diferentes em uma mesma coleção

```
1 {
2   "tipo": "CD",
3   "artista": "Nirvana",
4   "titulo": "Nevermind",
5   "genero": "Grunge",
6   "dataLancamento": "24/09/1991",
7   "listaMusicas": [
8     {
9       "musica": "1",
10      "titulo": "Smells Like Teen Spirit",
11      "duracao": "5:02"
12    },
13    {
14      "musica": "2",
15      "titulo": "In Bloom",
16      "duracao": "4:15"
17    }
18  ]
19 }
```

```
1 {
2   "tipo": "Livro",
3   "titulo": "Introdução ao MongoDB",
4   "isbn": "978-85-7522-422-9",
5   "editora": "Novatec",
6   "autor": [
7     "Hows, David",
8     "Plugger, Eelco",
9     "Membrey, Peter"
10  ]
11 }
```

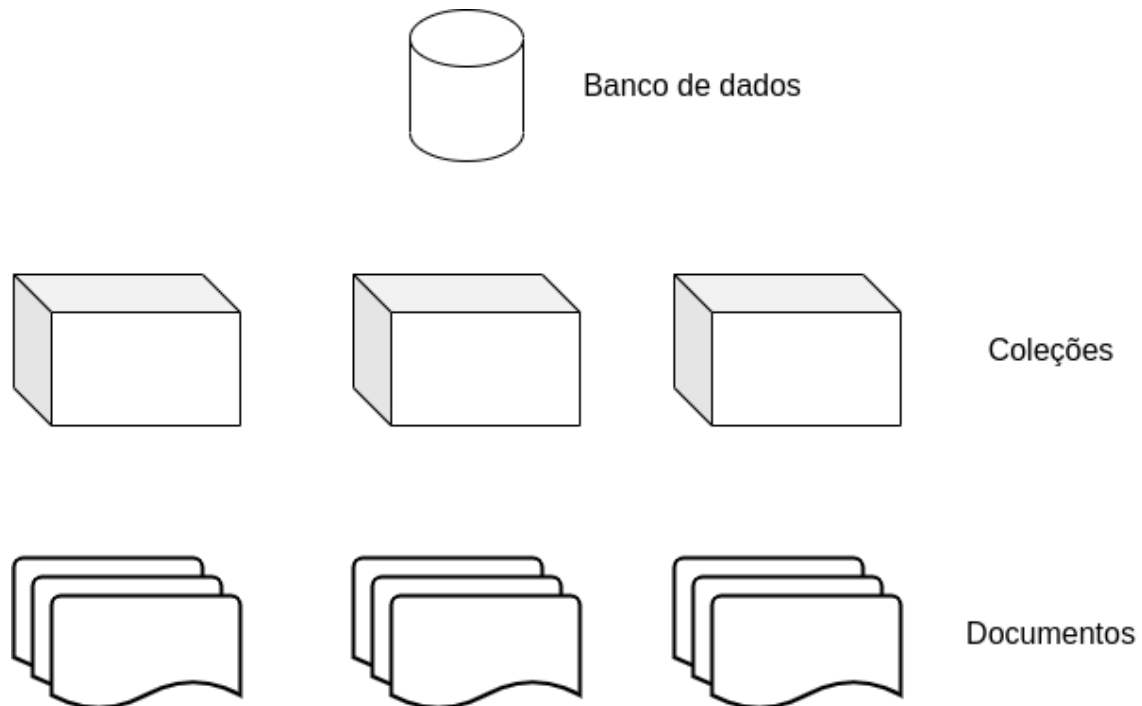
Fonte: elaborado com base em Hows, Membrey e Plugge, 2015.

Um banco de dados orientado a documentos é constituído por um conjunto de documentos que formam uma coleção, e um conjunto de coleções constituem o banco de dados. Assim, uma coleção pode ser compreendida



como um contêiner para armazenamento de documentos. A Figura 7 apresenta o modelo de um banco de dados MongoDB para representar o conceito de documentos, coleções e o banco de dados.

Figura 6 – Modelo de um banco de dados MongoDB



Fonte: elaborado com base em Hows, Membrey e Plugge, 2015.

## 2.1 Documentos e tipos de dados suportados

Os bancos de dados orientados a documentos são considerados uma extensão dos bancos de dados orientados a chave-valor. Desse modo, os documentos são sempre constituídos de pares chave-valor. Relembrando o exemplo apresentado na Figura 6:

**Par: “tipo”: “CD”**

**Chave:** tipo

**Valor:** CD

De acordo com Hows, Membrey e Plugge (2015, p. 74-75), as chaves de um par chave-valor de um documento, são escritas na forma de *strings*, porém os valores podem variar de acordo com tipo de dados que precise ser armazenado. O Quadro 2 apresenta os tipos de dados básicos suportados pelo MongoDB e uma breve descrição de quando são utilizados.



Quadro 2 – Tipos de dados básicos suportados pelo MongoDB

Tipos de dados	Finalidade
String	Utilizado principalmente para armazenar valores textuais. Exemplo: {"universidade": "Uninter"}
Integer	Utilizado para armazenar valores numéricos. Exemplo: {"idade": 42}
Boolean	Utilizado para armazenar os valores "verdadeiro" ou "falso". Exemplo: {"praticaEsporte": true}
Double	Utilizado para armazenar valores de ponto flutuante. Exemplo: {"peso": 73.5}
Arrays	Utilizado para armazenar arrays. Exemplo: "animais": ["gato", "cachorro", "cavalo"]
Null	Utilizado para armazenar um valor nulo. Exemplo: "idade": null

Fonte: elaborado com base em Hows, Membrey e Plugge, 2015.

A Figura 8 apresenta um exemplo de um documento com os tipos de dados String, Integer, Boolean, Double, Array e Null.

Figura 8 – Tipos de dados

```
1 {  
2   "codigo": 1475, ← Integer  
3  
4   "nome": "Alex", ← String  
5  
6   "peso": 65.8, ← Double  
7  
8   "casado": true, ← Boolean  
9  
10  "pets": [  
11    "cachorro",  
12    "gato"  
13  ], ← Array  
14  
15  "habilitacao": null ← Null  
16 }
```

De acordo com Hows, Membrey e Plugge (2015, p. 79), todo documento criado no MongoDB possui um identificador chave único, definido como `_id`. Caso essa chave `_id` não seja adicionada manualmente com um valor específico ao criar o documento, o MongoDB adiciona automaticamente como sendo o primeiro atributo do documento, definindo um valor binário de 12 bytes.



## 2.2 Instalando e acessando o MongoDB

Além do banco de dados DynamoDB, o banco de dados MongoDB também disponibiliza uma versão *offline* para *download* e instalação no computador e, uma versão online para criação do banco de dados na nuvem. Nesta aula focaremos na versão online do MongoDB. Porém, realizaremos o acesso e criação do banco de dados pelo *prompt* de comando no Windows ou pelo terminal (*shell*) no Linux, de modo a abordar também a versão *offline*, visto que os comandos são exatamente os mesmos.

Para *download* da versão *offline*, basta acessar a página de *downloads* do MongoDB: <https://www.mongodb.com/try/download/community>. Em seguida, é necessário escolher a versão do MongoDB, o sistema operacional utilizado e o tipo do pacote, conforme pode ser observado na Figura 9.

Figura 9 – Download do MongoDB

Available Downloads

Version  
4.4.0 (current)

Platform  
Windows

Package  
msi

Download Copy Link

Após realizar o *download*, basta executar o arquivo de instalação e seguir o passo a passo do programa de instalação no Windows. Para usuários Linux, somente é necessária a execução do seguinte comando no terminal

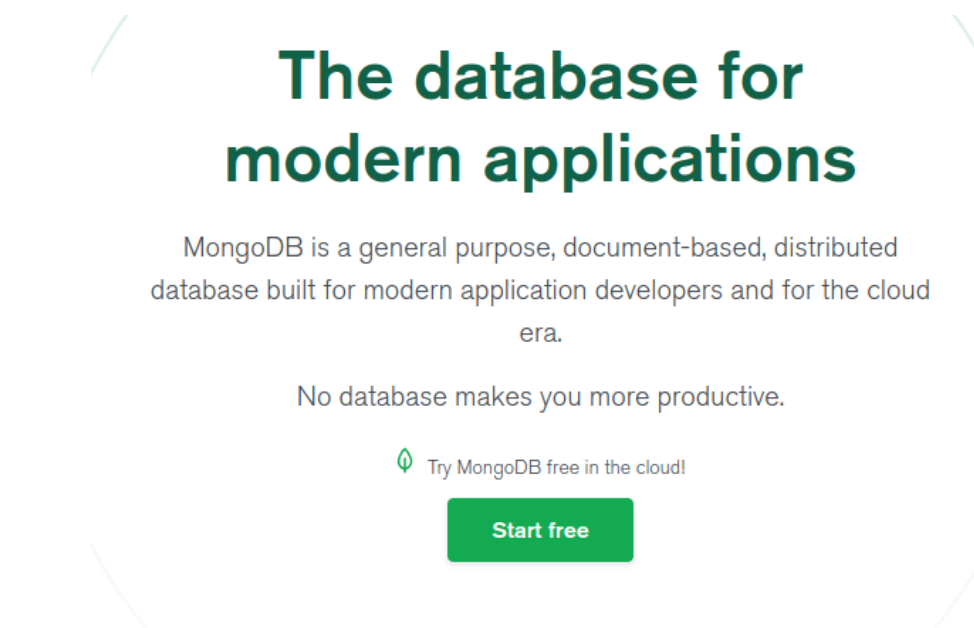
**sudo apt install mongodb**

Após a instalação, para executar o MongoDB basta digitar o comando “mongo” no Terminal do Linux ou, pelo Prompt de Comando do Windows acessar a pasta de instalação do MongoDB e digitar o mesmo comando.

## 2.3 Configurando o MongoDB

Conforme mencionado, daremos ênfase à versão *on-line* do MongoDB. Portanto, o primeiro passo é acessar o seguinte endereço no seu navegador de internet: <https://www.mongodb.com/>. Após carregada a página do MongoDB, o primeiro passo é clicar na opção *Start Free* (iniciar gratuitamente), conforme destacado em verde na Figura 10.

Figura 10 – Página inicial do site do MongoDB



O próximo passo é preencher o formulário com as informações básicas solicitadas para o cadastro, sendo elas:

- Sua empresa (*Your company*): o preenchimento deste campo é opcional;
- Como você usará o MongoDB (*How are you using MongoDB*): selecionar a opção *"I'm learning MongoDB"* (Estou aprendendo o MongoDB);
- Seu e-mail (*Your Work E-mail*): preencher com seu e-mail pessoal;
- Primeiro nome (*First Name*): seu primeiro nome;
- Último nome (*Last Name*): seu sobrenome;
- Senha (*Password*): senha de sua escolha, com no mínimo 8 caracteres.
- Como último passo, deve ser selecionada a caixa de seleção *"I agree to the terms of service and privacy policy"* (Eu concordo com os termos de serviço e política de privacidade).

Todas as informações no *site* estão em inglês, não havendo uma versão em português. A Figura 11 apresenta um exemplo desse formulário.




Figura 11 – Formulário para criação da conta no MongoDB

## Get started free

No credit card required.

How are you using MongoDB?

I'm learning MongoDB 

Your Work Email

usuario@xxyy.com

First Name


Usuario

Last Name

Silva

Password

••••••••


 8 characters minimum

☒ I agree to the [terms of service](#) and [privacy policy](#).

**Get started free**

Após preencher o formulário e clicar no botão "*Get started free*" (*Iniciar gratuitamente*), conforme destacado em verde na Figura 11, será aberta a página da sua conta, conforme apresentado na Figura 12.

Figura 12 – Interface para criação do banco de dados no MongoDB



## Create a cluster

Choose your cloud provider, region, and specs.

**Build a Cluster**

Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).




Após criar sua conta, o primeiro passo é clicar no botão "*Build a Cluster*" (Construir um Cluster), conforme destacado em verde na Figura 13. Na página seguinte, escolha a primeira opção, "*free*" (gratuita), para criar seu repositório de bancos de dados sem custos. O próximo passo é selecionar o servidor e a










localidade do nosso Cluster e, clicar no botão “*Create Cluster*” (Criar Cluster) destacado em verde na Figura 13.

Figura 13 – Seleção do servidor e localidade do *cluster*

Cloud Provider & Region GCP, Sao Paulo (southamerica-east1) ▾



★ Recommended region ⓘ

NORTH AMERICA / SOUTH AMERICA	EUROPE / MIDDLE EAST / AFRICA	ASIA PACIFIC
 Iowa (us-central1) ★	 Belgium (europe-west1) ★	 Taiwan (asia-east1) ★
 Sao Paulo (southamerica-east1) ★		 Tokyo (asia-northeast1) ★
		 Singapore (asia-southeast1) ★
		 Mumbai (asia-south1) ★

---

Cluster Tier M0 Sandbox (Shared RAM, 512 MB Storage) >  
Encrypted

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


[Back](#) [Create Cluster](#)

Após alguns minutos será criado o *cluster* e apresentada a interface conforme podemos observar na Figura 14.

Figura 14 – Interface com os dados do novo *cluster*

## Clusters

**SANDBOX**

 **Cluster0**  
Version 4.2.8

1 **CONNECT** 2 **METRICS** 3 **COLLECTIONS** ...

**CLUSTER TIER**  
M0 Sandbox (General)

**REGION**  
GCP / Sao Paulo (southamerica-east1)

**TYPE**  
Replica Set - 3 nodes

**LINKED REALM APP**  
None Linked



No botão *CONNECT*, indicado com o índice 1 na Figura 14, é possível criar uma conexão remota para o gerenciamento do MongoDB. No botão *METRICS*, indicado com o número 2, são apresentadas as métricas de uso do bando de dados e, no botão *COLLECTIONS*, destacado com o número 3, são listadas todas as coleções criadas para este cluster. Ao clicar no botão *CONNECT* é apresentada a interface mostrada na Figura 15 para configuração dos dados de acesso.

Figura 15 – Configuração dos dados de acesso ao Mongo DB.

### Connect to Cluster0

Setup connection security > Choose a connection method > Connect

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](#)

**You can't connect yet.** Set up your firewall access and user security permission below.

**1 Whitelist a connection IP address**

Add Your Current IP Address Add a Different IP Address Allow Access from Anywhere

**2 Create a Database User**

This first user will have **atlasAdmin** permissions for this project.  
Keep your credentials handy, you'll need them for the next step.

<b>Username</b>	<b>Password</b>	<span>Autogenerate Secure Password</span>
<input type="text" value="ex. dbUser"/>	<input type="text" value="ex. dbUserPassword"/>	<span>SHOW</span>
<span>Create Database User</span>		

Na opção número 1 da Figura 15, é necessário clicar no botão destacado em verde “*Add Your Current IP Address*” (Adicione Seu Endereço de IP Atual), para adicionar o seu endereço IP de acesso ao MongoDB. Na opção número 2 é necessário informar um nome de usuário e uma senha de acesso ao banco de dados. Por fim, basta clicar no botão destacado em vermelho “*Create Database User*” (Criar Usuário do Banco de Dados) e, em seguida, em “*Choose a connection method*” (Escolher o método de conexão), conforme destacado em verde na Figura 16.





Figura 16 – Escolha do método de acesso

## Connect to Cluster0

Setup connection security

Choose a connection method

Connect

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](#)

You're ready to connect. Choose how you want to connect in the next step.

### 1 Whitelist a connection IP address

✓ An IP address has been whitelisted. Add another whitelist entry in the [IP Whitelist tab](#).

### 2 Create a Database User

✓ A MongoDB user has been added to this project. Not yours? Create one in the [MongoDB Users tab](#).

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

Close

Choose a connection method

## Connect to Cluster0

✓ Setup connection security

Choose a connection method

Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.



#### Connect with the mongo shell

Interact with your cluster using MongoDB's interactive Javascript interface



#### Connect your application

Connect your application to your cluster using MongoDB's native drivers



#### Connect using MongoDB Compass

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



Conforme destacado em vermelho na Figura 16, selecionaremos a primeira opção, “*Connect with the mongo shell*” (Conectar com o Mongo shell). Na Figura 17 podemos observar como proceder.



Figura 17 – Conexão ao MongoDB pelo shell

Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

I do not have the mongo shell installed | **I have the mongo shell installed**

1 Select your mongo shell version

4.4

(To check your shell version, run `mongo --version`)

2 Run your connection string in your command line

Use this connection string in your application:

```
mongo "mongodb+srv://cluster0.n7i8w.gcp.mongodb.net/<dbname>" --
```

**Copy**

Replace `<dbname>` with the name of the database that connections will use by default. You will be prompted for the password for the MongoDB user, `alex`. When entering your password, make sure all special characters are [URL encoded](#).

Como realizamos no tópico 2.2 desta aula a instalação do MongoDB, basta selecionar a opção “*I have the mongo shell installed*” (Eu tenho o mongo shell instalado), destacada em verde na Figura 17, e em seguida clicar no botão “*Copy*” (Copiar), destacado em vermelho, para copiar o endereço de acesso conforme os dados de usuário que criamos no passo anterior. Agora, para acessar o MongoDB e criar nossos bancos de dados, basta abrir o terminal no Linux ou o *prompt* de comando no Windows, colar o endereço copiado, pressionar a tecla *Enter*, digitar a senha do usuário e, na sequência, estaremos conectados no MongoDB *on-line*, conforme pode ser observado na Figura 18.

Figura 18 – Conexão ao MongoDB

#### Comando de acesso e senha do usuário

```
(base) alex@AlexDell:~$ mongo "mongodb+srv://cluster0.n7i8w.gcp.mongodb.net/<dbname>" --username alex
MongoDB shell version v3.6.3
Enter password: █
```

#### Interface do MongoDB

```
MongoDB server version: 4.2.8
WARNING: shell and server versions do not match
2020-08-14T17:44:30.991-0300 I NETWORK [ReplicaSetMonitor-TaskExecutor-0] Successfully connected to cluster0-shard-00-01.n7i8w.gcp.mongodb.net:27017 (1 connections now open to cluster0-shard-00-01.n7i8w.gcp.mongodb.net:27017 with a 5 second timeout)
MongoDB Enterprise atlas-hyqof8-shard-0:PRIMARY> █
```



### TEMA 3 – OPERAÇÕES CRUD NO MONGODB

A sigla CRUD refere-se a um acrônimo em inglês para as operações *create*, *read*, *update* e *delete*, ou seja, criar, ler, atualizar e excluir, sendo essas as quatro operações básicas de um banco de dados. O Quadro 3 apresenta uma breve comparação entre essas quatro operações em bancos de dados relacionais e no MongoDB.

Quadro 3 – Operações CRUD no modelo relacional e MongoDB

Modelo Relacional	MongoDB
Create	Insert( )
Select (Read)	Find( )
Update	Update( )
Delete	Delete( )

Após conectado ao MongoDB, o primeiro passo é criar o novo banco de dados. Para isso, utilizamos o comando **“use”** similarmente como fazemos nos bancos de dados relacionais para selecionar um banco de dados existente. Como exemplo, criaremos um banco de dados chamado “vendas”. Para isso, basta digitar o comando **“use vendas”**.

Figura 19 – Criação do banco de dados no MongoDB

```
MongoDB server version: 4.2.8
WARNING: shell and server versions do not match
MongoDB Enterprise atlas-hyqof8-shard-0:PRIMARY> use vendas
switched to db vendas
MongoDB Enterprise atlas-hyqof8-shard-0:PRIMARY> 
```

Para visualizar o nome do banco de dados que estamos trabalhando atualmente, basta digitarmos o comando **“db.getName()”**. Porém, se em seguida digitarmos o comando **“show dbs”** para listar todos os bancos de dados existentes no MongoDB, nosso banco de dados “vendas” ainda não será listado, pois ele somente será efetivamente criado, após a inserção do primeiro registro.



### 3.1 Operação insert

Ao contrário da inserção de dados em tabelas nos bancos de dados relacionais, no MongoDB inserimos os documentos em coleções, conforme vimos no início desta aula. Portanto, o MongoDB oferece os seguintes métodos para inserir documentos em coleções.

**db.coleção.insertOne()**  
**db.coleção.insertMany()**

Para inserir um registro de um cadastro de clientes em uma coleção denominada “clientes”, basta digitarmos o seguinte comando:

```
db.clientes.insertOne (  
{  
  nome: “Alex”,  
  endereco: “Rua das avenidas, 288”,  
  telefone: “42-1111-2222”,  
  idade: 18
```

### 3.2 Operação find

A operação de leitura, ou seja, consulta aos dados em uma coleção no MongoDB, é realizada pela operação **find**, similar a operação **select** realizada nos bancos de dados relacionais. Para localizar os dados em uma coleção no MongoDB, digite o comando **db.coleção.find()**. Para listar o documento inserido na coleção clientes no tópico anterior, digite **db.clientes.find()**.

Com o objetivo de filtrar o retorno dos dados, podemos utilizar filtros específicos, como por exemplo, listar todos os clientes com 18 anos **db.clientes.find({idade: 18})**. Para realizar operações lógicas, como listar os clientes com idade maior ou menor do que 18, o MongoDB disponibiliza algumas funções específicas:

- gt (greater than) – maior que;
- gte (greater than or equals) – maior ou igual à;
- lt (less than) – menor que;
- lte (less than or equals) – menor ou igual à.



Então, para localizar os clientes maiores de 18 anos, utilizamos o seguinte comando: **db.clientes.find({idade: {\$gt: 18}})**.

### 3.3 Operação update

Para alterar os dados de um documento, no MongoDB utilizamos a **update**, similar como realizamos nos bancos de dados relacionais. Para editar os dados em uma coleção no MongoDB, digite o seguinte comando:

**db.coleção.updateOne()**

**db.coleção.updateMany()**

Para editar a idade de um cliente na coleção “clientes”, basta digitar o comando **db.clientes.updateOne({nome: “Alex”}, {\$set: {idade: 25}})**. No primeiro parâmetro (nome: “Alex”), é indicada a chave de busca para o registro e, o comando **set** define o atributo que será modificado (idade: 25).

### 3.4 Operação delete

Para realizar operações de exclusão de documentos, o MongoDB oferece duas opções:

**db.coleção.deleteOne()**

**db.coleção.deleteMany()**

Assim, para excluir os registros de clientes que possuem 25 anos, podemos utilizar o comando **db.clientes.deleteMany({idade: 25})**.

## TEMA 4 – CARACTERÍSTICAS DE CONSISTÊNCIA, TRANSAÇÕES E DISPONIBILIDADE

Da mesma forma que a AWS está presente em várias regiões do mundo para a disponibilidade do DynamoDB, o MongoDB também está presente em vários lugares garantindo a disponibilidade do serviço através de cada região. Desse modo, conforme ocorre no DynamoDB, os documentos armazenados no MongoDB também são sincronizados entre os data centers da região escolhida.



## 4.1 Características de transações

No MongoDB, quando é realizada uma operação em um único documento, essa operação é atômica. Conforme MongoDB (2018), como podemos utilizar documentos e matrizes incorporadas para capturar relacionamentos entre dados em uma única estrutura de documento em vez de normalizar em vários documentos e coleções, essa atomicidade de documento único elimina a necessidade de transações de vários documentos para muitos casos de uso práticos. Para situações que exigem atomicidade de leituras e gravações em vários documentos, o MongoDB oferece suporte a transações de vários documentos. Com transações distribuídas, as transações podem ser usadas em várias operações, coleções, bancos de dados e documentos (MongoDB, 2018).

## 4.2 Características de disponibilidade

Conforme Boaglio (2015, p. 149-150), alta disponibilidade é a possibilidade de ter os dados replicados em diferentes lugares, denominados como nodos, e se um nodo cair, outro assume o seu lugar, evitando assim que a aplicação deixe de funcionar. Essa arquitetura é conhecida como *replica set*, onde um nodo é o primário, de onde os dados são lidos e escritos e, os demais são os nodos secundários, em que os dados são copiados do nodo primário para serem utilizados para consulta.

Ainda de acordo com Boaglio (2015, p. 150), caso o nodo primário falhe, um dos nodos secundários assume para ser o novo nodo primário. Novos nodos secundários podem ser adicionados a qualquer instante, sem interromper o cluster inteiro. Para verificar a disponibilidade de cada nodo, um recurso denominado *heartbeat* é utilizado, fazendo com que a cada dois segundos os nodos conversem para verificar se estão ativos.

## 4.3 Características de consistência

De acordo com Duarte (2017), o MongoDB garante conformidade com as propriedades ACID como documento. Se os dados estiverem espalhados entre diversos documentos, não há garantia alguma da consistência e integridade entre esses dados. Nesse sentido, a atomicidade também pode ser comprometida caso os dados estejam espalhados entre diversas coleções.



## TEMA 5 – CASOS DE USO APROPRIADOS

Para Marquesone (2017, p. 49), os bancos de dados NoSQL orientados a documentos são indicados para realizar o armazenamento de conteúdo de páginas *web*, na catalogação de documentos de uma empresa e no gerenciamento de inventário de um *e-commerce*, pois são aplicações que trabalham diretamente com coleções de documentos e, portanto, podem se beneficiar da flexibilidade que o armazenamento orientado a documentos oferece. Além desses cenários, o autor destaca que esse modelo de banco de dados também pode ser muito útil em demais aplicações que utilizam estruturas de dados no formato JSON e que se beneficiam da desnormalização das estruturas dos dados.

Conforme a AWS (2020), os bancos de dados orientados a documentos são ótimas opções para aplicativos de gerenciamento de conteúdo, como *blogs* e plataformas de vídeo, de modo que se o modelo de dados precisar mudar, somente os documentos afetados precisarão ser atualizados e nenhuma atualização de esquema é exigida e nenhum tempo de inatividade de banco de dados é necessário para fazer as alterações.

Esses bancos de dados são eficientes e eficazes para o armazenamento de informações de catálogo. Conforme a AWS (2020), em um aplicativo de comércio eletrônico, diferentes produtos costumam ter números de atributos diferentes, e gerenciar milhares de atributos em bancos de dados relacionais é ineficiente e afeta a performance de leitura. Ao usar um banco de dados de documentos, os atributos de cada produto podem ser descritos em um único documento para gerenciamento fácil e maior velocidade de leitura. Alterar os atributos de um produto não afetará os outros.

### FINALIZANDO

Nesta aula estudamos os conceitos, implementações e aplicações de uso de bancos de dados NoSQL orientados a documentos. No primeiro tema, vimos como são estruturados esses bancos de dados, como é composto um documento no formato JSON e também fizemos analogia entre os componentes de um banco de dados relacional e o modelo orientado a documentos.



No tema 2, abordamos a ferramenta de construção e gerenciamento de bancos de dados orientados a documentos MongoDB, uma das principais e talvez a ferramenta mais conhecida e utilizada no gerenciamento desses bancos de dados. Pudemos conhecer nesse tópico como criar uma conta na *nuvem* do MongoDB e como criar o nosso primeiro acesso através do terminal, tanto na nuvem como em uma versão instalada localmente em nosso computador.

Já no tema 3 nos aprofundamos nas operações para criação dos bancos de dados, inserção, atualização e remoção de documentos, com alguns exemplos práticos, e sempre tentando fazer comparações com as operações do modelo de dados relacional, para nos aprimorarmos a partir de conhecimentos já consolidados. Encerramos esta aula com uma breve análise das características de transações, disponibilidade e consistência do MongoDB, no tema 4, e com as principais áreas de aplicação dos bancos de dados NoSQL orientados a documentos, no tema 5.





## REFERÊNCIAS

AWS. **O que é um banco de dados de documentos?** 2020. Disponível em: <<https://aws.amazon.com/pt/nosql/document/>>. Acesso em: 28 abr. 2021.

BOAGLIO, F. **MongoDB**: Construa novas aplicações com novas tecnologias. São Paulo: Casa do Código, 2015.

DUARTE, L. **Boas práticas com MongoDB**. 22 de set. 2017. Disponível em: <<https://blog.umbler.com/br/boas-praticas-com-mongodb/>>. Acesso em: 28 de abr. 2021.

HOWS, D.; MEMBREY P.; PLUGGE, E. **Introdução ao MongoDB**. São Paulo: Novatec, 2015.

HURWITZ, J. et al. **Big Data para Leigos**. Rio de Janeiro: Alta Books, 2016.

LENNON, J. **Explore o MongoDB**: Saiba por que esse sistema de gerenciamento de bancos de dados é tão popular. 11 de jul. 2011. Disponível em: <<https://www.ibm.com/developerworks/br/library/os-mongodb4/os-mongodb4-pdf.pdf>>. Acesso em: 28 abr. 2021.

MARQUESONE, R. **Big Data**: Técnicas e tecnologias para extração de valor dos dados. São Paulo: Casa do Código, 2017.

MEDEIROS, H. **Introdução ao MongoDB**. 2014. Disponível em: <<https://www.devmedia.com.br/introducao-ao-mongodb/30792#Banco>>. Acesso em: 28 abr. 2021.

MONGODB **Manual**. 2018. Disponível em: <<https://docs.mongodb.com/manual/introduction/>>. Acesso em: 28 abr. 2021.

MONTEIRO, D. **Introdução para modelagem de dados para banco orientado a documentos**. 18 de abr. 2019. Disponível em: <<https://imasters.com.br/banco-de-dados/introducao-para-modelagem-de-dados-para-banco-orientado-documentos>>. Acesso em: 28 abr. 2021.