E-BOOK
DO ESTUDANTE

Banco de dados NOSQL

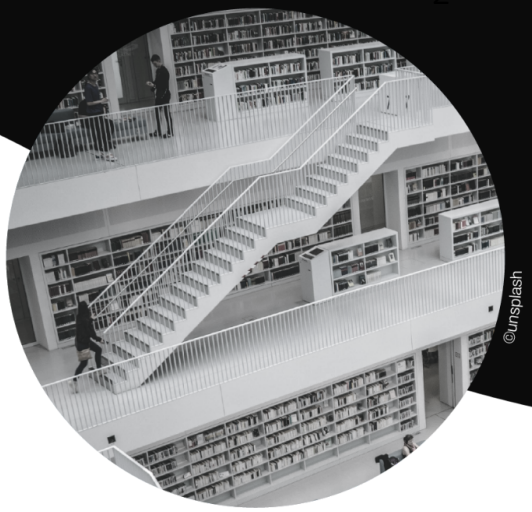
Biblioteca

RESILIA

SUMÁRIO

Banco de dados NOSQL

Biblioteca



Os bancos de dados não relacionais ou bancos de dados NoSQL (*Not Only* SQL) são uma alternativa aos bancos de dados relacionais tradicionais, como o MySQL.
Eles foram criados para atender à crescente demanda do mercado por soluções de armazenamento de dados **escaláveis**, **flexíveis** e capazes de lidar com **grande volume de dados** não estruturados.

Nesse contexto, é importante também conhecer o **MongoDB**, um dos mais populares bancos de dados NoSQL, que utiliza o modelo de documentos para armazenamento em vez de tabelas relacionais. Isso permite que dados heterogêneos e complexos sejam armazenados de forma eficiente e flexível, sem a necessidade de definir um esquema rígido antes do tempo.

Ao contrário dos bancos de dados relacionais, que se concentram na normalização dos dados e na definição de relações entre tabelas, o MongoDB prioriza a **performance** e a **escalabilidade**, sendo ideal para aplicações de alta intensidade de dados, como jogos, aplicativos móveis e análise de dados, por exemplo.

Aqui, vamos explorar os **conceitos básicos** de bancos de dados NoSQL e do MongoDB e ver como eles podem ser utilizados para **solucionar desafios de armazenamento de dados** em diferentes aplicações no contexto do desenvolvimento web.

CONTEXTUALIZANDO

Os bancos de dados NoSQL foram desenvolvidos para suprir as **limitações** dos bancos de dados relacionais **tradicionais**, criados na década de 1970 para atender a necessidades de sistemas de informação simples e estáticos. Com o avanço da internet e a popularização de aplicações web e móveis, houve a necessidade de soluções de armazenamento de dados que fossem escaláveis e capazes de lidar com grandes volumes de dados não estruturados.

Nesse sentido, os bancos de dados NoSQL passaram a oferecer **vantagens** sobre os tradicionais, tais como a capacidade de lidar com dados complexos,

escalabilidade horizontal, alta disponibilidade e alto desempenho em aplicações de alta intensidade de dados. Além disso, permitiam modelar dados de forma flexível e dinâmica, sem a necessidade de definir um esquema rígido antes da implementação, o que é particularmente importante em aplicações que estão em constante **mudança**, onde os requisitos de dados podem variar rapidamente.

Para entendermos melhor a **diferença entre um banco de dados SQL e um banco de dados NoSQL**, vamos imaginar que uma empresa possui uma loja virtual e precisa armazenar informações sobre os seus produtos e clientes.

Caso o time de desenvolvimento do site estivesse usando um **banco de dados relacional** (SQL), criaria **tabelas** separadas para cada tipo de informação, como uma tabela "produtos" e uma tabela "clientes". Cada tabela teria colunas específicas para armazenar informações, como "nome", "preço", "descrição", etc.

No entanto, utilizando um **banco de dados NoSQL**, como o MongoDB, as informações são armazenadas em **documentos** e cada documento é uma estrutura de dados semelhante a um **objeto**, que pode conter várias informações diferentes e estruturas complexas, como listas e objetos aninhados.

Por exemplo, em vez de criar duas tabelas separadas, o time pode ter um único documento para cada produto e cliente, com informações como "nome", "preço" e "endereço". Isso torna o armazenamento e a recuperação de informações mais fáceis e flexíveis, pois não é necessário se preocupar com relações entre tabelas.

Em resumo, a principal diferença entre bancos de dados SQL e NoSQL é a **forma** como as informações são estruturadas e armazenadas. Enquanto os bancos de dados SQL usam **tabelas** e **relações** para armazenar informações, os bancos de dados NoSQL usam **documentos** para armazenar informações de forma mais flexível e escalável.

Vamos nos aprofundar em bancos de dados NoSQL?

BANCOS DE DADOS NOSQL

Os bancos de dados NoSQL são sistemas de gerenciamento de banco de dados que não necessitam da estrutura de tabelas, colunas e chaves. A utilização deles vem crescendo rapidamente devido às seguintes **funcionalidades**:

- **Escalabilidade horizontal:** Foram projetados para serem altamente escaláveis, ou seja, para lidarem com uma quantidadea muito grande de dados, o que permite a adição de mais “nós” (recursos) à medida que as necessidades de armazenamento de dados aumentam.

- **Flexibilidade de modelagem de dados:** Ao contrário dos bancos de dados relacionais (SQL), que utilizam as tabelas estruturadas para armazenar dados, os bancos de dados NoSQL permitem a modelagem de dados de maneira mais flexível, como documentos, gráficos, chaves-valor, entre outros.

- **Performance:** Foram projetados para fornecer alta performance em comparação com os bancos de dados relacionais. Para isso, usam técnicas como cache e distribuição de dados para garantir uma resposta rápida aos usuários, mesmo com grandes quantidades de dados. Isso ajuda muito principalmente em aplicações como *e-commerce* e redes sociais, que precisam de mensagens e itens de forma instantânea.

- **Aplicações em nuvem:** Com a popularização das nuvens como Microsoft Azure, Google Cloud, AWS, muitas empresas estão procurando maneiras de migrar seus aplicativos para esse tipo de armazenamento. Os bancos de dados NoSQL são uma opção bastante popular para aplicações na nuvem devido à sua escalabilidade e flexibilidade, e em muitas das nuvens são os bancos mais utilizados.

Assim, os bancos de dados NoSQL são uma escolha ideal para aplicações que exigem grandes volumes de dados, alta escalabilidade e performance, já que oferecem uma maneira mais flexível para modelar os dados e são capazes de lidar com as demandas da era digital.

MongoDB

Os bancos NoSQL são uma alternativa aos SQL. Para essa relação ficar mais clara podemos fazer a seguinte comparação: enquanto o SQL é como um **livro de registro** organizado em tabelas, o MongoDB é como uma **grande coleção de documentos**.

Em vez de usar tabelas e relações rigorosamente definidas, o MongoDB armazena dados em documentos no formato **JSON** (JavaScript Object Notation), permitindo que a modelagem de dados seja mais flexível. Isso significa que é possível adicionar, remover ou modificar campos de documentos sem precisar alterar a estrutura de todo o banco de dados. Ele também é altamente escalável e permite a adição de mais **nós** ao *cluster* de forma simples e eficiente, sem interrupções para o usuário. Além disso, o MongoDB tem uma **arquitetura distribuída** que permite distribuir dados em diferentes nós, aumentando a disponibilidade e performance.

Vamos ver um **exemplo prático** para deixar claro como podemos utilizar o MongoDB em nossas aplicações!

O primeiro passo é baixar o MongoDB e é importante escolher a opção **“MongoDB *Community Server*”**, que é a versão para comunidade e que permite utilizar a maioria dos recursos do MongoDB.

Try MongoDB Community Edition | MongoDB

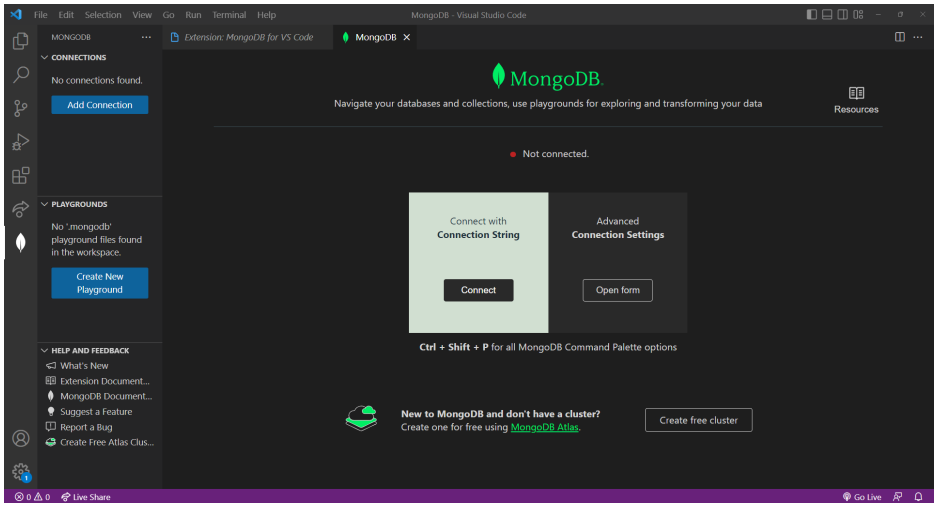
Depois de baixar, o instalador já traz as opções totalmente pré-configuradas. O que é preciso selecionar é na parte de ***“choose setup type”***, onde deve-se selecionar “complete”. Depois disso, é só seguir com **NEXT**.

Atenção! Na última etapa da instalação, há uma *checkbox* assinalada para instalar o **“mongo compass”**, que é a interface visual do MongoDB. Como ele não será utilizado para o exemplo prático, sua instalação é opcional.

Após termos o MongoDB instalado, vamos instalar uma **extensão no VS Code** para que possamos utilizar e executar os comandos no MongoDB por ele:

MongoDB For VS Code | MongoDB

Com a extensão instalada, vamos começar a utilizá-la e conhecer a sintaxe do MongoDB. Para isso, vamos clicar no **ícone de “folha”** criado em nosso VS Code:



Nesta interface que foi aberta, vamos clicar na opção **“Open form”** e ir em **“Connect”**. Se tudo estiver certo com o MongoDB, aparece uma mensagem de sucesso.

Após conectado, vamos na opção lateral escrita como **“Create New Playground”** para criar um novo ambiente de desenvolvimento dentro do VS Code e podermos executar os comandos MongoDB. O mongo automaticamente cria um novo arquivo com exemplos de códigos, como este:

```
// Select the database to use.

use('mongodbVSCodePlaygroundDB');
```

Esse código está informando que é preciso selecionar o banco de dados que será utilizado. Porém a função “use” do mongo nos permite também criar um banco, caso não exista. Para executar esse código, vamos selecionar todo o código e apertar no botão de **“RUN”**, localizado no canto superior direito.

Haverá um pedido de confirmação sobre a execução, que deve ser aceito. Então, o MongoDB retorna o seguinte *feedback*: “*switched to db **mongodbVSCodePlaygroundDB***”, que significa que estamos utilizando o banco **“mongodbVSCodePlaygroundDB”**.

Agora, que já estamos utilizando o banco teste, vamos executar outro comando que também aparece no arquivo de exemplo do Mongo. Vamos inserir alguns dados dentro da **“pasta”** (Collection) de nome “sales”, que também não criamos. Isso ocorre porque o Mongo vai criar automaticamente a coleção, caso ela não exista.

Vamos executar o seguinte código:

```
db.sales.insertMany([

  { '_id': 1, 'item': 'abc', 'price': 10, 'quantity': 2, 'date': new
Date('2014-03-01T08:00:00Z') },

  { '_id': 2, 'item': 'jkl', 'price': 20, 'quantity': 1, 'date': new
Date('2014-03-01T09:00:00Z') },

  { '_id': 3, 'item': 'xyz', 'price': 5, 'quantity': 10, 'date': new
Date('2014-03-15T09:00:00Z') },

  { '_id': 4, 'item': 'xyz', 'price': 5, 'quantity':  20, 'date': new
Date('2014-04-04T11:21:39.736Z') },

  { '_id': 5, 'item': 'abc', 'price': 10, 'quantity': 10, 'date': new
Date('2014-04-04T21:23:13.331Z') },

  { '_id': 6, 'item': 'def', 'price': 7.5, 'quantity': 5, 'date': new
Date('2015-06-04T05:08:13Z') },

  { '_id': 7, 'item': 'def', 'price': 7.5, 'quantity': 10, 'date': new
Date('2015-09-10T08:43:00Z') },

  { '_id': 8, 'item': 'abc', 'price': 10, 'quantity': 5, 'date': new
Date('2016-02-06T20:20:13Z') },

]);
```

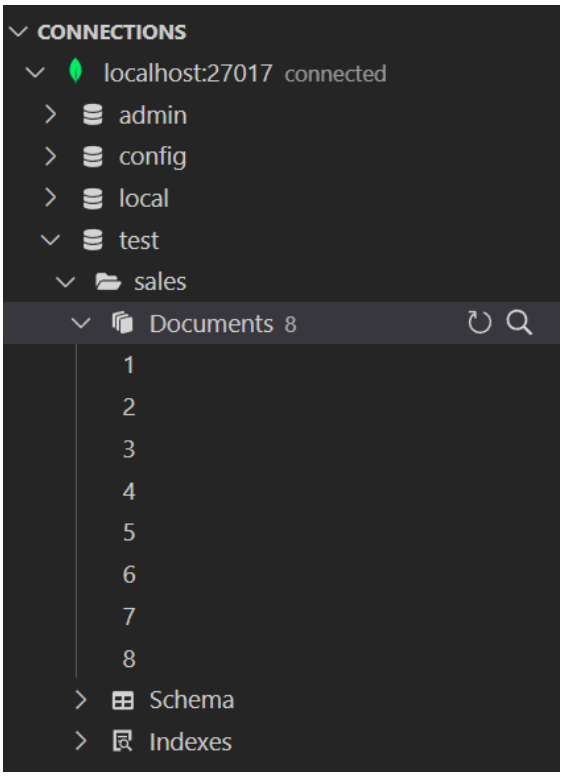
Esse código está inserindo vários dados dentro da coleção “sales”. É possível notar que na aba de ***connections*** temos o nosso ***localhost*** e dentro dele estão os nossos **bancos**. Dentro dos bancos estão as nossas **coleções** e, dentro delas, os nossos ***documents*** (documentos).

Dentro do Mongo, a estruturação funciona de forma diferente a um banco SQL. Ele sempre vai trabalhar com a seguinte **estrutura**:

- `banco de dados`
  - `coleções`
    - `arquivos`
      - `dados`

Ou seja, sempre há um **banco** englobando tudo. A **coleção** será a pasta que armazena, organiza e separa os arquivos. Já os **arquivos** vão conter os **dados**, onde cada arquivo contém um **objeto** com o conjunto de dados.

No exemplo anterior, no banco de dados **"test"** há a coleção **"sales"**. Dentro da coleção há um total de oito **arquivos**, sendo eles os oitos **dados** que foram inseridos no último exemplo do código. Essa estrutura do Mongo permite, além de uma **organização focada na velocidade e desempenho**, que cada arquivo contenha uma quantidade diferente de dados conforme são inseridos. Isso ajuda a manipular diferentes estruturas de dados dentro da mesma coleção, como podemos ver na imagem a seguir:



Até aqui, vimos como instalar e utilizar o Mongo dentro do VS Code. A seguir, veremos como conseguimos realizar todas as operações do CRUD e vamos criar um exemplo de banco em MongoDB.

Glossário

Cluster(s) e nó(s): Cluster é uma coleção de computadores interconectados que trabalham juntos como se fosse um único sistema. Em um cluster de banco de dados, cada computador (ou nó) é responsável por armazenar uma parte dos dados e realizar operações de leitura e escrita.

Para refletir

- Por que utilizar banco de dados NoSQL? Qual a importância dele para as aplicações em nosso dia a dia?
- Qual a importância de se aprofundar e conhecer o mundo do banco de dados NoSQL? O que esse conhecimento o pode agregar em seu percurso profissional?

Criando um banco de dados com MongoDB

Imagine que estamos trabalhando como um time de desenvolvimento e precisamos criar um **banco de dados** para uma loja virtual que armazene informações sobre os produtos e os clientes.

Para começar, vamos criar um novo *playground* dentro do VS Code. Para isso é preciso acessar o ícone de folha que simboliza o MongoDB dentro do VS Code e apertar no botão “Create New Playground”.

É importante lembrar que caso a janela do VS Code tenha sido reiniciada, fechada ou recarregada, haverá uma conexão salva, que fica dentro de Connections. Basta clicar nela e o MongoDB estará conectado.

Depois da reconexão, vamos limpar o arquivo que o VS Code criou quando apertamos no botão de “Create New Playground”. Para criar o **banco** da loja virtual, vamos utilizar o seguinte comando:

```
use("LojaVirtual")
```

Lembrando que o comando **“use”** permite utilizar um banco de dados já existente. Mas se o banco de dados não existir, um novo banco será criado com o nome indicado.

Depois do banco, vamos criar nossa **coleção** de nome “produtos”, executando o seguinte comando:

```
use("LojaVirtual")
```

```
db.createCollection("produtos")
```

Após então criarmos nossa coleção de nome “produtos” dentro do nosso banco de dados “LojaVirtual” iremos então começar a inserir nossos produtos, para isso podemos utilizar dois métodos, sendo eles:

```
db.produtos.insertMany()
```

```
db.produtos.insertOne()
```

Sempre dentro do Mongo teremos esta **sintaxe**: o “Many” sempre para o plural e o “One” para somente um item. Com isso, podemos inserir vários itens ao mesmo tempo utilizando o ***insertMany()*** ou inserir somente um item através do ***insertOne()***. No nosso caso, vamos inserir um total de **três elementos**, para isso iremos utilizar o *insertMany():*

```
use("LojaVirtual")

db.createCollection("produtos")

db.produtos.insertMany([

  {_id: 1, titulo: "Televisor", valor: 2.000},

  {_id: 2, titulo: "Tênis", valor: 150.00, cores: ["preto",
"banco", "roxo"]},

  {_id: 3, titulo: "Caneca", valor: 80.00, estampas: ["avatar",
"avatar 2"]}

])
```

Estamos criando três produtos diferentes que exigem tipos de **estruturas** diferentes. O produto de “_id” “dois” (2) tem a propriedade “cores” que nenhum outro tem. O mesmo ocorre para o último produto, que tem a propriedade “estampas” que nenhum outro possui. Essa é a **flexibilidade** que um banco de dados NoSQL permite. É possível inserir atributos diferentes para cada produto, onde cada produto tem sua própria estrutura, o que permite que os nossos dados possam ser melhor armazenados.

Após executarmos o comando, vamos verificar como o produto de “_id” “dois” (2) foi inserido. Para isso iremos realizar o método:

```
use("LojaVirtual")
```

```
db.produtos.findOne({_id: 2})
```

O **“findOne”** foi utilizado para pesquisar um item através de uma **condição**, ou

seja, estamos pesquisando um documento dentro da **coleção** “produtos” onde o **arquivo** tem o “_id” igual a dois (2). Temos o retorno do dado dentro do arquivo:

```
{
  "_id": 2,
  "titulo": "Tênis",
  "valor": 150,
  "cores": [
    "preto",
    "banco",
    "roxo"
  ]
}
```

É possível notar neste dado retornado que o MongoDB utiliza um armazenamento em **JSON**. Então, todos os dados são salvos em **objetos** contendo **atributos com valores**, novamente permitindo uma melhor flexibilidade em customizar cada produto.

Agora então que já vimos o ***Insert*** e o ***Find***, vamos ver como é possível **alterar um produto**. Vamos utilizar o seguinte código:

```
use("LojaVirtual")

db.produtos.updateOne({_id: 2}, {$set: {cores: ["preto", "branco"]}})
```

Aqui estamos utilizando o método ***updateOne***. Assim como o ***insert***, temos o ***updateMany***. Porém, neste caso, queremos apenas atualizar o produto de *“_id”* dois (2), e estamos então **“setando”** (atribuindo) um novo valor ao método “cores” que antes continha os dados `"cores": ["preto", "banco", "roxo"]` e agora estamos mudando valor para `["preto", "branco"]`. Percebeu como a sintaxe mudou um pouquinho?

Temos, então, o método **updateOne** e dentro dele estamos passando dois objetos. O primeiro `{_id: 2}` é a nossa condição, mais conhecida como a **query**, ou seja, utilizamos a query **parâmetro** para que seja possível selecionar o produto correto para atualizar. Depois, temos um segundo objeto que contém o que vamos alterar dentro do produto que selecionamos, onde estamos setando um novo valor ao atributo de nome “cores”.

Depois de mudar, vamos conferir se o dado realmente foi alterado, selecionando-o novamente:

```
use("LojaVirtual")

db.produtos.findOne({_id: 2})
```

Ele então nos retorna o seguinte dado:

```
{
  "_id": 2,
  "titulo": "Tênis",
  "valor": 150,
  "cores": [
    "preto",
    "branco"
  ]
}
```

Se notarmos, temos apenas os valores “preto” e “banco” dentro do **array** que está armazenado no atributo “cores”. Isso significa que conseguimos atualizar nosso dado. Agora, apenas precisamos deletar um dado.

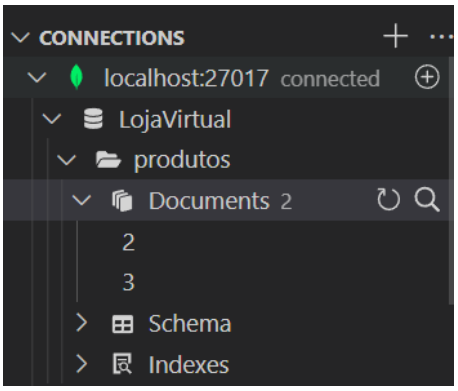
Para deletar o produto que contém o “_id” igual a um (1), vamos executar o seguinte código:

```
use("LojaVirtual")

db.produtos.deleteOne({_id: 1})
```

Novamente estamos utilizando o **“One”**, o que significa que eu posso deletar vários dados, caso seja necessário, utilizando o **“Many”** em seu lugar. Porém, em nosso caso, precisamos deletar um único produto, o que contém o “_id” igual a um (1).

Depois de executar o código, podemos observar que o produto de _id igual a um (1) não existe mais, pois foi deletado, como podemos ver na aba lateral:



Ao longo do texto, foi possível destacar quais são as **vantagens** de se utilizar um banco de dados NoSQL, quais são as **diferenças** de NoSQL para o SQL e qual a importância do NoSQL nesta era onde muitas aplicações são dependentes de grandes escalas de dados.

Conhecemos também o **MongoDB**, onde criamos o nosso primeiro **CRUD** em um banco de dados NoSQL, utilizando os métodos **findOne, insertOne, updateOne, e deleteOne**, e o nosso banco de dados para a loja online.

O assunto de NoSQL que envolve seus conceitos, sintaxes e utilizações é complexo e requer aprofundamento e dedicação. Por isso, é importante sempre consumir conteúdos sobre ele e praticar bastante.

Atividade: Finalizando a loja virtual

Utilizando o que vimos neste e-book, crie uma nova coleção de nome “usuarios” e dentro dela:

- Crie um total de 4 usuários.
- Busque pelo usuário que contenha o “_id” igual a 4.
- Altere o nome do usuário que contenha o “_id” igual a 3.
- Delete o usuário com “_id” igual a 2.

Para refletir

- Quais são as duas palavras que o MongoDB utiliza para se referir a um “único elemento” ou “vários elementos”?
- Para você, qual a importância de ter conhecido todos os métodos do CRUD utilizando o MongoDB?
- Quais são as vantagens em usar o MongoDB?

Para ir além

- Quem trabalha com desenvolvimento precisa saber como utilizar a sintaxe do MongoDB. Por isso, para conhecer e se aprofundar dentro dos comandos que o MongoDB disponibiliza e para conseguir manipular os dados, consulte a documentação oficial do MongoDB:
  <https://www.mongodb.com/docs/manual/reference/method/db.collection.find/>

  <https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/>

  <https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteOne/>

  <https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/>

NUVEM DE PALAVRAS