

## Unidade 1

### Sistema de Gerenciamento de Banco de Dados não Relacional

#### Aula 1

Banco de dados não relacional

#### Introdução



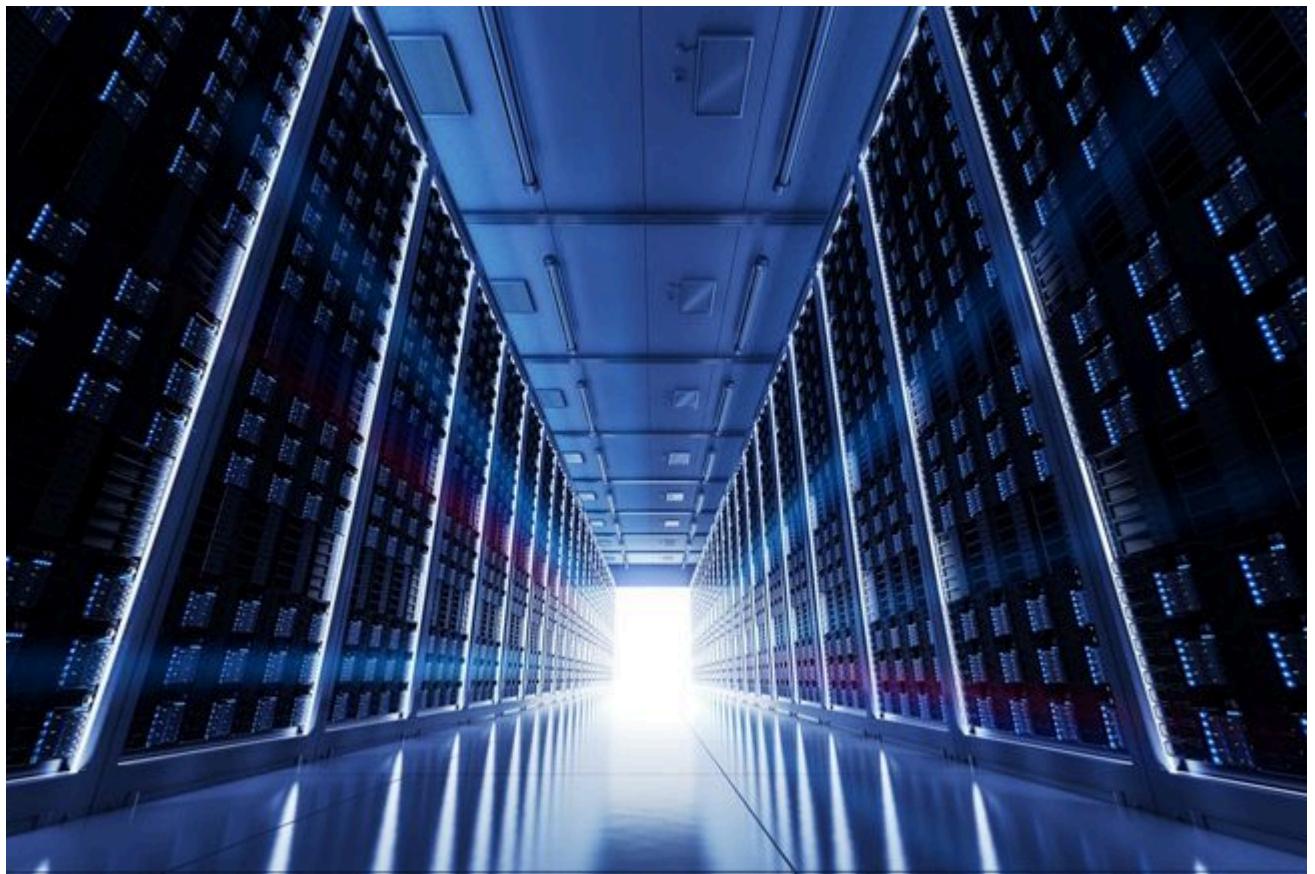
Vamos começar nossos estudos abordando a história dos bancos de dados! Trata-se de uma história rica e que cresceu consideravelmente com o advento dos computadores e mudou imensamente desde sua criação. A criação da World Wide Web, na década de 1990, possibilitou impulsionar o crescimento exponencial da indústria de banco de dados. E na década de 2010 tivemos uma maior conscientização sobre dados, com o surgimento de *big data* e uma maior ênfase na proteção de dados.

Continuando nossos estudos, veremos um pouco a respeito do banco de dados não relacionais, o qual consiste em um modelo que armazena os dados em um formato não tabular e tende a ser mais flexível do que as estruturas de banco de dados relacionais baseadas em SQL.

# Bancos de Dados Não Relacionais

E finalizamos vendo algumas das principais aplicações do banco de dados não relacionais, que vão desde armazenamento de conteúdos, aplicações móveis, internet das coisas e outros.

## História dos bancos de dados



A história dos bancos de dados é rica e cresceu consideravelmente com o advento dos computadores, apesar de ter mudado imensamente desde sua criação. A aplicação inicial da tecnologia de banco de dados foi para resolver problemas com os sistemas de processamento de arquivos. No início da década de 1960, Charles Bachman projetou o primeiro banco de dados computadorizado, conhecido como *Integrated Data Store* (IDS), baseado no modelo de dados de rede. No fim da década, a IBM desenvolveu com base no modelo de banco de dados hierárquico, o *Information Management System*.

Foi na década de 1970 que o modelo de banco de dados relacional foi desenvolvido por Edgar Codd. Muitos dos modelos de banco de dados que usamos hoje são baseados em relacionamentos (GEEKS FOR GEEKS, 2020). Em seu modelo, o esquema do banco de dados, ou organização lógica, é desconectado do armazenamento físico de informações, e isso se tornou o princípio padrão para sistemas de banco de dados (QUICKBASE, 2022).

Mais tarde, a década de 1980 marcou o crescimento dos bancos de dados. A IBM desenvolveu o *Structured Query Language* (SQL) como parte do projeto R. Os sistemas de banco de dados

# Bancos de Dados Não Relacionais

relacionais tornaram-se um sucesso comercial à medida que o rápido aumento nas vendas de computadores impulsionou o mercado de banco de dados, o que causou um grande declínio na popularidade dos modelos de banco de dados de rede e hierárquicos. Os modelos de navegação anteriores desapareceram, enquanto a comercialização de sistemas relacionais viu esse tipo de banco de dados aumentar em uso e popularidade.

Os sistemas de banco de dados orientados a objetos se tornaram mais populares nos anos 1990, novas ferramentas de cliente para desenvolvimento de aplicativos foram lançadas, incluindo o Oracle Developer, PowerBuilder, VB e outros. Diversas ferramentas de produtividade pessoal, como ODBC e Excel/Access, também foram desenvolvidas. Protótipos para Sistemas de Gerenciamento de Banco de Dados de Objetos, ou ODBMS, foram criados no início da década de 1990 (QUICKBASE, 2022).

A criação da World Wide Web, na década de 1990, possibilitou impulsionar o crescimento exponencial da indústria de banco de dados. Altos investimentos em negócios online alimentaram a demanda por sistemas de banco de dados cliente-servidor. Com o aumento do uso da tecnologia de ponto de venda, o processamento de transações on-line e o processamento analítico on-line começaram a amadurecer.

Desde a década de 1980, o SQL serviu como a linguagem de consulta padrão. Mas em 1998 Carlo Strozzi cunhou o termo “NoSQL” ao nomear seu banco de dados Strozzi NoSQL. Essa oferta inicial ainda era de natureza relacional, no entanto (QUICKBASE, 2022).

A década de 2010 foi uma década de maior conscientização sobre dados, com o surgimento de *big data* e uma maior ênfase na proteção de dados. Atualmente, os bancos de dados estão em toda parte e são usados para melhorar nosso dia a dia. Do armazenamento em nuvem pessoal à previsão do tempo, muitos dos serviços que utilizamos hoje são possíveis devido aos bancos de dados. Alguns dos bancos de dados relacionais atuais incluem gigantes como Oracle, MySQL e DB2.

## História dos bancos de dados



Os bancos de dados não relacionais (ou NoSQL) armazenam seus dados em um formato não tabular e tendem a ser mais flexíveis do que as estruturas de banco de dados relacionais tradicionais baseadas em SQL. Baseiam-se em estruturas de dados como documentos e um documento pode ser altamente detalhado enquanto contém uma variedade e tipos de informações em diferentes formatos (MONGODB, 2022). Os principais tipos de banco de dados são:

## 1. Bancos de dados de valores-chave

São alguns dos bancos de dados NoSQL menos complexos, pois todos os seus dados consistem em uma chave indexada e um valor. Eles usam um mecanismo de *hash* para que, dada uma chave, o banco de dados possa recuperar rapidamente um valor associado. Os mecanismos de *hash* fornecem acesso em tempo constante, o que significa que mantêm alto desempenho mesmo em grande escala. As chaves podem ser qualquer tipo de objeto, mas normalmente são uma *string*. Eles facilitam o armazenamento de grandes quantidades de dados e realizam consultas de pesquisa rapidamente (DATASTAX, 2022; PATTINSON, 2020).

## 2. Bancos de dados de documentos

Os bancos de dados de documentos expandem a ideia básica de armazenamentos de valores-chave, em que os “documentos” são mais complexos, pois contêm dados e cada documento recebe uma chave exclusiva, que é usada para recuperar o documento. Eles são projetados para armazenar, recuperar e gerenciar informações orientadas a documentos, geralmente armazenadas como JSON. Cada documento pode conter diferentes tipos de dados. Grupos de

documentos são chamados de coleções e cada documento em uma coleção pode ter uma estrutura diferente (DATASTAX, 2022; PATTINSON, 2020).

Como o banco de dados de documentos pode inspecionar o conteúdo do documento, o banco de dados pode realizar algum processamento de recuperação adicional. E os bancos de dados de documentos têm um esquema flexível conforme definido pelo conteúdo do documento.

### 3. Bancos de dados tabulares

Os bancos de dados tabulares organizam os dados em linhas e colunas, mas com um toque diferente. Também conhecidos como armazenamentos de colunas largas ou armazenamentos de linhas particionados, eles oferecem a opção de organizar linhas relacionadas em partições que são armazenadas juntas nas mesmas réplicas para permitir consultas rápidas. E assim como os bancos de dados de valores-chave e documentos, os bancos de dados tabulares usam *hash* para recuperar linhas da tabela (DATASTAX, 2022; PATTINSON, 2020).

### 4. Bancos de dados de gráficos ou grafos

Armazenam seus dados usando uma metáfora gráfica para explorar as relações entre os dados. Os nós no gráfico representam itens de dados e as arestas representam os relacionamentos entre os itens de dados. Os bancos de dados Graph são projetados para dados altamente complexos e conectados, que superam os recursos de relacionamento e junção de um RDBMS. Os bancos de dados gráficos geralmente são excepcionalmente bons em encontrar semelhanças e anomalias entre grandes conjuntos de dados (DATASTAX, 2022; PATTINSON, 2020).

### 5. Bancos de dados multi modelo

São projetados para oferecer suporte a vários modelos de dados em um único *back-end* integrado. A maioria dos sistemas de gerenciamento de banco de dados é organizada em torno de um único modelo de dados que determina como os dados podem ser organizados, armazenados e manipulados. Por outro lado, um banco de dados multi modelo permite que uma empresa armazene partes dos dados do sistema em diferentes modelos de dados, simplificando o desenvolvimento de aplicativos (DATASTAX, 2022; PATTINSON, 2020).

## Aplicações do banco de dados não relacional



Os bancos de dados NoSQL são usados de várias maneiras. Os bancos de dados gráficos são usados para analisar conexões nos dados, enquanto os armazenamentos de chave-valor são frequentemente usados para caches e em arquiteturas de microserviços. Devido à flexibilidade do modelo de documento, os bancos de dados de documentos são usados para uma ampla variedade de aplicativos, desde a criação de aplicativos móveis até a consolidação de muitas fontes de dados em uma única visualização e o suporte a arquiteturas orientadas a eventos em tempo real. Vejamos algumas aplicações do banco de dados não relacional a seguir:

## 1. Loja de perfis de usuário

Para habilitar transações online, preferências do usuário, autenticação de usuário e muito mais, é necessário armazenar o perfil do usuário por Web e aplicativo móvel.

Nos últimos tempos, os usuários de aplicativos Web e móveis cresceram muito. O banco de dados relacional não poderia lidar com um volume tão grande de dados de perfil de usuário, pois apresenta limitação física com relação à infraestrutura, no caso de servidores. O uso da capacidade NOSQL pode ser facilmente aumentado adicionando servidores, o que torna o dimensionamento econômico.

## 2. Armazenamento de conteúdo e metadados

# Bancos de Dados Não Relacionais

Muitas empresas como editoras exigem um local onde possam armazenar grande quantidade de dados, que incluem artigos, conteúdo digital e e-books, a fim de mesclar várias ferramentas de aprendizado em uma única plataforma.

Os aplicativos que são baseados em conteúdo, para esses metadados de aplicativos, são dados acessados com muita frequência que precisam de menores tempos de resposta.

Para a construção de aplicativos baseados em conteúdo, o uso de NoSQL oferece flexibilidade no acesso mais rápido aos dados e no armazenamento de diferentes tipos de conteúdo.

## 3. Aplicações móveis

Como os usuários de smartphones estão aumentando muito rapidamente, os aplicativos móveis enfrentam problemas relacionados ao crescimento e ao volume de dados.

Usando o banco de dados NoSQL, o desenvolvimento de aplicativos móveis pode ser iniciado com tamanho pequeno e pode ser facilmente expandido à medida que o número de usuários aumenta, o que é muito difícil se você considerar bancos de dados relacionais.

Como o banco de dados NoSQL armazena os dados sem esquema, o desenvolvedor do aplicativo pode atualizar os aplicativos sem precisar fazer grandes modificações no banco de dados.

## 4. Internet das Coisas

Hoje, bilhões de dispositivos estão conectados à internet, como smartphones, tablets, eletrodomésticos, sistemas instalados em hospitais, carros e armazéns. Para tais dispositivos, grande volume e variedade de dados são gerados e continuam gerando mais dados.

O NoSQL permite que as organizações expandam o acesso simultâneo a dados de bilhões de dispositivos e sistemas conectados, armazenem uma grande quantidade de dados e atendam ao desempenho exigido.

## 5. Segmentação de anúncios

Exibir anúncios ou ofertas na página atual é uma decisão com receita direta. Para determinar qual grupo de usuários segmentar, na página da Web onde exibir anúncios, as plataformas reúnem características comportamentais e demográficas dos usuários.

Um banco de dados NoSQL permite que as empresas de publicidade rastreiem os detalhes do usuário e coloquem muito rapidamente anúncios que aumentam a probabilidade de cliques.

## Videoaula: Banco de dados não relacional



**Este conteúdo é um vídeo!**

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos diretamente no aplicativo

# Bancos de Dados Não Relacionais

para assistir mesmo sem conexão à internet.

Embora existam diferentes maneiras que podem ser incorporadas para entender como os bancos de dados NoSQL funcionam, veremos agora alguns dos recursos mais comuns que definem um banco de dados NoSQL básico. Ainda temos assuntos interessantes que ainda precisam ser abordados! Então não deixe de conferir esse vídeo.

## Saiba mais



Para entender melhor banco de dados não relacionais, seguem alguns links, cujo conteúdo aborda com mais detalhes esse assunto. Vale a pena conferir:

1. [Dados não relacionais e NoSQL](#)
2. [Comparativo entre os modelos de banco de dados relacional e não relacional](#)
3. [Dados não relacionais e NoSQL](#)

---

## Referências



DATASTAX. What is NoSQL? **Data Stax**, 2022. Disponível em: <https://www.datastax.com/what-is/nosql>. Acesso em: 9 maio 2022.

GEEKS FOR GEEKS. History of DBMS. **Geeks for Geeks**, 2020. Disponível em: <https://www.geeksforgeeks.org/history-of-dbms/>. Acesso em: 9 maio 2022.

MONGODB. What Is a Non-Relational Database? **MongoDB**, 2022. Disponível em: <https://www.mongodb.com/databases/non-relational>. Acesso em: 9 maio 2022.

PATTINSON, T. Relational vs. non-relational databases. **Plural Sight**, 2020. Disponível em: <https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases>. Acesso em: 9 maio 2022.

QUICKBASE. A Timeline of Database History & Database Management. **Quick Base**, 2022. Disponível em: <https://www.quickbase.com/articles/timeline-of-database-history>. Acesso em: 9 maio 2022.

## Aula 2

Instalação e primeiros passos - MongoDB

### Introdução



Iniciaremos nosso estudo abordando a arquitetura do MongoDB, que é uma forma lógica de categorizar os dados que serão armazenados no banco de dados. NoSQL é um tipo de banco de dados que ajuda a realizar operações em big data e armazená-lo em um formato válido. É amplamente utilizado devido à sua flexibilidade e uma grande variedade de serviços.

Em seguida, veremos uma das formas de instalação do MongoDB, em que basta seguir os passos ou mesmo acessar a própria plataforma do MongoDB que também apresentará as respectivas instruções. Finalizaremos apresentando alguns dos principais comandos do MongoDB, não se limitando ao nosso estudo, podemos verificar muito outros na própria plataforma.

## Arquitetura do MongoDB

# Bancos de Dados Não Relacionais



Como definição, MongoDB é um banco de dados de código aberto que usa um modelo de dados orientado a documentos e uma linguagem de consulta não estruturada. É um dos sistemas e bancos de dados NoSQL mais poderosos atualmente.

O padrão de arquitetura é uma forma lógica de categorizar os dados que serão armazenados no banco de dados. NoSQL é um tipo de banco de dados que ajuda a realizar operações em big data e armazená-lo em um formato válido. É amplamente utilizado devido a sua flexibilidade e uma grande variedade de serviços.

## 1. A arquitetura do banco de dados MongoDB NoSQL

O banco de dados de documentos busca e acumula dados na forma de pares de valores-chave, mas aqui os valores são chamados de documentos. Documento pode ser declarado como uma estrutura de dados complexa e aqui pode ser uma forma de texto, arrays, strings, JSON, XML ou qualquer outro formato. O uso de documentos aninhados também é muito comum e muito eficaz, pois a maioria dos dados criados geralmente está na forma de JSONs e não é estruturada (GEEKS FOR GEEKS, 2020).

As **vantagens** são:

- Este tipo de formato é muito útil e apto para dados semi estruturados;
- A recuperação de armazenamento e o gerenciamento de documentos são fáceis.

# Bancos de Dados Não Relacionais

A ampli

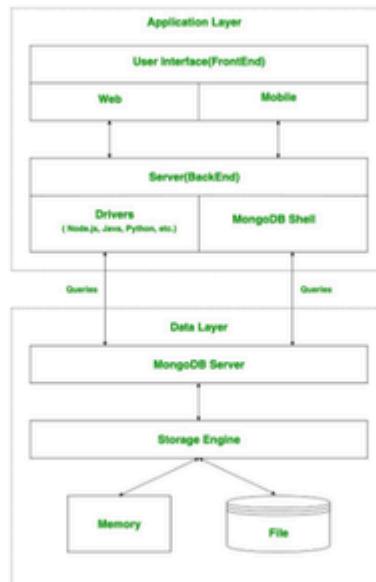
Já as **limitações** são:

- O manuseio de vários documentos é um desafio;
- As operações de agregação podem não funcionar com precisão.

A seguir estão os componentes da arquitetura MongoDB:

- **Base de dados:** em palavras simples, pode ser chamado de contêiner físico de dados. Cada um dos bancos de dados tem seu próprio conjunto de arquivos no sistema de arquivos com vários bancos de dados existentes em um único servidor MongoDB.
- **Coleção:** um grupo de documentos de banco de dados pode ser chamado de coleção. Dentro da coleção, vários documentos podem ter campos variados, mas principalmente os documentos dentro de uma coleção destinam-se ao mesmo propósito ou para servir ao mesmo objetivo final.
- **Documento:** um conjunto de pares chave-valor pode ser designado como um documento, estes são associados a esquemas dinâmicos. A vantagem de ter esquemas dinâmicos é que um documento em uma única coleção não precisa ter a mesma estrutura ou os mesmos campos. Além disso, os campos comuns em um documento de coleção podem ter vários tipos de dados.

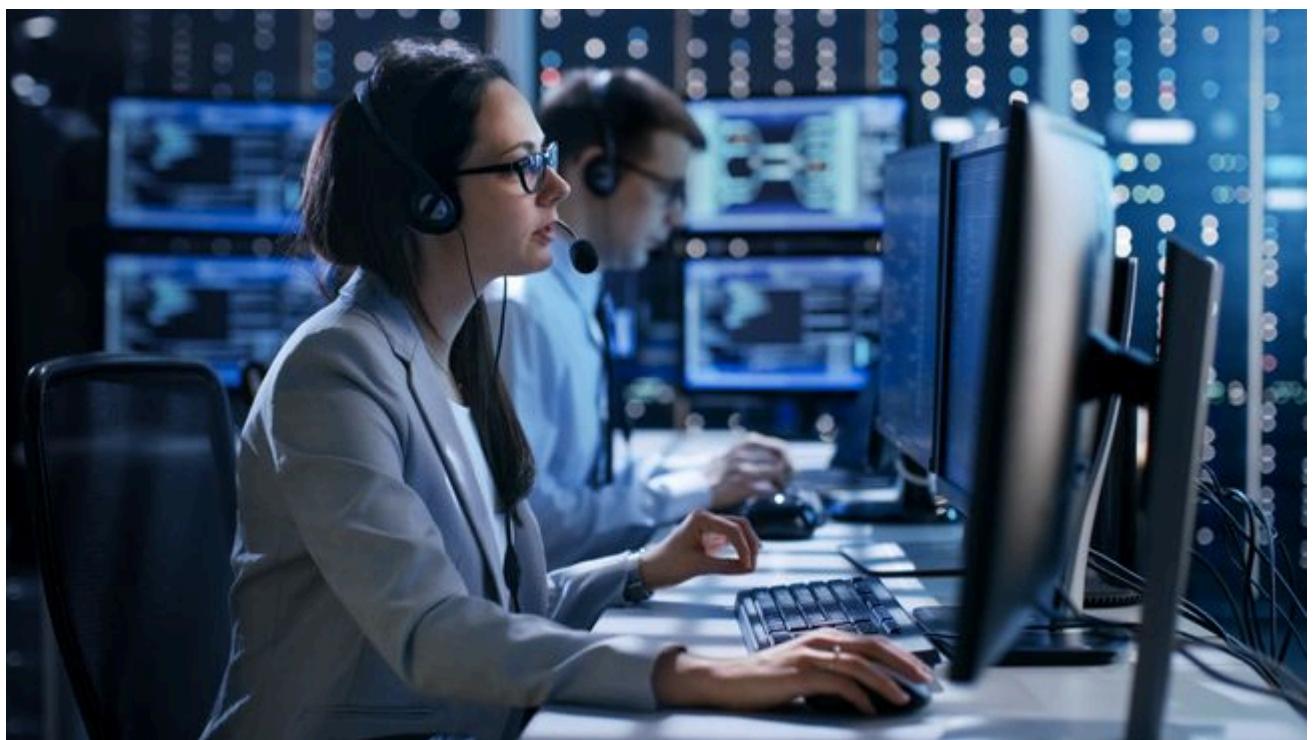
Como sabemos, o ambiente MongoDB oferece um servidor que você pode iniciar e criar vários bancos de dados nele usando o MongoDB, a Figura 1 abaixo mostra como o MongoDB funciona.



Fonte: Geeks for Geeks (2020).

MongoDB funciona em duas camadas: **camada de aplicação** e **camada de dados**. A **camada de aplicação** tem duas partes, a primeira é um Frontend (Interface do Usuário) e a segunda é o Backend (servidor). O **frontend** é o local onde o usuário utiliza o MongoDB com a ajuda de um Web ou Mobile. O **backend** contém um servidor que é usado para executar a lógica do lado do servidor e contém drivers para interagir com o servidor MongoDB com a ajuda de consultas. Essas consultas são enviadas ao servidor MongoDB presente na **camada de dados**. Agora, o servidor MongoDB recebe as consultas e passa as consultas recebidas para o mecanismo de armazenamento. Este mecanismo é responsável por ler ou gravar os dados nos arquivos ou na memória, ou seja, ele basicamente gerencia os dados.

## Instalando o MongoDB



Para realizar a instalação do MongoDB, basta seguir os passos recomendados pelo próprio MongoDB (2022).

- **O que você precisa inicialmente?**

O ["MongoDB"](#) suporta uma variedade de plataformas de 64 bits. Consulte a tabela Plataformas com suporte para verificar se o MongoDB é compatível com a plataforma na qual você deseja instalá-lo.

1. Acesse o endereço: ["MongoDB"](#)

2. Selecione seu sistema operacional entre Windows, Linux, Mac OSX e Solaris. Uma explicação detalhada sobre a instalação do MongoDB é fornecida em seu site.
3. Abra o Windows Explorer/Explorador de Arquivos.
4. Altere o caminho do diretório para onde você baixou o .msiarquivo MongoDB. Por padrão, isso é %HOMEPATH%\Downloads.
5. Clique duas vezes no .msiarquivo.
6. O Windows Installer orienta você no processo de instalação.

**Se você escolher a opção de instalação Personalizada, poderá especificar um diretório de instalação. O MongoDB não apresenta outras dependências do sistema. Você pode instalar e executar o MongoDB de qualquer pasta que escolher.**

Vamos à execução do MongoDB

## 1. Configure o ambiente MongoDB

O MongoDB requer um diretório de dados para armazenar todos os dados. O caminho do diretório de dados padrão do MongoDB é o caminho absoluto \data\dbna, unidade a partir da qual você inicia o MongoDB. Crie essa pasta executando o seguinte comando em um **prompt de comando** conforme o Código 1:

Código 1 | Comando em um prompt de comando

```
1 md \ dados \ db
```

Fonte: MongoDB (2022).

Você pode especificar um caminho alternativo para arquivos de dados usando --dbpath, opção para mongod.exe, por exemplo como apresentado no Código 2:

## Código 2 | caminho alternativo para arquivos de dados

```
1 "C:\Arquivos de Programas\MongoDB\Server\4.2\bin\mongod.exe"  
--dbpath d:\test\mongodb\data md \ dados \ db
```

Fonte: MongoDB (2022).

Se o seu caminho incluir espaços, coloque todo o caminho entre aspas duplas, como apresentado no Código 3:

## Código 3 | caminho alternativo para arquivos de dados

```
1 "C:\Arquivos de Programas\MongoDB\Server\4.0\bin\mongod.exe" -  
--dbpath "d:\test\mongo db data"
```

# Bancos de Dados Não Relacionais

Fonte: MongoDB, 2022.

## 2. Inicie o MongoDB.

Para iniciar o MongoDB, execute mongod.exe. Por exemplo, no **prompt de comando**, apresentado no Código 4:

Código 4 | Comando em um prompt de comando

```
1 "C:\Arquivos de Programas\MongoDB\Server\4.0\bin\mongod.exe"
```

Fonte: MongoDB (2022).

Isso inicia o processo principal do banco de dados MongoDB. A mensagem na saída do console indica que o processo está sendo executado com sucesso.

Dependendo do nível de segurança do seu sistema, o Windows pode exibir uma caixa de diálogo **“Alerta de segurança”** sobre o bloqueio de “alguns recursos” da comunicação em redes. Para obter informações adicionais sobre segurança e MongoDB, consulte a Documentação de segurança [“Security”](#).

## 3. Verifique se o MongoDB foi iniciado com sucesso

Verifique se o MongoDB foi iniciado com sucesso verificando a saída do processo para a seguinte linha, apresentado no Código 5:

Código 5 | Comando em um prompt de comando

```
1 [initandlisten] aguardando conexões na porta 27017
```

Fonte: MongoDB (2022).

A saída deve ser visível no terminal ou na janela do *shell*. Você pode ver avisos na saída do processo. Contanto que você veja a linha de log mostrada acima, você pode ignorar com segurança esses avisos durante sua avaliação inicial do MongoDB.

## Conhecendo o comando Mongo DB



## 1. Por que comandos do MongoDB?

1. Ele pode controlar facilmente os dados que são colocados globalmente, garantindo desempenho e conformidade rápidos.
2. Ele fornece um modelo de dados flexível. Isso acontece no caso em que o aplicativo precisa ser construído do zero ou com o caso de atualização de um único registro.
3. Sem tempo de inatividade, se o aplicativo for dimensionado.

## 2. Recursos

1. O comando MongoDB usa um conceito de replicação mestre-escravo. Para evitar o tempo de inatividade do banco de dados, esse recurso de réplica provou ser um recurso essencial.
2. Esse banco de dados pode ser executado em vários servidores; portanto, os dados são duplicados em vários servidores. O resultado é uma grande vantagem em caso de falha de hardware.

3. O comando MongoDB vem com o recurso de fragmentação automática, no qual o processo distribui dados por várias partições físicas conhecidas como fragmentos. O resultado de que o balanceamento de carga automático acontece.
4. É sem esquema. Portanto, mais eficiente.

**Comandos básicos do MongoDB** (PEDAMKAR, 2020).

- **Criar banco de dados**

No uso do MongoDB, DATABASE\_NAME é usado para criar um banco de dados. Se este banco de dados de nomes não existir, ele será criado, caso contrário retornará o existente, como demonstrado no Código 1.

Código 1 | Criando banco de dados

```
1 use mydb123
```

Fonte: MongoDB (2022).

Para verificar o banco de dados atual agora como se vê no Código 2:

## Código 2 | Verificando o banco de dados

```
1 db
2 mydb123
```

Fonte: MongoDB (2022).

Por padrão, o comando MongoDB vem com o nome do banco de dados “teste”. Suponha que você inseriu um documento sem especificar o banco de dados, ele será automaticamente armazenado em um banco de dados “teste”.

### 2. Criar coleção

Para criar a coleção, o comando do MongoDB usado é:

```
db.createCollection(name, options)
```

Aqui, o nome é o nome da coleção e opções é um documento usado para especificar a configuração da coleção. Embora o parâmetro “Options” seja opcional, é bom fornecê-lo.

### 3. Inserir documento

O método `insert()` ou `save()` é usado para inserir dados em qualquer coleção de banco de dados como apresentado no Código 3.

### Código 3 | Inserir documento

```
1 db.mycol.insert({  
2     _id: ObjectId('7df78ad8902c'),  
3     title: 'MongoDB Overview',  
4     description: 'MongoDB is no sql database',  
5 })
```

Fonte: MongoDB (2022).

Aqui “mycol” é o nome da coleção. Se a coleção não existir, o comando MongoDB criará a coleção de banco de dados e ela será inserida.

#### 4. Documento de consulta

A consulta à coleção é feita pelo método `find()`, como apresentado no Código 4. Como o método `find()` mostrará as descobertas de forma não estruturada, para obter os resultados em um método estruturado é usado um método `pretty()`.

Código 4 | Documento de consulta

```
1 db.mycol.find().pretty()
```

Fonte: MongoDB (2022).

## Comandos intermediários do MongoDB

### 1. Limit()

Este comando do MongoDB limita o número de registros que precisam ser usados no MongoDB. O argumento desta função aceita apenas o tipo numérico. O argumento é o número do documento que precisa ser exibido, e é expresso conforme o Código 5.

## Código 5 | Comando Limit

```
1 db.COLLECTION_NAME.find().limit(NUMBER)
```

Fonte: MongoDB (2022).

## 2. Sort()

Isso é para os registros do MongoDB, onde 1 e -1 são usados para classificar os documentos. 1 é para ascendente enquanto -1 é para descendente, e é expresso conforme o Código 6.

## Código 6 | Comando Sort

```
1 db.COLLECTION_NAME.find().sort({KEY:1})
```

Fonte: MongoDB (2022).

### Dicas para usar comandos do MongoDB

- O MongoDB é, por padrão, sensível a maiúsculas e minúsculas;
- **Proteja o MongoDB usando um firewall** é melhor protegê-lo com um firewall e mapeá-lo para a interface correta;
- **Sem junções:** as junções não são suportadas pelo MongoDB;
- Todos os comandos mencionados e muitos outros podem ser mais explorados usando a documentação oficial do MongoDB.

### Videoaula: Instalação e primeiros passos - MongoDB



#### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

O MongoDB é um banco de dados orientado a documentos de código aberto projetado para armazenar uma grande escala de dados e também permite trabalhar com esses dados de maneira muito eficiente. Você pode instalar o MongoDB usando dois métodos diferentes, um usando msi, e neste vídeo demonstraremos como instalar o MongoDB usando msi, então você precisa seguir cada passo cuidadosamente.

## Saiba mais



Para entender melhor banco de dados não relacionais, seguem alguns links que abordam com mais detalhes esse assunto. Vale a pena conferir.

["Adicionar um driver MongoDB"](#)

["Comandos do banco de dados"](#)

---

## Referências



GEEKS FOR GEEKS. How MongoDB works? **Geeks for Geeks**, 2020. Disponível em: <https://www.geeksforgeeks.org/how-mongodb-works/?ref=gcse>. Acesso em: 13 maio 2022.  
MONGODB. Install MongoDB. **MongoDB**, 2022. Disponível em: <https://www.mongodb.com/docs/guides/server/install/>. Acesso em: 13 maio 2022.  
PEDAMKAR, P. MongoDB Commands. **Educba**, 2022. Disponível em: <https://www.educba.com/mongodb-commands/?source=leftnav>. Acesso em: 13 maio 2022.

## Aula 3

Introdução MongoDB

### Introdução

# Bancos de Dados Não Relacionais



Começaremos nossos estudos dando um passo inicial com o MongoDB. Veremos como iniciar e executar o mongo. E como utilizar o comando relacionado ao banco de dados. Em seguida veremos os comandos utilizados para criar e excluir um banco de dados no Mongo DB. E, por fim, falaremos das certificações que o MongoDB oferece, a saber: o de Desenvolvedor MongoDB e o Administrador de banco de dados MongoDB. Esses certificados fornecem credencial para profissionais que desejam se destacar neste setor. Essas são para indivíduos que conhecem os fundamentos do design e da construção de aplicativos usando o MongoDB.

## Iniciando o MongoDB

# Bancos de Dados Não Relacionais

A ampli



Supondo que você já instalou o MongoDB Server com opções padrão, especialmente a pasta de instalação como

# Bancos de Dados Não Relacionais

C:\Program Files\MongoDB\Server\5.0. Dentro desta pasta, você tem o diretório bin contendo **mongod.exe**.

Supondo também que o caminho do banco de dados seja:

c:\data\db\

Para iniciar o MongoDB Server no Windows, inicie o Mongo Daemon (mongod.exe) usando o seguinte Comando 1:

Comando 1| Iniciar o MongoDB

```
C:\> "C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe"
```

Observe que o programa que estamos executando é **mongod.exe** e não mongo.exe. O **mongo.exe** é usado para iniciar o **Mongo Shell**, enquanto o mongod.exe é usado para executar o Mongo Server. O servidor MongoDB foi iniciado com sucesso Figura 1.



The screenshot shows a terminal window with the following text:

```
2023-05-26T16:26:21.950+00:00: [main] INFO: MongoDB 5.0.10 running on port 27017
2023-05-26T16:26:21.950+00:00: [main] INFO: Listening on /tmp/mongodb-27017.sock
2023-05-26T16:26:21.950+00:00: [main] INFO: Listening for connections from 0.0.0.0:27017
2023-05-26T16:26:21.950+00:00: [main] INFO: No authentication specified, using anonymous users. Read more about security here: https://www.mongodb.com/docs/manual/security/
```

Fonte: elaborada pela autora.

A partir das mensagens registradas no console, você pode observar que:

- O Mongo Server é iniciado como um processo com ID de processo (pid): 13804;
- O Mongo Server está escutando no número da porta: 27017. Você pode ver no final dos logs [initandlisten] aguardando conexões na porta 27017;
- O Mongo Server está usando o banco de dados presente no local C:\data\db\.

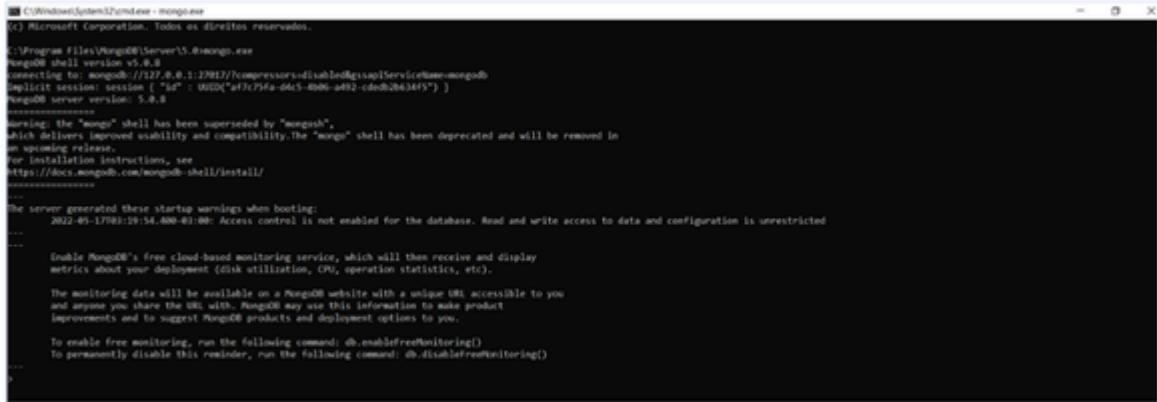
### Não feche esta janela do prompt de comando!

Agora, você pode se conectar a este servidor como clientes de outras janelas do Prompt de Comando.

Quando tiver certeza de que o MongoDB está em execução, abra outra janela de comando e execute o seguinte Comando 2.

Comando 2 | Iniciar o MongoDB

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
```



```
C:\Windows\system32\cmd.exe - mongo.exe
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
MongoDB shell version v5.0.8
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("af7c79fa-0d65-4b06-ad92-cdedb26436f5") }
MongoDB server version: 5.0.8
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====

The server generated these startup warnings when booting:
2023-09-17T08:19:34.000-03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```

Fonte: elaborada pela autora.

Quando nenhum parâmetro é fornecido com o comando mongo, a funcionalidade padrão é que o shell mongo tenta fazer uma conexão com o servidor MongoDB em execução no localhost na porta 27017.

Mas se você gosta de se conectar ao servidor MongoDB que está sendo executado em uma máquina diferente conectada à sua rede, você pode usar as opções do shell mongo conforme mostrado abaixo no Comando 3.

# Bancos de Dados Não Relacionais

Comando 3 | Comando com opções do shell mongo

```
mongo --host <host> --port <port_number>
```

Agora vamos executar uma consulta simples db – Comando 4 – para conhecer o banco de dados para o qual o shell está apontando.

Comando 4 | Comando com opções do shell mongo

```
> db  
test
```

**test** é um banco de dados padrão.

## Porta padrão do MongoDB – 27017

Quando uma instância do MongoDB Server é iniciada em uma máquina, ela precisa começar a escutar em uma porta. Por padrão, o número da porta 27017 é usado para instâncias mongod e mongos. E como alterar essa porta padrão?

Às vezes, podemos exigir iniciar a instância do MongoDB Server em uma porta diferente. O motivo pode ser qualquer um.

Nesse caso, podemos alterar o número de porta padrão do MongoDB ao iniciar o Mongo Daemon passando a opção –port conforme mostrado abaixo no Comando 5.

### Comando 5 | Alterar porta do MongoDB

```
C:\>"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" -port  
27052
```

## Banco de dados MongoDB

**MongoDB Database** é uma coleção de MongoDB Collections e MongoDB Collection é uma coleção de MongoDB Documents.

A seguir estão as operações que podem ser feitas em um banco de dados no MongoDB – estas operações serão detalhadas no segundo bloco desta unidade:

- MongoDB – Criar banco de dados;
- MongoDB – Excluir banco de dados.

## Comando USE DATABASE

Para começar a usar ou alternar para um banco de dados, use o seguinte Comando 6.

Comando 6 | Comando USE DATABASE

```
use <database_name>
```

A seguir está um exemplo para alternar para um banco de dados. Execute o comando no MongoDB Shell, como apresentado no Comando 3, e use o seguinte Comando 7.

Comando 7 | Comando USE DATABASE para alterar um banco de dados

```
> use tutorialkart  
switched to db tutorialkart
```

Agora você mudou para o banco de dados tutorialkart.

## Conceitos básicos



O MongoDB reúne as vantagens do NoSQL. Os conceitos de banco de dados NoSQL foram desenvolvidos para tornar as aplicações mais modernas e rápidas. Os bancos de dados NoSQL superam as desvantagens do SQL.

- **Dados estruturados** – MongoDB pode lidar com dados estruturados, dados semi estruturados e dados não estruturados;
- **Desenvolvimento ágil** – todos os recursos NoSQL do MongoDB o tornam perfeito para a metodologia Agile para desenvolvimento de aplicativos. Os aplicativos modernos são projetados para ter menor tempo de desenvolvimento e que realizam atualizações de forma mais rápida.
- **Escalabilidade** – MongoDB é horizontalmente escalável e, portanto, pode responder aos crescentes requisitos de aplicativos com um menor custo.

O MongoDB cria banco de dados – e o comando USE DATABASE do MongoDB é usado não apenas para selecionar um banco de dados para executar consultas, mas também para criar um banco de dados. Se o nome do banco de dados fornecido ao comando USE Database ainda não estiver presente no MongoDB, um novo banco de dados com o nome será criado quando você inserir um documento em uma coleção nesse banco de dados.

### Exemplo 1 – Criar um banco de dados MongoDB

Segue um exemplo onde tentaremos criar um banco de dados chamado tutorialkart.  
Abra o Mongo Shell e siga os comandos em sequência – Comando 8.

## Comando 8 | Criar um banco de dados MongoDB

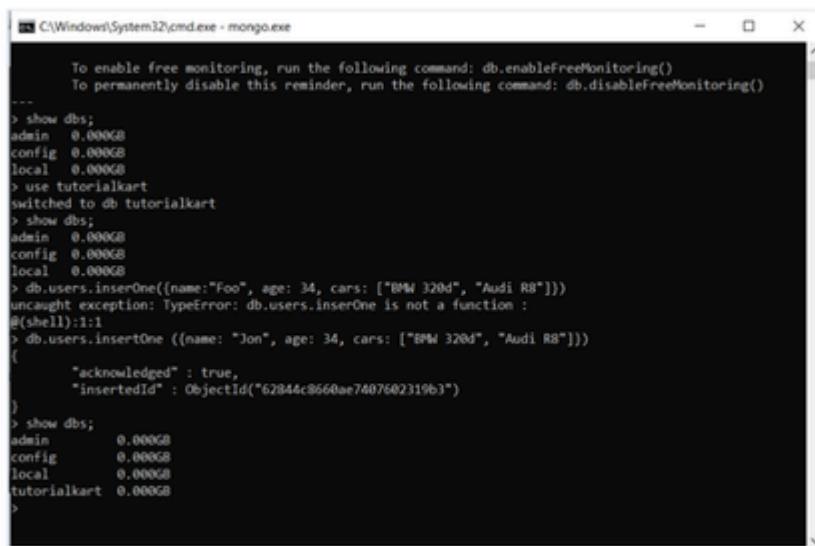
```
>> show dbs;
admin 0.000GB
local 0.000GB

> use tutorialkart
switched to db tutorialkart

> show dbs;
admin 0.000GB
local 0.000GB

> db.users.insertOne( { name: "Jon", age: 34, cars: [ "BMW 320d",
  "Audi R8" ] } )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("62844c8660ae7407602319b3")
}

> show dbs;
admin      0.000GB
local      0.000GB
tutorialkart 0.000GB
>
```



The screenshot shows a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe - mongo.exe'. The window displays the MongoDB shell interface. The user has run several commands to check databases, switch to the 'tutorialkart' database, insert a document into the 'users' collection, and then check the databases again. A reminder message about monitoring is visible at the top of the shell.

```
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
> show dbs;
admin 0.000GB
config 0.000GB
local 0.000GB
> use tutorialkart
switched to db tutorialkart
> show dbs;
admin 0.000GB
config 0.000GB
local 0.000GB
> db.users.insertOne({name:"Foo", age: 34, cars: ["BMW 320d", "Audi R8"]})
uncaught exception: TypeError: db.users.insertOne is not a function
@shell:1:1
> db.users.insertOne ({name: "Jon", age: 34, cars: ["BMW 320d", "Audi R8"]})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("62844c8660ae7407602319b3")
}
> show dbs;
admin      0.000GB
config      0.000GB
local      0.000GB
tutorialkart 0.000GB
>
```

Fonte: elaborada pela autora.

A seguir está a explicação para cada comando MongoDB que executamos acima:

# Bancos de Dados Não Relacionais

1. show dbs – como já existem dois bancos de dados, serão mostrados: admin e local.
2. use tutorialkart – trocando para banco de dados **tutorialkart**.
3. show dbs – mas o **tutorialkart** ainda não foi criado.
4. db.users.insertOne() – verifica se o banco de dados está presente. Se o banco de dados não estiver presente, ele cria um com o nome do banco de dados para o qual foi alternado (na etapa 2) e insere o Documento no banco de dados.
5. show dbs - agora existem três bancos de dados, incluindo nosso banco de dados **tutorialkart** recém-criado.

Também podemos por meio do MongoDB Delete Database – descartar ou excluir um banco de dados MongoDB .

Para excluir ou descartar um banco de dados do MongoDB, basta seguir as estas etapas:

- Seleccione o banco de dados que deseja excluir com a ajuda do comando USE <database>. A seguir está a sintaxe do comando USE

```
use <database_name>;
```

Elimine o banco de dados com a ajuda do comando db.dropDatabase(). A seguir está a sintaxe do comando USE

```
db.dropDatabase()
```

## Exemplo 2 – Exclua o banco de dados MongoDB

A seguir está um exemplo em que tentaremos excluir o banco de dados chamado tutorialkart. Abra o Mongo Shell – no seu prompt de comando – e siga os comandos em sequência.

Comando 9 | Excluir um banco de dados MongoDB

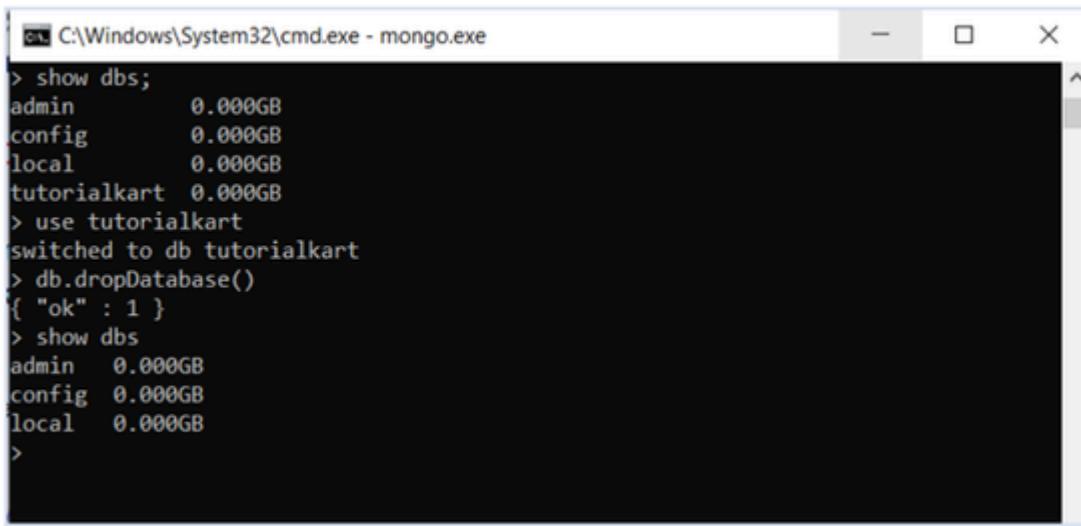
```
>> > show dbs
admin      0.000GB
local      0.000GB
tutorialkart 0.000GB

> use tutorialkart
switched to db tutorialkart

> db.dropDatabase()
{ "ok" : 1 }

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>>
```

Figura 4 | Excluindo um banco de dados MongoDB



```
C:\Windows\System32\cmd.exe - mongo.exe
> show dbs;
admin      0.000GB
config     0.000GB
local      0.000GB
tutorialkart 0.000GB
> use tutorialkart
switched to db tutorialkart
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

Fonte: elaborada pela autora.

A seguir a explicação para cada comando mongodb que executamos acima

1. show dbs – existem três bancos de dados. Vamos excluir o banco de dados ***tutorialkart*** nesta demonstração.
2. use tutorialkart – trocado para banco de dados ***tutorialkart***.
3. db.dropDatabase() - elimina o banco de dados que está atualmente em uso, ou seja, banco de dados ***tutorialkart***.
4. show dbs – agora existem apenas dois bancos de dados, porque o banco de dados ***tutorialkart*** não está mais presente.

## Certificações no MongoDB

# Bancos de Dados Não Relacionais



A certificação MongoDB fornece credencial para profissionais que desejam se destacar neste setor. Essas certificações são para indivíduos que conhecem os fundamentos do design e da construção de aplicativos usando o MongoDB. Essas certificações implicam que desenvolvedores e DBAs tenham meios atualizados para provar sua experiência no MongoDB. Assim, ele é usado para ajudar as organizações a contratar e desenvolver profissionais qualificados, conhecedores dos conceitos e mecânica do banco de dados e habilidades práticas necessárias para construir aplicativos apoiados pelo MongoDB.

Existem **dois exames** oferecidos pelo MongoDB para indivíduos que estão neste setor, a saber:

- Desenvolvedor MongoDB;
- Administrador de banco de dados MongoDB.

Esses certificados podem ser licenciados para permitir que *startups* e empresas ultrapassem seus limites do que um banco de dados pode fazer. Os detentores de certificados são treinados para serem líderes, educadores e especialistas que podem compartilhar seu conhecimento e experiência em primeira mão com proprietários de empresas para sua melhoria. Se alguém quer investir no desenvolvimento de carreira, vale a pena perseguir isso.

A **certificação** não deve ser confundida com o certificado dos cursos gratuitos que o MongoDB oferece, que são bons.

O exame está disponível apenas em determinados horários, por um período de uma semana. E o exame é pago, o valor atual do exame é de US\$ 150. Se você passar, eles lhe darão um link e um

# Bancos de Dados Não Relacionais

número de referência que podem ser usados para verificar a certificação. Inclui uma repetição gratuita, caso não tenha tido sucesso na sua primeira tentativa. É composto por 60 questões de múltipla escolha. Não há penalidades para respostas incorretas. E você terá um prazo de 90 minutos para concluir seu exame de múltipla escolha.

A boa notícia é que você pode fazê-lo em casa a qualquer hora de sua escolha dentro dessa semana programada por eles, usando um software especial de monitoramento baixado na máquina em que o exame será realizado.

Certamente, você deve fazer um dos cursos introdutórios que eles oferecem para o desenvolvimento do MongoDB. Hoje, existem três nessa categoria. Além de algumas especificidades da linguagem de programação, os cursos são fundamentalmente os mesmos. Saiba que após o término de um curso, os vídeos de resposta ficam inacessíveis, embora as palestras permaneçam acessíveis.

Os principais benefícios da certificação MongoDB, são:

- **Adquira Habilidades Valorizadas:** os Profissionais Certificados MongoDB são empregados em organizações de todos os tamanhos, desde grandes empresas até startups.
- **Seja promovido ou contratado:** promova a credibilidade do seu conhecimento.
- **Aumente sua comunidade profissional:** faça networking com outras pessoas do meio e cresça a sua rede de conhecimento e de oportunidades.

Depois de se tornar certificado MongoDB, sua certificação nunca expira. Observe, no entanto, que as certificações do MongoDB se alinham a uma versão principal específica do MongoDB e permanecerão válidas para essa versão.

## Videoaula: Introdução MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Por que utilizar o MongoDB? O MongoDB guardará os dados em documentos ao invés das famosas tabelas, sendo possível alterar os registros, especialmente a sua estrutura, o que nessa plataforma é conhecida também como documentos. O MongoDB também oferece diretamente um desempenho e uma disponibilidade que é considerada muito alta. Vamos ver então os principais recursos do MongoDB? Te aguardo no nosso vídeo!

## Saiba mais



Para entender melhor banco de dados não relacionais, seguem alguns links que abordam com mais detalhes esse assunto. Vale a pena conferir:

1. [Tutoriais para MongoDB](#)
  2. [Vantagens de um Banco de Dados NoSQL, MongoDB](#)
  3. [MongoDB: o que é e como usar o banco de dados NoSQL?](#)
- 

## Referências



MONGODB. Install MongoDB. **MongoDB**, 2022. Disponível em:  
<https://www.mongodb.com/docs/guides/server/install/>. Acesso em: 13 maio 2022.

## Aula 4

Evolução e novas features - MongoDB

### Introdução



Começamos nossos estudos apresentando um pouco dos novos recursos do MongoDB, o que há de novo na versão 5.0, como recursos nativos de séries temporais, redistribuição de dados, compartilhamento de arquivos com melhor controle de versões, e o MongoDB sem servidor. Veremos, em seguida, as transações no MongoDB, que são aplicações utilizadas quando é necessário acessar e modificar vários documentos em uma única operação com integridade garantida, e o MongoDB introduziu transações ACID no mecanismo de banco de dados para atender às necessidades desses casos de uso.

E, por fim, falaremos dos operadores de agregação, que no MongoDB, consiste em operações que agrupam os dados de vários documentos e operam de várias formas para retornar um resultado combinado.

## Novos recursos do MongoDB



O que há de novo no MongoDB? Com a nova versão 5.0 disponibilizada, veremos algumas vantagens que o MongoDB trouxe.

## 1. Recursos nativos de séries temporais

Uma “série temporal” se refere a qualquer tipologia de dados em que os registros são criados sequencialmente em diferentes pontos no tempo. Casos de usos corriqueiros incluem fluxos de medição de sensor e registros de histórico de transações, em que cada registro corresponde diretamente a um momento específico.

Assim, os dados de séries temporais são difíceis de gerenciar e, como resultado, um desenvolvedor é deixado para lidar com um volume muito alto de dados. Além disso, esse tipo de dados é interrompido com atualizações mínimas. Portanto, as consultas nos dados geralmente dependem do uso intenso de filtros baseados em tempo.

O MongoDB 5.0 suporta nativamente todo o ciclo de vida de dados de séries temporais (desde coleta, armazenamento, consulta, análise em tempo real e visualização, até arquivamento online ou invalidação automática à medida que os dados envelhecem), tornando mais rápido a construção e execução. Com o lançamento do MongoDB 5.0, o software se expandiu para a plataforma universal de dados de aplicativos, facilitando para os desenvolvedores o processamento de dados de séries temporais e expandindo ainda mais seus cenários de aplicativos na Internet das Coisas, análise financeira, logística e outros aspectos (HUSSAIN, 2021).

A coleção de séries temporais do MongoDB armazena automaticamente dados de séries temporais em um formato altamente otimizado e compactado, reduzindo assim o tamanho do armazenamento, e a fim de obter melhor desempenho e maior escala. Ele também reduz o ciclo

de desenvolvimento, permitindo que os desenvolvedores construam rapidamente um modelo otimizado para os requisitos de desempenho e análise de aplicativos de séries temporais. Um exemplo de comando para criar uma coleção de dados de série temporal é apresentada no Comando 1:

Comando 1 | Exemplo de comando para criar uma coleção de dados de série temporal

```
1 db.createCollection("collection_name", {timeseries:{timeField:"timestamp"}  
})
```

Fonte: MONGODB (2021).

## 2. Redistribuição de dados

O compartilhamento de dados é fundamental para tornar o banco de dados gerenciável. Com a fragmentação, o cluster de bancos de dados pode compartilhar conjuntos de dados maiores e lidar com solicitações adicionais do desenvolvedor ao mesmo tempo.

A plataforma MongoDB 5.0 permite que os usuários alterem a chave compartilhada para coleções sob demanda à medida que as cargas de trabalho e os bancos de dados continuam a crescer e evoluir, sem tempo de inatividade do banco de dados ou migrações complexas dentro do conjunto de dados (MONGODB, 2021).

## 3. Compatibilidade de aplicativos à prova de futuro

O MongoDB 5.0 traz melhorias significativas à prova de futuro, que está relacionado ao controle de versões e versões. O banco de dados ganhou uma API com controle de versão que permite evitar alterações interrompidas conforme você atualiza para novas versões.

Como resultado, o ciclo de vida do aplicativo pode ser desacoplado do ciclo de vida do banco de dados, fornecendo um nível de proteção do investimento à frente de outros bancos de dados. Além disso, os desenvolvedores podem ter certeza de que o código de seu aplicativo continuará

a ser executado inalterado por anos, sem interrupção, mesmo que o banco de dados seja atualizado e aprimorado abaixo dele (MONGODB, 2021).

### 3. MongoDB sem servidor

A plataforma MongoDB 5.0 apresentou, dentre de suas novidades, novas instâncias do Atlas sem servidor. Atlas é a oferta oficial de banco de dados como serviço da empresa para plataformas de nuvem populares. Com a implantação sem servidor, poderá obter o provisionamento automático dos recursos corretos para a carga de trabalho atual. A plataforma se adapta automaticamente às novas demandas, para que não seja necessário o dimensionamento manual de sua infraestrutura.

O Atlas sem servidor é gerenciado pelo próprio MongoDB; este usa a versão mais recente do banco de dados com suporte para atualizações automáticas. A criação de instâncias sem servidor permite acessar novos *clusters* do MongoDB, em que é possível escolher um provedor de nuvem, criar um novo banco de dados e se conectar a partir do seu aplicativo.

## Transações distribuídas e chaves de particionamento mutáveis



Uma **transação** é uma sequência de operações de banco de dados que só serão bem-sucedidas se todas as operações dentro da transação forem executadas corretamente.

Existem aplicações para as quais é necessário acessar e modificar vários documentos em uma única operação com integridade garantida, mesmo com bancos de dados orientados a

documentos. O MongoDB introduziu transações ACID de vários documentos ainda na versão 4.0 do mecanismo de banco de dados para atender às necessidades desses casos de uso.

Para situações que exigem atomicidade de leituras e gravações em vários documentos (em uma única ou várias coleções), o MongoDB oferece suporte a transações de vários documentos. Com transações distribuídas, as transações podem ser usadas em várias operações, coleções, bancos de dados, documentos e fragmentos. As transações com vários documentos são atômicas, quando:

- Uma transação é confirmada, todas as alterações de dados feitas na transação são salvas e visíveis fora da transação. Ou seja, uma transação não confirmará algumas de suas alterações enquanto reverte outras. Por exemplo: se uma transação for confirmada e a gravação 1 estiver visível no fragmento A, mas a gravação 2 ainda não estiver visível no fragmento B, uma leitura externa na preocupação de leitura "local" poderá ler os resultados da gravação 1 sem ver a gravação 2.
- Quando uma transação é abortada, todas as alterações de dados feitas na transação são descartadas sem nunca se tornarem visíveis. Por exemplo: se qualquer operação na transação falhar, a transação é abortada e todas as alterações de dados feitas na transação são descartadas sem nunca se tornarem visíveis.

## 1. Chaves de fragmentação ou particionamento

O MongoDB fornece escalabilidade horizontal com a ajuda de fragmentação, o que significa distribuir dados em vários servidores, e para isso uma grande quantidade de dados é particionada em blocos de dados usando a chave de fragmento, e esses fragmentos de dados são uniformemente distribuídos em fragmentos que residem em muitos servidores físicos.

A chave de fragmentos determina a distribuição dos documentos da coleção entre os fragmentos do cluster. O MongoDB partitiona dados na coleção usando intervalos de valores de chave de fragmentação. Quando um fragmento cresce além do tamanho permitido, o MongoDB tenta dividir o fragmento em fragmentos menores, sempre com base em intervalos na chave de fragmentação (MONOGODB, 2021).

Para fazer isso, você diz ao MongoDB para usar um de seus índices como uma chave de fragmentação. Em seguida, ele divide seus documentos em partes com chaves de fragmentação semelhantes. Esses pedaços são então distribuídos para seus conjuntos de réplicas, em ordem aproximada de chave de fragmento.

A escolha da chave de fragmentação determina três coisas importantes:

1. A distribuição de leituras e gravações: ao dividir as leituras uniformemente em todos os conjuntos de réplicas, você pode dimensionar o tamanho do conjunto de trabalho linearmente com o número de estilhaços. Você utilizará RAM e discos igualmente em todas as máquinas.
2. O tamanho dos seus pedaços: o MongoDB dividirá grandes pedaços em menores se, e somente se, as chaves de fragmentação forem diferentes. Se você tiver muitos documentos com a mesma chave de fragmentação, acabará com pedaços grandes, que são ruins, não apenas porque fazem com que os dados sejam distribuídos de forma

desigual, mas também porque, uma vez que crescem muito, você não pode mover os fragmentos.

3. O número de fragmentos que cada consulta atinge: é bom garantir que a maioria das consultas atinja o menor número possível de fragmentos. A latência de uma consulta depende diretamente da latência do servidor mais lento que ela atinge.

## Evolução dos operadores de agregação



Vamos presumir que você necessite realizar consultas que agrupam os dados para retornar um único resultado, é que isso possa ser executado diretamente nas queries do banco de dados. Porém, quando vamos realizar este procedimento em bancos não relacionais essas operações não são tão fáceis assim. No MongoDB, por exemplo, elas são executadas a partir de funções de agregação, operações usadas para processar os dados que retornam os resultados calculados. A agregação basicamente agrupa os dados de vários documentos e opera de várias formas para retornar um resultado combinado (BUGAN, 2020).

A agregação é uma forma de processar um grande número de documentos em uma coleção por meio de sua passagem por diferentes estágios. Os estágios compõem o que é conhecido como pipeline. Os estágios em um pipeline podem filtrar, classificar, agrupar, remodelar e modificar documentos que passam pelo pipeline.

Um dos casos de uso mais comuns de agregação é calcular valores agregados para grupos de documentos. Isso é semelhante à agregação básica disponível no SQL com a cláusula GROUP BY e as funções COUNT, SUM e AVG. A agregação do MongoDB vai além e pode também executar junções do tipo relacional, remodelar documentos, criar novas coleções e atualizar coleções existentes e assim por diante.

Embora existam outros métodos de obtenção de dados agregados no MongoDB, a estrutura de agregação é a abordagem recomendada para a maioria dos trabalhos.

Assim como em muitos outros sistemas de banco de dados, o MongoDB permite que você execute uma variedade de operações de agregação. Eles permitem que você processe registros de dados de várias maneiras, como agrupar dados, classificar dados em uma ordem específica ou reestruturar documentos retornados, bem como filtrar dados como em uma consulta.

No MongoDB, as operações de agregação processam os registros/documentos de dados e retornam os resultados computados. Ele coleta valores de vários documentos e os agrupa e, em seguida, executa diferentes tipos de operações nesses dados agrupados, como soma, média, mínimo, máximo, etc., para retornar um resultado calculado. É semelhante à função agregada do SQL. O MongoDB 5.0 fornece duas maneiras de realizar a agregação:

- Pipeline de agregação;
- Métodos de agregação de propósito único.

## 1. Pipeline de agregação

Um pipeline de agregação consiste em um ou mais estágios que processam documentos:

- Cada estágio realiza uma operação nos documentos de entrada. Por exemplo: um estágio pode filtrar documentos, agrupar documentos e calcular valores.
- Os documentos que saem de um estágio são passados para o próximo estágio;
- Um pipeline de agregação pode retornar resultados para grupos de documentos. Por exemplo: retorne os valores total, médio, máximo e mínimo.

## 2. Métodos de agregação de propósito único

São usados quando precisamos de acesso simplificado ao documento, como contar o número de documentos ou encontrar todos os valores distintos em um documento. Ele simplesmente fornece o acesso ao processo de agregação comum usando os métodos count(), distinct() e estimatedDocumentCount() – podemos ver mais detalhados na Figura 1 – portanto, não possui a flexibilidade e os recursos do pipeline de agregação.

Método	Descrição
<code>db.collection.estimatedDocumentCount()</code>	Retorna uma contagem aproximada dos documentos em uma coleção ou exibição.
<code>db.collection.count()</code>	Retorna uma contagem do número de documentos em uma coleção ou exibição.
<code>db.collection.distinct()</code>	Retorna uma matriz de documentos que possuem valores distintos para o campo especificado.

Fonte: MONGODB (2021).

## Videoaula: Evolução e novas features - MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Quais são os tipos de dados no MongoDB? Nele, os documentos são armazenados em BSON, que é o formato codificado binário de JSON e usando BSON podemos fazer chamadas de procedimento remoto no MongoDB. O formato de dados BSON suporta vários tipos de dados. Veremos, então, os tipos de dados do MongoDB. Te aguardo no nosso vídeo!

## Saiba mais



Para entender melhor banco de dados não relacionais, seguem alguns links que abordam com mais detalhes esse assunto. Vale a pena conferir:

1. [Tutoriais para MongoDB](#)
  2. [Vantagens de um banco de dados NoSQL, MongoDB](#)
  3. [MongoDB: o que é e como usar o banco de dados NoSQL?](#)
  4. [O MongoDB 5.0 vem com dados na forma de séries temporais, mudanças na numeração e muito mais](#)
- 

## Referências

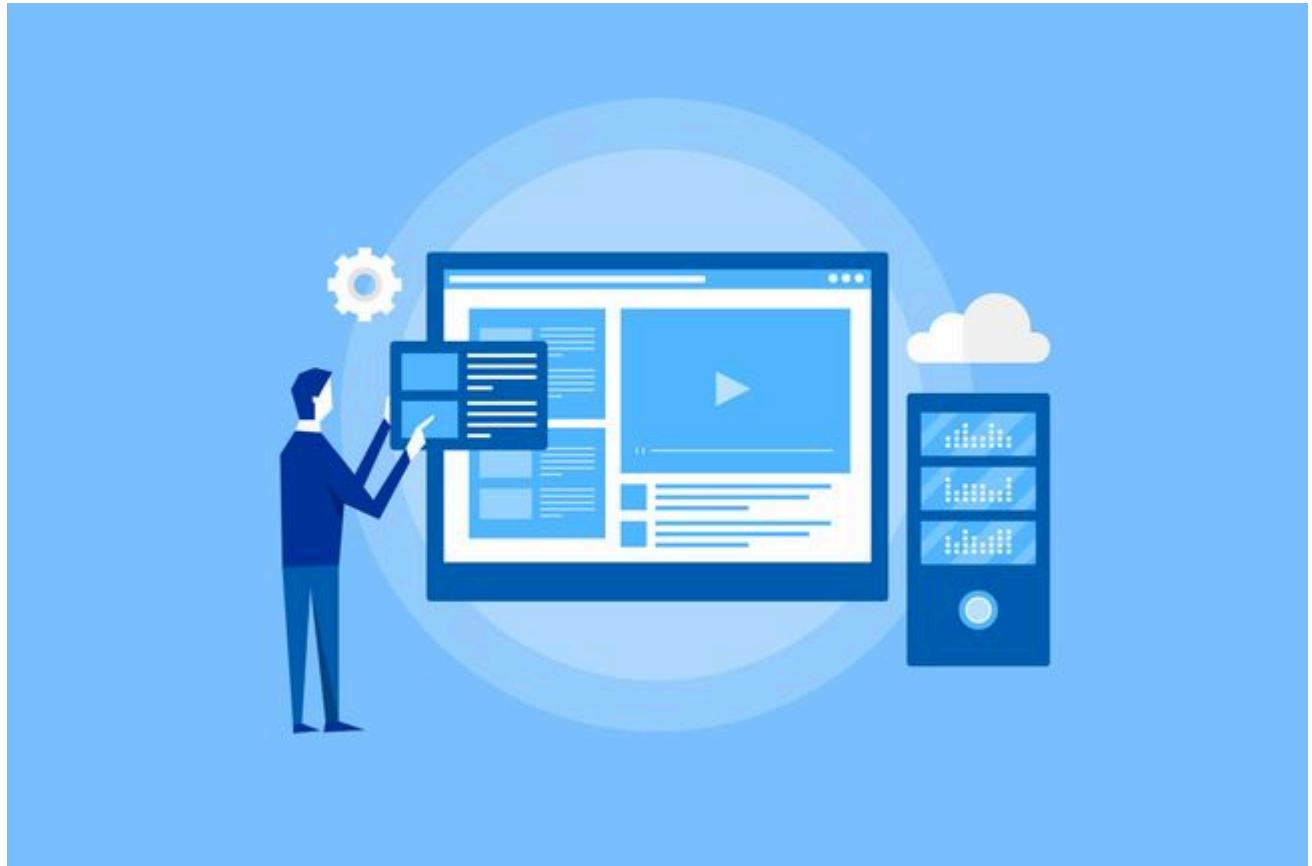


- BUGAN, R. Mongo Aggregation. **Medium**, 2020. Disponível em:  
<https://medium.com/digitalproductsdev/mongo-aggregation-9c3aa3e1dd40>. Acesso em: 22 maio 2022.
- HUSSAIN, S. What's new in MongoDB 5.0. **Medium**, 2021. Disponível em:  
<https://medium.com/cloud-believers/whats-new-in-mongodb-5-0-f9706d3dc547>. Acesso em: 21 de maio 2022.
- MONGODB. Release Notes for MongoDB 5.0. **MongoDB**, 2021. Disponível em:  
<https://www.mongodb.com/docs/manual/release-notes/5.0/#indexes/>. Acesso em: 21 maio 2022.

## Aula 5

Revisão da unidade

**Compreensão das características arquiteturais dos bancos de dados não relacionais e instalação e configuração do sistema de gerenciamento de banco de dados não relacional**



Dentro do contexto dos bancos de dados não relacionais, existe a plataforma de código aberto MongoDB, a qual é conhecida principalmente como banco de dados de documentos, embora nos últimos anos tenha se movido para permitir uma abordagem de banco de dados multi modelo (MONGODB, 2022).

No MongoDB um registro trata de um documento armazenado no formato binário (JSON), e seus documentos ficam agrupados em coleções (ou *collections*). As coleções são análogas às tabelas no banco de dados relacionais (ou tradicional). Podemos ver na Tabela 1 uma comparação entre os termos considerados no banco de dados relacional e no MongoDB.

Tabela 1 | Comparação entre os termos no banco de dados relacional e no MongoDB

Termos/conceitos SQL	Termos conceitos MongoDB
Banco de dados (database)	Banco de dados (database)
Tabela (table)	Coleção (collection)
Linha (row)	Documento ou documento BSON
Coluna (column)	Campo (field)
Índice (index)	índice (index)
Table joins	documentos embutidos e linkados
Chave primária (primary key)	Chave Primária (primary key)
Qualquer coluna pode ser definida como campo único ou uma combinação de de colunas como chave primária	Em MongoDB a chave primária é automaticamente definida como _id

Fonte: Santos (2016).

Os componentes principais do MongoDB, são:

- **Mongod:** o processo de banco de dados principal. Este manipula as requisições dos dados.
- **Mongos:** é o controlador e o roteador de consulta para os *clusters* fragmentados. Faz o serviço de roteamento para as configurações do MongoDB, processa e consulta na camada da aplicação para determinar a localização dos dados.
- **Mongosh:** é o shell interativo que promove uma interface poderosa para administradores de sistemas, assim como uma forma para os desenvolvedores testarem suas consultas.

Um *documento* no MongoDB é o equivalente a um registro em um banco de dados tradicional. Consiste em campos de nome e valor. Cada campo é uma associação entre um nome e um valor e é semelhante a uma coluna em um banco de dados relacional.

Uma coleção no MongoDB é um grupo de documentos do MongoDB e corresponde a uma tabela criada com qualquer outro banco de dados relacional. Não tem estrutura pré-definida.

Um banco de dados é um contêiner de coleções, em que cada um tem seu próprio conjunto de arquivos no sistema de arquivos. Um servidor MongoDB pode armazenar vários bancos de dados.

O foco constituinte do MongoDB tem sido com desenvolvedores de aplicativos, logo, a nova versão da plataforma está voltada principalmente na produtividade do desenvolvedor. No topo da lista, estão as APIs com versão, que começam com a versão 5.0. Especificamente, isso significa que o MongoDB se compromete com a compatibilidade com versões anteriores, desde que o aplicativo use os comandos definidos pela API Versioned.

# Bancos de Dados Não Relacionais

O MongoDB promete, por um período indefinido, que as APIs para aplicativos desenvolvidos para MongoDB 5.0 e posteriores continuarão funcionando sem exigir modificações de código, mesmo que a versão subjacente do banco de dados seja alterada. A necessidade de tal recurso se torna mais essencial, pois, a partir do 5.0, o MongoDB passará para um ciclo de lançamento trimestral. Outro destaque do MongoDB 5.0 se chama suporte a séries temporais "nativas". Trata-se de um tipo especial de coleção que é otimizado para a estrutura de dados de série temporal que reestrutura o esquema em um formato altamente compacto. Também pode ser visto como um aprimoramento de produtividade porque, com o suporte nativo de séries temporais, o MongoDB 5.0 lida com tarefas de rotina associadas a dados de séries temporais dentro do mecanismo de banco de dados (BAER, 2021).

## Vídeo: revisão da unidade



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

O MongoDB é um banco de dados de código aberto, gratuito, de alta performance, sem esquemas e orientado a documentos. O MongoDB é o mais utilizado por organizações de todos os tamanhos. Nele, as informações são armazenadas em documentos BSON, semelhante aos objetos JSON usados largamente na Web. Mas como tirar o maior proveito possível do MongoDB? O objetivo principal deste vídeo é apresentar boas práticas que podem te ajudar quando utilizar o MongoDB.

## Estudo de caso



Como bem sabemos, não há atalhos na criação do banco de dados, correto? Imagine que você trabalhe para uma empresa de desenvolvimento Web e a organização tenha estabelecido o modelo de banco de dados perfeito e você tenha pensado que ele ficará estático por anos, senão, décadas.

E, assim como 99,999% das organizações mundiais, tudo está em fluxo contínuo e você está se saindo melhor do que o esperado (estou feliz por você!), mas chegou o momento em que são exigidas iterações constantes na maneira como você faz seu negócio.

Claro! Nenhuma empresa desenvolveu um ótimo software sem iterações.

E a chave para enfrentar esses desafios dos negócios de hoje é a base de uma infraestrutura de dados forte e flexível. Assim, imagine encontrar um sistema de gerenciamento de banco de dados que se alinhe com os objetivos tecnológicos da sua organização! Bem emocionante, certo?

Apesar de que os tempos tenham mudado muito, junto da demanda por mais diversidade e escalabilidade, acabaram criando-se diversas alternativas no mercado para escolher softwares de banco de dados gratuitos e de código aberto, soluções estas que são até implacáveis em popularidade.

Embora atualmente tenhamos inúmeras soluções de gerenciamento de banco de dados disponíveis, pode ser difícil escolher a solução certa para a empresa. Foi necessário, então, fazer algumas comparações de soluções comuns e os melhores casos de uso que podem ajudá-lo a decidir por uma ferramenta. Apresenta um estudo comparativo a respeito dos bancos de dados MySQL e Cassandra *versus* o MongoDB, contendo também as principais diferenças entre eles e análise com base em parâmetros como seu desempenho, flexibilidade, segurança, etc. se comportam. Explore os casos de uso e os prós e contras a fim de apresentar para a organização, para que você possa, assim, auxiliar na melhor escolha.

## Refletia

MongoDB é um sistema de gerenciamento de banco de dados não relacional (DBMS) de código aberto que usa documentos flexíveis em vez de tabelas e linhas para processar e armazenar várias formas de dados. Lembre-se que:

- Como uma solução NoSQL, o MongoDB não requer um sistema de gerenciamento de banco de dados relacional (RDBMS), por isso fornece um modelo de armazenamento de dados elástico que permite aos usuários armazenar e consultar tipos de dados multivariados com facilidade.
- Documentos ou coleções de documentos do MongoDB são as unidades básicas de dados. Formatados como Binary JSON (Java Script Object Notation), esses documentos podem armazenar vários tipos de dados e ser distribuídos em vários sistemas.
- Como o MongoDB emprega um design de esquema dinâmico, os usuários têm flexibilidade incomparável ao criar registros de dados, consultar coleções de documentos por meio da agregação do MongoDB e analisar grandes quantidades de informações.

Para que possa aprofundar mais no MongoDB, seguem links com algumas dicas interessantes e tutoriais:

["Tutorial de MongoDB para Iniciantes"](#)

["O que há de novo no MongoDB 5.0"](#)

## Videoaula: Resolução do estudo de caso



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

## Comparando o MongoDB com outros bancos de dados

Com tantas soluções de gerenciamento de banco de dados disponíveis atualmente, pode ser difícil escolher a solução certa para sua empresa. Aqui estão algumas comparações de soluções comuns e os melhores casos de uso que podem ajudá-lo a decidir.

### MongoDB

MongoDB é um dos bancos de dados orientados a documentos mais populares sob a bandeira do banco de dados NoSQL. Foi desenvolvido a partir de uma ideia em 2007 e sua primeira versão foi lançada em 2010. É desenvolvido e mantido pela MongoDB Inc.

Ele emprega o formato de pares chave-valor, aqui chamado de armazenamento de documentos. Os armazenamentos de documentos no MongoDB são criados em arquivos BSON que são, na

verdade, uma versão pouco modificada dos arquivos JSON e, portanto, todos os JS são suportados.

Por isso, é frequentemente usado para projetos Node.js. Além disso, o JSON facilita a troca de dados entre aplicativos da Web e servidores em um formato legível por humanos.

Além disso, **oferece maior eficiência e confiabilidade** que, por sua vez, podem atender às suas demandas de capacidade de armazenamento de dados e velocidade.

Além disso, a **implementação sem esquema do MongoDB** elimina a necessidade de definir uma estrutura de banco de dados fixa. Esses modelos permitem a representação de relacionamentos hierárquicos e facilitam a capacidade de alterar a estrutura do registro.

## Recursos do MongoDB

- **Indexação para melhor desempenho de consulta:** oferece suporte à indexação para melhorar o desempenho das operações de pesquisa no banco de dados. Ele fornece variedades de índices que contêm dados de alguns campos. Além disso, a pesquisa é realizada em índices, em vez de em todo o documento, resultando em maior velocidade e desempenho de pesquisa.
- **Replicação para contornar vulnerabilidades:** permite criar várias cópias de dados e implantá-los em vários servidores usando replicação. Isso serve como um recurso para *backup*; se algum servidor falhar, você poderá recuperar os dados de servidores alternativos.
- **Sharding para melhor execução de consultas complexas:** para dimensionar aplicativos da Web de forma eficiente com tempo de inatividade zero, o *sharding* é um método usado pelo MongoDB para distribuir grandes conjuntos de dados em várias coleções de dados. Cada fragmento no cluster funciona como um banco de dados separado que forma um único banco de dados com outros fragmentos, executa consultas complexas e contribui para um melhor balanceamento de carga.
- **Balanceamento de carga para alta escalabilidade:** permite um mecanismo de simultaneidade de controle para atender solicitações simultâneas de clientes a vários servidores de banco de dados. Dessa forma, também reduz a quantidade de carga em servidores individuais, garante consistência na visualização de dados a qualquer momento e ajuda na criação de aplicativos escaláveis.
- **Superta consultas ad-hoc para análises em tempo real:** uma consulta ad-hoc é um comando de curta duração que fornece resultados diferentes para as consultas que estão sendo executadas. O MongoDB suporta consultas ad-hoc que podem ser atualizadas em tempo real usando MongoDB Query Language (MQL). Portanto, o MongoDB, um banco de dados de esquema flexível e orientado a documentos, é a melhor escolha para aplicativos corporativos que exigem *análises em tempo real*.

## Quando usar o MongoDB?

Se os requisitos forem os seguintes:

- Quando precisar de alta disponibilidade de dados com recuperação de dados automática, rápida e instantânea;
- Se não tem um administrador de banco de dados;

- Se a maioria dos serviços são baseados em nuvem, é o mais adequado, pois sua arquitetura de expansão nativa habilitada por 'sharding' se alinha bem com o dimensionamento horizontal e a agilidade oferecidos pela computação em nuvem.

Com uma variedade de bancos de dados disponíveis no mercado, frequentemente, os usuários ficam em dúvida entre o MongoDB e o MySQL, mas para descobrir a melhor opção é importante pesquisar a respeito dos pontos fortes de cada um para entender qual será melhor para as necessidades da organização.

As organizações que utilizam um banco de dados relacional como o MySQL, por exemplo, podem enfrentar certas dificuldades enquanto gerenciam e armazenam seus dados com as mudanças nas exigências. Ao mesmo tempo, novas empresas estão se perguntando que banco de dados escolher para não enfrentarem problemas em seu pipeline de desenvolvimento. Vamos ver, então, as principais diferenças entre o MongoDB e o MySQL.

## 1. MongoDB x MySQL

MySQL usa uma linguagem de consulta estruturada para acessar os dados armazenados. Nesse formato, os esquemas são usados para criar estruturas de banco de dados, utilizando tabelas como forma de padronizar os tipos de dados para que os valores sejam pesquisáveis e possam ser consultados adequadamente. Uma solução madura, o MySQL é útil para uma variedade de situações, incluindo bancos de dados de sites, aplicativos e gerenciamento de produtos comerciais.

Devido à sua natureza rígida, o MySQL é preferível ao MongoDB quando a integridade e o isolamento dos dados são essenciais, como ao gerenciar dados transacionais. Mas o formato menos restritivo e o alto desempenho do MongoDB o tornam uma escolha melhor, principalmente quando a disponibilidade e a velocidade são as principais preocupações.

Dando continuidade aos estudos comparativos, veremos agora o MongoDB *versus* Cassandra. O MongoDB, é um banco de dados baseado em documentos, como já vimos, já o Cassandra é um banco de dados baseado em modelo de colunas. Vamos abordar um pouco mais a respeito das suas principais diferenças.

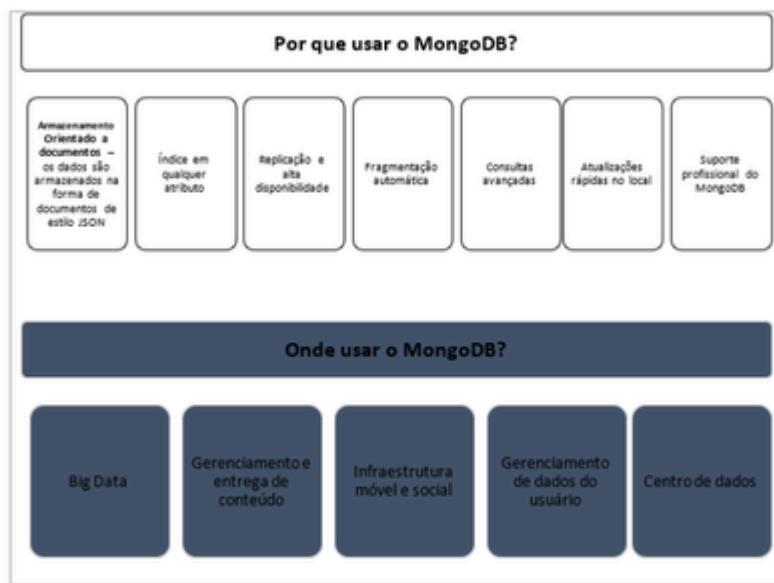
## 2. MongoDB *versus* Cassandra

Embora o Cassandra e o MongoDB sejam considerados bancos de dados NoSQL, eles têm pontos fortes diferentes. O Cassandra usa uma estrutura de tabela tradicional com linhas e colunas, que permite aos usuários manter a uniformidade e a durabilidade ao formatar os dados antes de serem compilados.

O Cassandra pode oferecer uma transição mais fácil para empresas que procuram uma solução NoSQL porque tem uma sintaxe semelhante ao SQL; ele também lida de maneira confiável com a implantação e a replicação sem muita configuração. No entanto, ele não pode igualar a flexibilidade do MongoDB para lidar com conjuntos de dados estruturados e não estruturados ou seu desempenho e confiabilidade para aplicativos em nuvem de missão crítica.

## Resumo visual

# Bancos de Dados Não Relacionais



## Referencia



MONGODB. Company. **MongoDB**, 2022. Disponível em: <https://www.mongodb.com/pt-br/company>. Acesso em: 25 maio 2022.

BAER, T. MongoDB 5.0 is here, spotlighting productivity and extensibility. **ZDnet**, 2021. Disponível em: <https://www.zdnet.com/article/mongodb-5-0-is-here-spotlighting-productivity-and-extensibility/>. Acesso em: 25 maio 2022.

## Unidade 2

Manipulação de Dados em Ambiente NoSQL

### Aula 1

MongoDB básico

#### Introdução



Olá, estudante! Seja muito bem-vindo! Iniciaremos os estudos desta etapa de aprendizagem aprendendo mais detalhes sobre tipos de dados, a fim de entender como estes são inseridos e resgatados no MongoDB. Para determinar os tipos de campos, o mongo shell fornece os operadores `.instanceof` e `.typeof`, como também vários métodos para retornar, seja como uma string ou como um objeto.

Em seguida, vamos compreender como é possível criar documentos numa coleção usando os operadores CRUD do MongoDB por meio de um MongoDB Playground. Nesse contexto, notaremos a possibilidade de utilizar o método `insertOne()` para inserir um documento e o método `insertMany()` para inserir mais de um documento.

Para finalizar, analisaremos o `ObjectId`, recurso em que cada documento na coleção tem um campo `_id` usado para identificar exclusivamente o documento numa coleção específica.

Entenderemos que o campo `_id` atua como a chave primária para os documentos da coleção e pode ser utilizado em qualquer formato. No entanto, o modelo-padrão é o `ObjectId` do documento.

## Tipos – inserindo e resgatando dados

# Bancos de Dados Não Relacionais



Para assimilar corretamente os conteúdos que serão apresentados neste bloco de estudos, é importante considerar que as referências desta aula são relativas ao mongo shell incluído no download do servidor MongoDB. Caso você queira obter mais informações sobre o novo MongoDB shell mongosh, consulte a *Documentação do mongosh*.

É interessante entender que existem diferenças entre os dois shells – mongo shell e mongosh –, as quais podem ser verificadas no próprio site do MongoDB, que exibe uma comparação entre os shells mencionados.

Em resumo, o novo mongosh oferece vantagens quando confrontado com o mongo shell, como: realce de sintaxe, histórico de comandos e registros aprimorados. Contudo, o mongosh está atualmente disponível como uma versão beta. Sendo assim, o produto, seus recursos e a documentação correspondente podem mudar durante a fase de testes.

Dito isso, em nossos estudos vamos seguir com o mongo shell, uma vez que é o modelo utilizado pela plataforma MongoDB no momento presente.

## Verifique os tipos de dados no mongo shell

Para determinar os tipos de campos, o mongo shell fornece os operadores `.instanceof` e `.typeof`.

- **Instanceof:** retorna um booleano para testar se um valor é uma instância de algum tipo.

Por exemplo, a operação exibida a seguir (Código 1) testa se o campo `_id` é uma instância do tipo `ObjectID`:

Código 1 | Operação para testar o campo `_id`

```
mydoc._id instanceof ObjectID
```

Fonte: MongoDB ([s. d.]b).

A operação retornará `true`.

**Typeof:** retornará o tipo de um campo. Por exemplo, a operação a seguir (Código 2) retorna o tipo do campo `_id` :

Código 2 | Operação para retornar um campo

```
typeof mydoc._id
```

Fonte: MongoDB ([s. d.]b).

Nesse caso, `typeof` retornará o tipo de objeto mais genérico do tipo `object` em vez de `ObjectID`.

### Inserindo e resgatando dados

#### Date (em português: *Data*)

O mongo shell fornece vários métodos para retornar a data, seja como uma string ou como um objeto Date:

- `Date()`: método que retorna a data atual como uma string.
- `new Date()`: construtor que retorna um objeto Date usando o `ISODate()`wrapper.
- `ISODate()`: construtor que retorna um objeto Date usando o `ISODate()`wrapper.

Internamente, os objetos **Date** são armazenados como um inteiro de 64 bits assinado, representando o número de milissegundos. Entretanto, nem todas as operações e drivers de banco de dados suportam o intervalo completo de 64 bits. Desse modo, você poderá trabalhar de forma segura com datas cujos anos estejam dentro do intervalo inclusivo de 0 até 9999.

#### Data de retorno como uma string

Para retornar a data como uma string, use o método `Date()`, como no Código 3, a seguir:

Código 3 | Aplicação de string Date()

```
var myDateString = Date();
```

Fonte: MongoDB ([s. d.]b).

Para imprimir o valor da variável, digite o nome da variável no shell, como no Código 4, a seguir:

Código 4 | Imprimir valor de variável

```
myDateString;
```

Fonte: MongoDB ([s. d.]b).

O resultado é o valor de myDateString, como mostra o Código 5:

Código 5 | Resultado do valor myDateString

```
Wed Jun 01 2022 18:03:25 GMT-0500 (EST);
```

Fonte: MongoDB ([s. d.]b).

Para verificar o tipo, use o operador typeof, como indicado no Código 6:

Código 6 | Verificação do tipo

```
typeof myDateString;
```

Fonte: MongoDB ([s. d.]b).

A operação retorna string.

## Retornar Date

O mongo shell envolve objetos do tipo Date com o auxiliar ISODate, porém os objetos permanecem do tipo Date.

O exemplo a seguir (Código 7) usa os construtores new Date() e o ISODate() para retornar o objeto Date:

Código 7 | Retornando o objeto Date

```
var myDate = new Date();
var myDateInitUsingISODateWrapper = ISODate();
```

Fonte: MongoDB ([s. d.]b).

Você também poderá usar o operador new com o construtor ISODate(). Para imprimir o valor da variável, digite o nome da variável no shell, como demonstrado no Código 8:

Código 8 | Imprimindo o valor da variável

```
myDate
```

Fonte: MongoDB ([s. d.]b).

O resultado é o valor Date de myDate encapsulado no auxiliar ISODate(), semelhante ao modelo apresentado no Código 9, a seguir:

Código 9 | Resultado do valor da variável

```
ISODate("2022-06-01T06:01:17.171Z")
```

Fonte: MongoDB ([s. d.]b).

Para verificar o tipo, use o operador instanceof, como mostra o Código 10:

Código 10 | Verificação do tipo

```
myDate instanceof Date  
myDateInitUsingISODateWrapper instanceof Date
```

Fonte: MongoDB ([s. d.]b).

E a operação retornará true para ambos.

## Criando documentos



Você pode criar documentos numa coleção usando os operadores CRUD do MongoDB a partir de um MongoDB Playground:

- Use o método `insertOne()` para inserir um documento.
- Use o método `insertMany()` para inserir mais de um documento.

## Pré-requisitos

- **Crie uma conexão com uma implantação do MongoDB**

Use esta opção se o Atlas hospedar sua implantação ou se você tiver uma cadeia de conexão para uma implantação disponível. Defina como sua conexão é salva com a configuração Default Connection Saving Location, de acordo com os aspectos elencados no Quadro 1, a seguir:

# Bancos de Dados Não Relacionais



Quadro 1 | Definição de conexão com a configuração Default Connection Saving Location

Contexto	Descrição
Global	Salve sua conexão globalmente no VS Code, para que possa ser acessada em qualquer espaço de trabalho.
Workspace	Salve sua conexão em seu espaço de trabalho. Você não pode acessar a conexão de um espaço de trabalho diferente.
Session	Salve a conexão apenas para uma sessão do VS Code. A conexão é perdida quando você fecha o VS Code.

Fonte: MongoDB ([s. d.]b).

- Ative a conexão com a implantação do MongoDB

1. Obtenha sua string de conexão.

- a. Navegue até a visualização de clusters do Atlas.
- b. Clique em Conectar para o cluster desejado.
- c. Clique em Conectar com o MongoDB Compass.
- d. Copie a string de conexão fornecida.

2. Abra a paleta de comandos do Visual Studio Code.

Para colar sua cadeia de conexão e conectar-se ao seu cluster, é preciso abrir a Paleta de Comandos do Visual Studio Code. Caso você não saiba efetuar tal procedimento, basta considerar o passo a passo a seguir:

- a. Execute a ação descrita no Quadro 2, de acordo com seu sistema operacional.

# Bancos de Dados Não Relacionais



Quadro 2 | Definição de conexão com a configuração Default Connection Saving Location

Sistema operacional	Método	Ações
macOS	Atalho de teclado	Pressione Command + Shift + P
Windows e Linux	Atalho de teclado	Pressione Control + Shift + P

Fonte: MongoDB ([s. d.]b).

- b. Na Paleta de Comandos, selecione MongoDB: Abrir página de visão geral.
- c. Na página de visão geral, clique em Conectar com a cadeia de conexão.

**3. Cole sua string de conexão na Paleta de Comandos.**

**4. Pressione a tecla Enter ou Return.**

**5. Ativar uma conexão.**

Você pode conectar o MongoDB no Visual Studio Code, porém é possível realizar apenas uma implantação por vez. Para alterar a conexão ativa por uma implantação diferente ou para se conectar a uma implantação da qual você saiu, é preciso observar os seguintes passos:

- a. No Visual Studio Code, clique na exibição do MongoDB na Barra de atividades.
- b. Clique com o botão direito do mouse na conexão que deseja ativar e depois em Conectar.

**6. Conectar o shell à sua implantação ativa.**

- a. Na lista MongoDB for VS Code Connections, clique com o botão direito do mouse em sua implantação ativa.
- b. Selecione Iniciar MongoDB Shell.
- c. O MongoDB para VS Code abre a janela Terminal no VS Code e inicia o shell conectado à sua implantação selecionada.

## Criar um documento

Para criar um documento, use a sintaxe em seu Playground como mostra o Código 11. Para executar seu Playground, pressione o botão Play no canto superior direito da Visualização do Playground.

Código 11 | Criar documento no MongoDB

```
db.collection.insertOne(  
  <document>,  
  {  
    writeConcern: <document>  
  }  
)
```

Fonte: MongoDB ([s. d.]b).

## Exemplo 1

Para executar, comece com um MongoDB Playground em branco limpando o modelo Playground, se estiver carregado, como no Código 12.

- a. Mude para o banco de dados test.
- b. Insira oito documentos na coleção test.sales.

Código 12 | Criação de documento

```
use("test");

db.sales.insertOne(
  { "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : new
Date("2022-06-01T08:00:00Z") }
);
```

Fonte: MongoDB ([s. d.]b).

Para criar muitos documentos, use a sintaxe apresentada no Código 13 em seu Playground.

Código 13 | Criação de vários documentos

```
db.collection.insertMany(
  [ <document 1> , <document 2>, ... ],
  {
    writeConcern: <document>,
    ordered: <boolean>
  }
)
```

Fonte: MongoDB ([s. d.]b).

## Exemplo 2

Para executar este exemplo, comece com um MongoDB Playground em branco limpando o modelo Playground, se estiver carregado, como mostra o Código 14.

1. Mude para o banco de dados test.
2. Insira oito documentos na coleção test.sales.

Código 14 | Exemplo para criação de vários documentos

```
use("test");

db.sales.insertMany([
  { "_id" : 2, "item" : "abc", "price" : 10, "quantity" : 2, "date" :
new Date("2014-03-01T08:00:00Z") },
  { "_id" : 3, "item" : "jkl", "price" : 20, "quantity" : 1, "date" :
new Date("2014-03-01T09:00:00Z") },
  { "_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 10, "date" :
new Date("2014-03-15T09:00:00Z") },
  { "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 20, "date" :
new Date("2014-04-04T11:21:39.736Z") },
  { "_id" : 6, "item" : "abc", "price" : 10, "quantity" : 10, "date" :
new Date("2014-04-04T21:23:13.331Z") },
  { "_id" : 7, "item" : "def", "price" : 7.5, "quantity": 5, "date" :
new Date("2015-06-04T05:08:13Z") },
  { "_id" : 8, "item" : "def", "price" : 7.5, "quantity": 10, "date" :
new Date("2015-09-10T08:43:00Z") },
  { "_id" : 9, "item" : "abc", "price" : 10, "quantity" : 5, "date" :
new Date("2016-02-06T20:20:13Z") },
]);
```

Fonte: MongoDB ([s. d.]b).

Código 15 | Resultados da criação de vários documentos

```
{  
  acknowledged: 1,  
  insertedIds: {  
    '0': 2,  
    '1': 3,  
    '2': 4,  
    '3': 5,  
    '4': 6,  
    '5': 7,  
    '6': 8,  
    '7': 9  
  }  
}
```

Fonte: MongoDB ([s. d.]b).

Observação: quando você pressiona o botão Play, o MongoDB for VS Code divide seu Playground e gera o documento seguinte no painel Playground Results.json. Se você tiver desativado a exibição dividida, o MongoDB for VS Code produzirá o documento seguinte numa nova guia.

## Aprofundando conhecimentos sobre o ObjectId



Cada documento na coleção tem um campo `_id` usado para identificar exclusivamente o documento numa coleção específica. O campo `_id` atua como a chave primária para os documentos da coleção e pode ser utilizado em qualquer formato. No entanto, o modelo-padrão é o ObjectId do documento (GEEKSFORGEEKS, 2020).

Um ObjectId é um campo do tipo BSON de 12 bytes:

- Os primeiros 4 bytes representam o Unix Timestamp do documento.
- Os próximos 3 bytes são o ID da máquina na qual o servidor MongoDB está sendo executado.
- Os 2 bytes seguintes são do ID do processo.
- O último campo, de 3 bytes, é usado para incrementar o ObjectId.

No MongoDB, cada documento armazenado numa coleção requer um campo `_id` exclusivo que atua como chave primária. Se um documento inserido omitir o campo `_id`, o driver MongoDB gerará automaticamente um ObjectId para o campo `_id`.

Os clientes MongoDB devem adicionar um campo `_id` com um ObjectId exclusivo. O uso de ObjectIDs para o campo `_id` oferece os seguintes benefícios adicionais:

- No mongo shell, você pode acessar o tempo de criação do ObjectId usando o método `ObjectId.getTimestamp()`.

- Classificar num campo `_id` que armazena valores `ObjectID` é aproximadamente equivalente à classificação por tempo de criação.

## Métodos e atributos

O `ObjectID()` abrange os atributos e métodos descritos no Quadro 3, a seguir.

Quadro 3 | Métodos e atributos do `ObjectID`

Atributo/Método	Descrição
<code>str</code>	Retorna a representação de string hexadecimal do objeto.
<code>ObjectID.getTimestamp()</code>	Retorna a parte do carimbo de data/hora do objeto como uma <code>Data</code> .
<code>ObjectID.toString()</code>	Retorna a representação JavaScript na forma de uma string literal <code>"ObjectID(...)"</code> .
<code>ObjectID.valueOf()</code>	Retorna a representação do objeto como uma string hexadecimal. A string retornada é o stratributo.

Fonte: MongoDB ([s. d.]b).

## Gerando um novo `ObjectID`

Para gerar um novo `ObjectID`, use o `ObjectID()` sem argumento, como mostra o Código 16, a seguir.

Código 16 | Criação de um novo ObjectId

```
x = ObjectId()
```

Nesse exemplo, o valor de x seria o mesmo apresentado pelo Código 17.

Código 17 | Valor do ObjectId

```
ObjectId("507f1f77bcf86cd799439011")
```

Fonte: MongoDB ([s. d.]b).

## Especificando uma string hexadecimal

O Código 18, a seguir, explica como é possível gerar um novo ObjectId usando uma string hexadecimal ObjectId() exclusiva.

Código 18 | Criando uma string hexadecimal

```
y = ObjectId("507f191e810c19729de860ea")
```

Fonte: MongoDB ([s. d.]b).

Nesse exemplo, o valor de y é o mesmo descrito no Código 19.

Código 19 | Valor de y criado a partir de uma string hexadecimal

```
ObjectId("507f191e810c19729de860ea")
```

Fonte: MongoDB ([s. d.]b).

## Acessando a string hexadecimal

Para acessar o atributo str de um objeto ObjectId(), siga o exemplo apresentado no Código 20.

Código 20 | Acessando a string hexadecimal

```
ObjectId("507f191e810c19729de860ea").str
```

Fonte: MongoDB ([s. d.]b).

Tal operação retornará a string hexadecimal exibida no Código 21.

Código 21 | Acessando a string hexadecimal

```
507f191e810c19729de860ea
```

Fonte: MongoDB ([s. d.]b).

## **ObjectId.getTimestamp()**

O ObjectId.getTimestamp() retorna a parte do carimbo de data/hora ObjectId() como uma Data. O exemplo a seguir (Código 22) chama o getTimestamp() método em um ObjectId.

Código 22 | Aplicação do ObjectId.getTimestamp()

```
ObjectId("507c7f79bcf86cd7994f6c0e").getTimestamp()
```

Fonte: MongoDB ([s. d.]b).

Esse procedimento retornará a saída demonstrada no Código 23.

Código 23 | Saída do exemplo de aplicação do ObjectId.getTimestamp()

```
ISODate("2012-10-15T21:26:17Z")
```

Fonte: MongoDB ([s. d.]b).

## ObjectId.toString()

O ObjectId.toString() retorna à representação de string do ObjectId(). Tal valor de string tem o formato de ObjectId(...).

O exemplo a seguir (Código 24) chama o método toString() em uma instância ObjectId() no mongo shell.

Código 24 | Exemplo de aplicação do ObjectId.toString()

```
ObjectId("507c7f79bcf86cd7994f6c0e").toString()
```

Fonte: MongoDB ([s. d.]b).

Tal operação retornará a string indicada no Código 25.

Código 25 | Exemplo da saída de aplicação do ObjectId.toString()

```
ObjectId("507c7f79bcf86cd7994f6c0e")
```

Fonte: MongoDB ([s. d.]b).

Você pode confirmar o tipo desse objeto usando a operação apresentada no Código 26.

Código 26 | Exemplo de operação da aplicação do ObjectId.toString()

```
typeof ObjectId("507c7f79bcf86cd7994f6c0e").toString()
```

Fonte: MongoDB ([s. d.]b).

## **ObjectId.valueOf()**

O ObjectId.valueOf() retorna o valor de ObjectId() como uma string hexadecimal minúscula. Tal valor é o atributo str do objeto ObjectId().

O exemplo a seguir (Código 27) chama o método valueOf() em um ObjectId().

Código 27 | Aplicação do ObjectId.valueOf()

```
ObjectId("507c7f79bcf86cd7994f6c0e").valueOf()
```

Fonte: MongoDB ([s. d.]b).

Esse procedimento retornará a string demonstrada no Código 28.

Código 28 | Aplicação do ObjectId.valueOf()

```
507c7f79bcf86cd7994f6c0e)
```

Fonte: MongoDB ([s. d.]b).

Você pode confirmar o tipo desse objeto usando a operação exibida no Código 29.

Código 29 | Confirmação do tipo do objeto com o ObjectId.valueOf()

```
typeof ObjectId("507c7f79bcf86cd7994f6c0e").valueOf()
```

Fonte: MongoDB ([s. d.]b).

## Videoaula: MongoDB básico



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Para que você aprofunde seus conhecimentos sobre o MongoDB, é importante entender que as operações básicas de manipulação de dados são representadas pela sigla CRUD, acrônimo dos termos em inglês Create (Criar), Read (Ler), Update (Atualizar) e Delete (Apagar), respectivamente. Trata-se de procedimentos do banco de dados que não podem ser ignorados durante a criação de uma aplicação. Por exemplo, qualquer aplicação moderna deve possuir a capacidade de criar um novo usuário, ler os dados desse consumidor, atualizar suas informações e, se necessário, excluir sua conta.

## Saiba mais



# Bancos de Dados Não Relacionais



Caso você queira garantir mais informações sobre bancos de dados não relacionais, consulte os conteúdos indicados a seguir. Vale a pena conferir!

["Portal MongoDB"](#)

["MongoDB: o que é e como usar o banco de dados NoSQL?"](#)

## Referências



CONNECT to your MongoDB deployment. **MongoDB**, [s. d.]d. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/connect/#std-label-vsce-connect-task>.

Acesso em: 2 maio 2022.

CREATE Documents. **MongoDB**, [s. d.]c. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/create-document-playground/>. Acesso em: 2 maio 2022.

DATA types in the mongo shell. **MongoDB**, [s. d.]b. Disponível em:

[https://www.mongodb.com/docs/v4.4/core/shell-types/?\\_ga=2.123168715.1904519913.1654112629-1303898093.1652445128&\\_gac=1.262226814.1652839811.CjwKCAjwj42UBhAAEiwACIhADrO3NZmq-qxw0opJDgn017KI6vWMcpvfSGwWLQsBTiT-pna8cIBpuBoCz3EQAvD\\_BwE#date](https://www.mongodb.com/docs/v4.4/core/shell-types/?_ga=2.123168715.1904519913.1654112629-1303898093.1652445128&_gac=1.262226814.1652839811.CjwKCAjwj42UBhAAEiwACIhADrO3NZmq-qxw0opJDgn017KI6vWMcpvfSGwWLQsBTiT-pna8cIBpuBoCz3EQAvD_BwE#date). Acesso em: 2 maio 2022.

THE mongo shell. **MongoDB**, [s. d.]a. Disponível em:

<https://www.mongodb.com/docs/v4.4/mongo/#std-label-compare-mongosh-mongo>. Acesso em: 2 maio 2022.

WHAT is ObjectId in MongoDB. **GeeksforGeeks**, 1 nov. 2020. Disponível em: <https://www.geeksforgeeks.org/what-is-objectid-in-mongodb/>. Acesso em: 2 maio 2022.

## Aula 2

CRUD no MongoDB

### Introdução



Os estudos desta etapa de aprendizagem começarão com uma análise das operações CRUD aplicadas no MongoDB, acrônimo formado pelas iniciais das palavras criação (adiciona novos *documentos* a uma *coleção*), leitura (usada para recuperar documentos), atualização (utilizada para atualizar ou modificar o documento existente na coleção) e exclusão (usada para excluir ou remover os documentos presentes numa coleção).

No MongoDB, o método `insert()` insere um ou mais documentos na coleção. Estudaremos esse procedimento de forma mais detalhada durante o desenvolvimento do conteúdo. São necessários dois parâmetros: o primeiro é o documento ou *array* do documento que queremos inserir; e o segundo trata de aspectos opcionais. Para buscar e atualizar os dados, podemos utilizar a biblioteca PyMongo, ferramenta que traz agilidade a tais processos.

## CRUD e sua aplicação com o MongoDB



Podemos utilizar o MongoDB para inúmeras aplicações, as quais abrangem desde a construção de um aplicativo (incluindo web e mobile) até a análise de dados ou mesmo a administração de um banco de dados. Diante disso, para todos os casos de uso é necessário interagir com o servidor MongoDB, já que a partir dele será possível efetuar operações como inserir ou excluir novos dados no aplicativo ou até mesmo realizar a leitura dos dados do aplicativo.

O MongoDB fornece um conjunto de operações básicas, mas também essenciais, que ajudarão você a interagir facilmente com o servidor. Tais procedimentos são conhecidos como operações **CRUD**, acrônimo que corresponde aos termos em inglês *Create, Read, Update* e *Delete*, como mostra a Figura 1 (GEEKSFORGEEKS, 2020).

Figura 1 | Significado das operações CRUD – MongoDB



Fonte: GeeksforGeeks (2020).

Vejamos, a seguir, como funcionam as operações CRUD no MongoDB:

## 1. Operações de criação (*Create*)

As operações de criação ou inserção adicionam novos *documentos* a uma *coleção*. Se a coleção não existir no momento, as operações de inserção a criarão (GEEKSFORGEEKS, 2020). O MongoDB disponibiliza os métodos apresentados no Quadro 1, a seguir, para inserir documentos em uma coleção.

Quadro 1 | Métodos para inserir documentos em uma coleção

Método	Descrição
• <code>db.collection.insertOne()</code>	É usado para inserir um único documento na coleção.
• <code>db.collection.insertMany()</code>	É usado para inserir vários documentos na coleção.

Fonte: GeeksforGeeks (2020).

No MongoDB, as operações de inserção visam a uma única coleção. Todas as operações de gravação no MongoDB são atômicas no nível de um único documento (nesse caso, todos ou nenhum dos aspectos da série serão alterados, isto é, não existe uma mudança parcial do banco de dados em operações atômicas). Vamos analisar, a seguir, alguns exemplos de aplicação.

**Exemplo 1:** inserir detalhes de um único aluno na forma de documento na coleção de alunos usando o método `db.collection.insertOne()`, como podemos ver no Código 1.

Código 1 | Operação para aplicar o método db.collection.insertOne()

```
1  db.student.insertOne({  
2      name: "Jon",  
3      age: 20,  
4      course: "MongoDB",  
5      mode: "online",  
6      })  
7  {  
8      "acknowledged" : true,  
9      "insertedId" : ObjectId ("5e549cdc80e6fa3gj48dban")  
10 }
```

Fonte: GeeksforGeeks (2020).

Exemplo 2: inserir detalhes de vários alunos na forma de documentos na coleção de alunos usando o método db.collection.insertMany(), como mostra o Código 2:

Código 2 | Operação para aplicar o método db.collection.insertMany()

```
1  db.student.insertMany([  
2      {  
3          name: "Jon",  
4          age: 20,  
5          course: "MongoDB",  
6          mode: "online"  
7      },  
8      {  
9          name: "Ana",  
10         age: 21,  
11         course: "MongoDB",  
12         mode: "online"  
13     }  
14 ])
```

```
12  ])
13  {
14      "acknowledged" : true,
15      "insertedIds" : [
16          ObjectId ("5e549cdc80e6fa3gj48dban"),
17          ObjectId ("5e549cdc80e6fa3gj48dbf9")
18      ]
19
20
21
```

Fonte: GeeksforGeeks (2020).

## 2. Operações de leitura (*Read*)

As operações de leitura são usadas para recuperar documentos da coleção ou, em outras palavras, utilizam-se as operações de leitura para consultar uma coleção de um documento (GEEKSFORGEEKS, 2020). Podemos executar a operação de leitura aplicando o método apresentado no Quadro 2, a seguir.

Quadro 2 | Método para realizar a leitura de documentos

Método	Descrição
• <code>db.collection.find()</code>	É usado para inserir um único documento na coleção.
• <code>db.collection.insertMany()</code>	É usado para inserir vários documentos na coleção.

Fonte: GeeksforGeeks (2020).

Vejamos, a seguir, um exemplo de aplicação.

**Exemplo 3:** recuperando os detalhes dos alunos da coleção de alunos usando o método `db.collection.find()`, como podemos ver no Código 3.

Código 3 | Operação para aplicar o método db.collection.find()

```
1  db.student.find().pretty()
2  {
3      "_id" : ObjectId ("5e549cdc80e6fa3gj48dban"),
4      name: "Jon",
5      age: 20,
6      course: "MongoDB",
7      mode: "online"
8  }
9  {
10     "_id" : ObjectId ("5e549cdc80e6fa3gj48dbf9"),
11     name: "Ana",
12     age: 21,
13     course: "MongoDB",
14     mode: "online"
15 }
```

Fonte: GeeksforGeeks (2020).

### 3. Operações de atualização (*Update*)

As operações de atualização são usadas para atualizar ou modificar o documento existente na coleção (GEEKSFORGEEKS, 2020). Podemos executar tal procedimento usando os métodos apresentados no Quadro 3, a seguir.

Quadro 3 | Métodos para realizar a atualização de documentos

Método	Descrição
• <code>db.collection.updateOne()</code>	É usado para atualizar um único documento na coleção que satisfaça os critérios fornecidos.
• <code>db.collection.updateMany()</code>	É usado para atualizar vários documentos na coleção que atendam aos critérios fornecidos.
• <code>db.collection.replaceOne()</code>	É usado para substituir um único documento na coleção que satisfaça os critérios fornecidos.

Fonte: GeeksforGeeks (2020).

Vamos observar, a seguir, um exemplo de aplicação.

**Exemplo 4:** atualizando a idade da aluna Ana na coleção de alunos usando o método `db.collection.updateOne()`, como mostra o Código 4.

Código 4 | Operação para aplicar o método db.collection.updateOne()

```
1  db.student.updateOne({name: "Ana"}, {$set: age:25})
2  {"acknowledged": true, "matchedCount":1, "modifiedCount": 0}
3
4  db.student.find().pretty()
5  {
6      "_id" : ObjectId ("5e549cdc80e6fa3gj48dban"),
7      name: "Jon",
8      age: 20,
9      course: "MongoDB",
10     mode: "online"
11 }
12 {
13     "_id" : ObjectId ("5e549cdc80e6fa3gj48dbf9"),
14     name: "Ana",
15     age: 25,
16     course: "MongoDB",
17     mode: "online"
18 }
19
20
```

Fonte: GeeksforGeeks (2020).

#### 4. Operações de exclusão (*Delete*)

As operações de exclusão são usadas para excluir ou remover os documentos existentes em uma coleção (GEEKSFORGEEKS, 2020). Podemos executar as operações de exclusão usando os métodos elencados no Quadro 4, a seguir.

Quadro 4 | Métodos para realizar a exclusão de documentos

Método	Descrição
• <code>db.collection.deleteOne()</code>	É usado para excluir um único documento da coleção que satisfaça os critérios fornecidos.
• <code>db.collection.deleteMany()</code>	É usado para excluir vários documentos da coleção que atendam aos critérios fornecidos.

Fonte: GeeksforGeeks (2020).

Vejamos, a seguir, um exemplo de aplicação.

**Exemplo 5:** excluindo um documento da coleção de alunos usando o método `db.collection.deleteOne()`, como indica o Código 5.

Código 5 | Operação para aplicar o método db.collection.deleteOne()

```
1  db.student.find().pretty()
2  {
3      "_id" : ObjectId ("5e549cdc80e6fa3gj48dban"),
4      name: "Jon",
5      age: 20,
6      course: "MongoDB",
7      mode: "online"
8  }
9  {
10     "_id" : ObjectId ("5e549cdc80e6fa3gj48dbf9"),
11     name: "Ana",
12     age: 25,
13     course: "MongoDB",
14     mode: "online"
15 }
16 db.student.deleteOne({name: "Ana"})
17 {"acknowledged": true, "deletedCount" : 1}
18 db.student.find().pretty()
19 {
20     "_id" : ObjectId ("5e549cdc80e6fa3gj48dban"),
21     name: "Jon",
22     age: 20,
23     course: "MongoDB",
24     mode: "online"
25 }
```

Fonte: GeeksforGeeks (2020).

Ao longo dos anos, o MongoDB tornou-se uma preferência popular por ser um banco de dados altamente escalável. Atualmente vem sendo usado como plataforma de armazenamento dos dados de back-end de muitas organizações renomadas, como: IBM, Twitter, Facebook, Google, entre outras. Portanto, estudar as principais operações desse software é de fundamental importância para as aplicações do cotidiano.

## Inserindo dados



No MongoDB, o método `insert()` insere um ou mais documentos na coleção. Além disso, são necessários dois parâmetros: o primeiro é o documento ou *array* do documento que queremos inserir; e o segundo trata de aspectos opcionais, o que aprenderemos de forma mais detalhada a seguir.

Usando esse método, também podemos criar uma coleção inserindo documentos. É possível inserir documentos com ou sem o campo `_id`. Se você inserir um documento na coleção sem o campo `_id`, o MongoDB adicionará automaticamente um campo `_id` e o atribuirá a um `ObjectId` exclusivo. Por outro lado, se você inserir um documento com o campo `_id`, o valor do campo `_id` deverá ser único para evitar o erro de chave duplicada. Para concluir, vale destacar que esse método também pode ser usado em transações de vários documentos (MONGODB, [s. d.]a). O método `insert()` apresenta a sintaxe indicada no Código 6.

Código 6 | Parâmetro do método insert()

```
1 db.collection.insert(  
2     <document or array of documents>,  
3     {  
4         writeConcern: <document>,  
5         ordered: <boolean>  
6     }  
7 )
```

Fonte: GeeksforGeeks (2020).

- **Parâmetros:**

- document: o primeiro parâmetro é o documento ou uma matriz de documentos. Documentos são uma estrutura criada de pares de arquivos e valores semelhantes aos objetos JSON.
- O segundo parâmetro é opcional.

- **Parâmetros opcionais:**

- writeConcern: é usado apenas quando você não deseja utilizar a preocupação de gravação-padrão. O tipo desse parâmetro é um documento.
- Ordered: o valor-padrão desse parâmetro é true. Se for verdade, insere os documentos de maneira ordenada. Caso contrário, insere documentos aleatoriamente.

O método insert() **retorna** um objeto que contém o status da operação.

- Esse método retorna WriteResult quando você insere um único documento na coleção.
- Além disso, retorna BulkWriteResult quando você insere vários documentos na coleção.

Vejamos alguns exemplos de aplicação do método de inserir dados.

**Exemplo 6:** para desenvolver este exemplo, trabalharemos com:

- **Banco de dados:** boletim.
- **Coleção:** estudante.

- **Documento:** nenhum documento inicialmente, porém vamos inserir no formulário o nome do aluno e as notas da prova (*test score*).
  - a. **Inserindo apenas um documento:** no Código 7, apresentado a seguir, veremos como é possível inserir um documento.

Código 7 | Aplicação do parâmetro do método insert().

```
1 > db.collection.insert({Name: "Luiz", Test Score: 500})
2   WriteResult({nInserted:1})
3
4 > db.student.find().pretty()
5 {
6   "_id": ObjectId("800ea67c0cg247879ba96498"),
7   "Name": "Luiz",
8   "Test Score": 500
9 }
```

Fonte: adaptado de GeeksforGeeks (2020).

- b. **Inserindo vários documentos na coleção:** no Código 8, vamos entender como é possível inserir vários documentos. Tal processo é executado passando uma matriz de documentos no método insert().

Código 8 | Aplicação do parâmetro do método insert() para vários

```
1 > db.student.insert([{"Name": "Maria", Test Score: 600},  
2                      {"Name": "Francisco", Test Score: 400},  
3                      {"Name": "Elisa", Test Score: 788}])
```

Fonte: adaptado de GeeksforGeeks (2020).

Teremos como resultado o que se apresenta no Código 9.

Código 9 | Resultado da aplicação do parâmetro do método insert()

```
1 > db.student.insert([{"Name": "Maria", Test Score: 600},  
2                      {"Name": "Francisco", Test Score: 400},  
3                      {"Name": "Elisa", Test Score: 788}])  
4  
5 > db.collection.insert({"Name": "Luiz", Test Score: 500})  
6 WriteResult({nInserted:1})  
7  
8 > db.student.find().pretty()  
9 {  
  "_id": ObjectId ("800ea67c0cg247879ba96498"),  
  "Name": "Luiz",  
  "Test Score": 500  
}  
{  
  "_id": ObjectId ("800ea67c0cg247879ba9649a"),  
  "Name": "Maria",  
  "Test Score": 600  
}  
{  
  "_id": ObjectId ("800ea67c0cg247879ba9649b"),  
  "Name": "Francisco",  
  "Test Score": 400  
}  
{  
  "_id": ObjectId ("800ea67c0cg247879ba9649b"),  
  "Name": "Elisa",  
  "Test Score": 788  
}
```

Fonte: adaptado de GeeksforGeeks (2020).

c. Inserindo um documento ao especificar um campo `_id`: no exemplo a seguir, o documento passado pelo método `insert()` inclui o campo `_id`. O valor de `_id` deve ser exclusivo na coleção para evitar erros de chave duplicada. O Código 10, a seguir, demonstra como inserir um documento na coleção do aluno com o campo `_id`. Posteriormente, no Código 11, será possível visualizar o resultado da aplicação como campo `_id`.

Código 10 | Inserindo um documento na coleção com o campo `_id`

```
1 > db.student.insert({_id: 105, Name: "Carla", Test Score:  
550})
```

Fonte: adaptado de GeeksforGeeks (2020).

Código 11 | Resultado da aplicação com o campo `_id`.

```
1  > db.student.insert([{"Name": "Maria", Test Score: 600},  
2           {"Name": "Francisco", Test Score: 400},  
3           {"Name": "Elisa", Test Score: 788}])'  
4  
5  > db.collection.insert({Name: "Luiz", Test Score: 500})  
6  WriteResult({"nInserted":1})  
7  
8  > db.student.find().pretty()  
9  {  
10    "_id": ObjectId ("800ea67c0cg247879ba96498"),  
11    "Name": "Luiz",  
12    "Test Score": 500  
13  }  
14  {  
15    "_id": ObjectId ("800ea67c0cg247879ba9649a"),  
16    "Name": "Maria",  
17    "Test Score": 600  
18  }  
19  
20  
21  
22  
23  
24  {  
25    "_id": ObjectId ("800ea67c0cg247879ba9649b"),  
26    "Name": "Francisco",  
27    "Test Score": 400  
28  }  
29  {  
30    "_id": 105, Name: "Carla", Test Score: 550}  
31  
32  
33  
34  
35  
36  
37
```

Fonte: adaptado de GeeksforGeeks (2020).

Por meio deste conteúdo, pudemos perceber como é fácil inserir documentos ou dados no MongoDB. Destaco, nesse contexto, uma observação importante: caso queira inserir um

documento, você também pode utilizar o db.post.save(document). Se não especificar o \_id dentro do documento, o método save() trabalhará do mesmo modo que o método insert(). Se você especificar o \_id, ele substituirá os dados do documento que contém o \_id especificado no método save() (PYMONGO, [s. d.]).

## Buscando e Atualizando dados



O MongoDB é um banco de dados multiplataforma orientado a documentos que funciona no conceito de coleções e documentos, oferecendo alta velocidade, disponibilidade e escalabilidade. O PyMongo fornece vários métodos para buscar os dados do MongoDB. Vamos estudá-los a seguir.

**1. Find One:** mecanismo usado para buscar dados da coleção no MongoDB. Retorna a primeira ocorrência. A sintaxe é escrita como mostra o Código 12, a seguir.

Código 12 | Aplicação do método find one

```
1 > find_one()
```

Fonte: adaptado de GeeksforGeeks (2020).

O próximo exemplo de aplicação (Código 13) tem a intenção de encontrar um dado. O método `find_one()`, nesse caso, retorna à primeira ocorrência na seleção.

Código 13 | Aplicação do método find one

```
1 import pymongo
2 client = pymongo.MongoClient("mongodb://localhost:27017/")
3
4 # Database Name
5 db = client["database"]
6
7 # Collection Name
8 col = db["GeeksForGeeks"]
9
10 x = col.find()
11
12 for data in x:
13     print(data)
```

Fonte: W3Schools (2018).

Teremos como resultado a saída indicada na Figura 2, a seguir.

Figura 2 | Saída da aplicação do método find\_one()

```
{'_id': ObjectId('5ebab3be1f2d57503239a207'), 'appliance': 'fan', 'quantity': 25, 'rating': '3 stars', 'company': 'havells'}
```

Fonte: W3Schools (2018).

2. **Find All:** método utilizado para buscar todas as ocorrências da coleção no MongoDB. Na seleção, use o método find(). A sintaxe é escrita como mostra o Código 14.

Código 14 | Aplicação do método find()

```
1 > find()
```

Fonte: adaptado de GeeksforGeeks (2020).

O Código 15, a seguir, exibe um exemplo de aplicação para buscar as ocorrências.

Código 15 | Exemplo de aplicação do método find()

```
1 import pymongo
2
3
4 client = pymongo.MongoClient("mongodb://localhost:27017/")
5
6 # Database Name
7 db = client["database"]
8
9 # Collection Name
10 col = db["GeeksForGeeks"]
11
12 x = col.find()
13
14 for data in x:
15     print(data)
```

Fonte: W3Schools (2018).

Como resultado, obteremos a saída apresentada na Figura 3.

Figura 3 | Saída da aplicação do método find()

```
{'_id': ObjectId('5ebab3be1f2d57503239a207'), 'appliance': 'fan', 'quantity': 25, 'rating': '3 stars', 'company': 'havells'}
{'_id': ObjectId('5ebab3be1f2d57503239a208'), 'appliance': 'cooler', 'quantity': 15, 'rating': '4 stars', 'company': 'symphony'}
{'_id': ObjectId('5ebab3be1f2d57503239a209'), 'appliance': 'ac', 'quantity': 20, 'rating': '5 stars', 'company': 'voltas'}
{'_id': ObjectId('5ebab3be1f2d57503239a20a'), 'appliance': 'tv', 'quantity': 12, 'rating': '3 stars', 'company': 'sony'}
```

Fonte: W3Schools (2018).

**3. Buscando campos específicos:** se você quiser buscar apenas alguns campos, então, no método find, passe o primeiro parâmetro como {} e o segundo como 1, para os campos que você quer buscar, e 0, para os que você não quer buscar. A sintaxe é escrita como mostra o Código 16, a seguir.

Código 16 | Aplicação do método find() para campos específicos

```
1 > find({}, {field_data:bool})
```

Fonte: adaptado de GeeksforGeeks (2020).

O Código 17, a seguir, traz um exemplo de aplicação para buscar as ocorrências em campos específicos.

Código 17 | Exemplo de aplicação do método find() para campos específicos

```
1 import pymongo
2
3
4 client = pymongo.MongoClient("mongodb://localhost:27017/")
5
6 # Database Name
7 db = client["database"]
8
9 # Collection Name
10 col = db["GeeksForGeeks"]
11
12 # Fields with values as 1 will
13 # only appear in the result
14 x = col.find({}, {'_id': 0, 'appliance': 1,
15 'rating': 1, 'company': 1})
16
17 for data in x:
18     print(data)
```

Fonte: W3Schools (2018).

Teremos como saída o que apresenta a Figura 4, a seguir.

Figura 4 | Saída da aplicação do método find()

```
{'appliance': 'fan', 'rating': '3 stars', 'company': 'havells'}  
{'appliance': 'cooler', 'rating': '4 stars', 'company': 'symphony'}  
{'appliance': 'ac', 'rating': '5 stars', 'company': 'voltas'}  
{'appliance': 'tv', 'rating': '3 stars', 'company': 'sony'}
```

Fonte: W3Schools (2018).

Os comandos de **atualização** nos ajudam a atualizar os dados da consulta já inseridos na coleção do banco de dados MongoDB.

**Métodos usados:** update\_one() e update\_many().

Após inserir os dados no MongoDB, veremos um exemplo de atualização dos dados de um determinado funcionário com id:21, como mostra o Código 18.

Código 18 | Exemplo de aplicação do método para atualização de dados

```
1 # Python code to illustrate
2 # updating data in MongoDB
3 # with Data of employee with id:21
4 from pymongo import MongoClient
5
6 try:
7     conn = MongoClient()
8     print("Connected successfully!!!")
9 except:
10     print("Could not connect to MongoDB")
11
12 # database
13 db = conn.database
14
15 # Created or Switched to collection names: my_gfg_collection
16 collection = db.my_gfg_collection
17
18 # update all the employee data whose eid is 21
19 result = collection.update_many(
20     {"eid":21},
21     {
22         "$set":{
23             "name":"Mr.Geeksforgeeks"
24         },
25         "$currentDate":{"lastModified":True}
26     }
27 )
28
29 print("Data updated with id",result)
30
31 # Print the new record
32 cursor = collection.find()
33 for record in cursor:
34     print(record)
```

Fonte: W3Schools (2018).

Obteremos como resultado o que apresenta o Código 19, a seguir.

Código 19 | Resultado da atualização da idade do funcionário – exemplo aplicado

```
1  Conectado com sucesso!!!
2  Dados atualizados com id
3  {'_id': ObjectId('5a02227b37b8552becf5ed2a'),
4  'name': 'Mr.Geeksforgeeks', 'eid': 21, 'local':
5  'delhi', 'lastModified': datetime.datetime(2017, 11, 7, 21, 19, 9,
6  698000)}
7  {'_id': ObjectId('5a02227c37b8552becf5ed2b'), 'nome':
8  'Mr.Shaurya', 'eid': 14, 'local': 'delhi'}
```

Fonte: W3Schools (2018).

Para garantir que documentos em massa sejam atualizados ao mesmo tempo no MongoDB, você precisará usar a opção “multi”. Caso contrário, por padrão, apenas um documento será modificado por vez.

O exemplo a seguir mostrará como é possível atualizar muitos documentos. Nesse caso, primeiro vamos encontrar o documento que sinaliza o ID do funcionário como “1” e alterar o nome do funcionário de “João Luiz” para “João Neto”.

- **Etapa 1:** emita o comando de atualização.
- **Etapa 2:** escolha a condição que deseja usar para decidir qual documento precisa ser atualizado. Em nosso exemplo, queremos que o documento que possui o ID do funcionário “22” seja atualizado.
- **Etapa 3:** escolha quais Nomes de campo você deseja modificar e insira seu novo valor de acordo com o Código 20, apresentado a seguir.

Código 20 | Aplicação do exemplo da atualização da idade do funcionário

```
1 db.Employee.update
2 (
3     {
4         Employeeid : 1
5     },
6     {
7         $set :
8         {
9             "EmployeeName" : "Joao Neto",
10            "Employeeid" : 22
11        }
12    )
13 )
```

Fonte: W3Schools (2018).

Se o comando for executado com sucesso e você utilizar o comando “find” para procurar o documento com o ID do funcionário como 22, será possível visualizar a saída demonstrada na Figura 5, a seguir.

Figura 5 | Resultado da aplicação do exemplo da atualização

```
C:\Windows\system32\cmd.exe - mongoDB
> db.Employee.update(
... { "Employeeid" : 1 },
... { $set :
... { "EmployeeName" : "NewMartin" , "Employeeid" : 22
... }})
writeResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Employee.find({ "Employeeid" : 22 }).forEach(printjson)
{
    "_id" : ObjectId("563479cc8a8a4246bd27d784"),
    "Employeeid" : 22,
    "EmployeeName" : "Joao Neto "
}
>
```

Atualização efetuada

Fonte: W3Schools (2018).

Como aprendemos, o PyMongo é uma distribuição Python que contém ferramentas para trabalhar com MongoDB, sendo um recurso bastante recomendado para o trabalho com esse software de banco de dados. Dentre as principais vantagens do PyMongo, destacam-se a sua grande comunidade de usuários e o incentivo contínuo a contribuições, o que é extremamente relevante para que tal ferramenta seja sempre atualizada.

## Videoaula: CRUD no MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

A biblioteca PyMongo pode ser utilizada tanto por meio do MongoDB como a partir da Python. Suporta arrays associados como dicionários em Python e drivers Python integrados para conectar o aplicativo Python ao banco de dados. É projetada para Big Data e pode ser

Disciplina

# Bancos de Dados Não Relacionais

facialmente implantada no MongoDB. Vamos aprofundar nossos conhecimentos sobre essa biblioteca no vídeo a seguir, que traz um guia de instalação do PyMongo no MongoDB.

## Saiba mais



Caso você queira garantir mais informações sobre bancos de dados não relacionais, consulte o conteúdo indicado a seguir. Vale a pena conferir!

[\*\*"Vantagens de um banco de dados NoSQL, MongoDB"\*\*](#)

---

## Referências



CONNECT to your MongoDB deployment. **MongoDB**, [s. d.]c. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/connect/#std-label-vsce-connect-task>.

Acesso em: 2 maio 2022.

CREATE Documents. **MongoDB**, [s. d.]a. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/create-document-playground/>. Acesso em: 2 maio 2022.

DATA types in the mongo shell. **MongoDB**, [s. d.]b. Disponível em:

[https://www.mongodb.com/docs/v4.4/core/shell-types/?  
\\_ga=2.123168715.1904519913.1654112629-1303898093.1652445128&\\_gac=1.262226814.1652839811.CjwKCAjwj42UBhAAEiwACIhADr03NZmq-qxw0opJDgnO17KI6vWMcpvfSGwWLQsBTiT-pna8cIBpuBoCz3EQAvD\\_BwE#date](https://www.mongodb.com/docs/v4.4/core/shell-types/?_ga=2.123168715.1904519913.1654112629-1303898093.1652445128&_gac=1.262226814.1652839811.CjwKCAjwj42UBhAAEiwACIhADr03NZmq-qxw0opJDgnO17KI6vWMcpvfSGwWLQsBTiT-pna8cIBpuBoCz3EQAvD_BwE#date)

Acesso em: 2 maio 2022.

PYMONGO 4.1.1 Documentation. **PyMongo**, [s. d.]. Disponível em:

<https://pymongo.readthedocs.io/en/stable/#:~:text=PyMongo%20is%20a%20Python%20distribution,how%20to%20get%20the%20distribution>. Acesso em: 13 jun. 2022.

PYTHON MongoDB Find. **W3Schools**, 7 jun. 2018. Disponível em:

[https://www.w3schools.com/python/python\\_mongodb\\_find.asp](https://www.w3schools.com/python/python_mongodb_find.asp). Acesso em: 13 jun. 2022.

WHAT is ObjectId in MongoDB. **GeeksforGeeks**, 1 nov. 2020. Disponível em:

<https://www.geeksforgeeks.org/what-is-objectid-in-mongodb/>. Acesso em: 2 maio 2022.

## Aula 3

Modelagem e manipulação no MongoDB

## Introdução



Vamos iniciar os estudos desta etapa de aprendizagem falando sobre a modelagem de dados no MongoDB. O principal desafio na modelagem de dados é equilibrar as necessidades do aplicativo, as características de desempenho do mecanismo de banco de dados e os padrões de recuperação de dados. Ao projetar modelos de dados, sempre considere o uso dos dados pelo aplicativo (ou seja, consultas, atualizações e processamento dos dados), bem como a estrutura inerente aos próprios dados.

Mais à frente, analisaremos os diferentes tipos de relacionamento na modelagem de dados do MongoDB, que são três: um para um, um para muitos e muitos para muitos. Neste material, vamos investigar cada um desses modelos de relacionamento de maneira detalhada.

## Modelando entidade e relacionamento



O principal desafio na modelagem de dados é equilibrar as necessidades do aplicativo, as características de desempenho do mecanismo de banco de dados e os padrões de recuperação de dados. Ao projetar modelos de dados, sempre considere o uso dos dados pelo aplicativo (ou seja, consultas, atualizações e processamento dos dados), bem como a estrutura inerente aos próprios dados (MONGODB, 2016).

Diferentemente dos bancos de dados relacionais ou SQL, nos quais você precisa determinar e declarar o esquema de uma tabela antes de inserir propriamente os dados, as coleções do MongoDB, por padrão, não exigem que seus documentos tenham o mesmo esquema, isto é:

- Os documentos numa única coleção não precisam ter o mesmo conjunto de campos, e o tipo de dados de um campo pode diferir entre os documentos de uma coleção.
- Para alterar a estrutura dos documentos numa coleção, como adicionar novos campos, remover campos existentes ou alterar os valores dos campos para um novo tipo, atualize os documentos para a nova estrutura.

Essa flexibilidade facilita o mapeamento de documentos para uma **entidade** ou **objeto**. Cada documento pode corresponder aos campos de dados da entidade representada, mesmo que o documento tenha variação substancial de outros documentos da coleção.

O aspecto mais importante que precisamos aprender no banco de dados MongoDB é o **relacionamento**. Nesse software, temos dois métodos para criar um relacionamento:

- Relações referenciadas.
- Relações incorporadas.

### Relações referenciadas

São conhecidas, também, como o formulário normalizado no qual todos os documentos relacionados são mantidos separados, como mostra a Figura 1.

Figura 1 | Exemplo de formulário referenciado



Fonte: Khan (2022).

Na imagem anterior, pudemos notar dois comentários, mas na prática do dia a dia pode haver mais. Esses documentos são separados uns dos outros. Mas como podemos saber qual comentário pertence a qual artigo? Aqui, a referência vem cumprir seu papel (KHAN, 2022). Repare que mantivemos o comment\_ID de cada documento de comentário em vez de manter todo o documento. Esse tipo de abordagem é conhecido como referências-filhas. A principal razão para essa nomeação é que o documento é considerado pai e os documentos de comentários são entendidos como filhos, de modo que há um relacionamento de pai-filho entre eles (KHAN, 2022).

Esse modelo de referência é sempre seguido quando utilizamos um banco de dados relacional. Os dados normalizados são muito úteis quando um aplicativo é requerido para acessar ou

consultar o documento separado do banco de dados. No entanto, num banco de dados NoSQL como no caso do MongoDB, há outra abordagem, chamada de relacionamento incorporado, a qual estudaremos de maneira detalhada a seguir.

## Relações incorporadas

O relacionamento incorporado também é conhecido como a forma desnormalizada de dados, em que os documentos são simplesmente desnormalizados pela incorporação dos dados no documento principal. Veja o exemplo da Figura 2, a seguir.

Figura 2 | Exemplo de relações incorporadas



Fonte: MongoDB (2016).

Considerando a imagem anterior, é possível notar que não precisamos de coletas separadas ou ID. Nos relacionamentos incorporados da modelagem de dados, todos os dados relacionados são incorporados ao documento principal, o que resulta em aplicativos que demandam menos consultas ao banco de dados, pois todas as informações podem ser obtidas em uma consulta. Por outro lado, é impossível examinar documentos incorporados por conta própria. Desse modo, se precisarmos acessar os documentos incorporados, escolher esse modelo de dados não será uma decisão sábia.

## Modelando um para um

# Bancos de Dados Não Relacionais

A ampli



A modelagem de dados é o processo de pegar dados não estruturados que são gerados a partir do cenário do mundo real e, em seguida, estruturá-los num modelo de dados lógico dentro de um banco de dados. Tal procedimento é realizado sob um conjunto de critérios. Por exemplo, digamos que você deseje projetar um modelo de dados de uma loja on-line. No início haverá muitos dados não estruturados que você precisará utilizar. Entretanto, para usá-los corretamente, você deverá organizá-los de forma estruturada, por meio de um processo conhecido como **modelagem de dados**, que representa a parte mais exigente e essencial do desenvolvimento, já que toda a aplicação depende disso.

Existem diferentes tipos de relacionamento entre dados baseados em bancos de dados distintos. Na modelagem de dados do MongoDB, temos três tipos diferentes de relacionamentos entre os dados, que são:

- Um para um.
- Um para muitos.
- Muitos para muitos.

Você já entenderá, pelo nome dos modelos, o que eles significam. Vamos explorar o primeiro exemplo nesta seção: a modelagem um para um.

## Modelando um para um com documentos incorporados

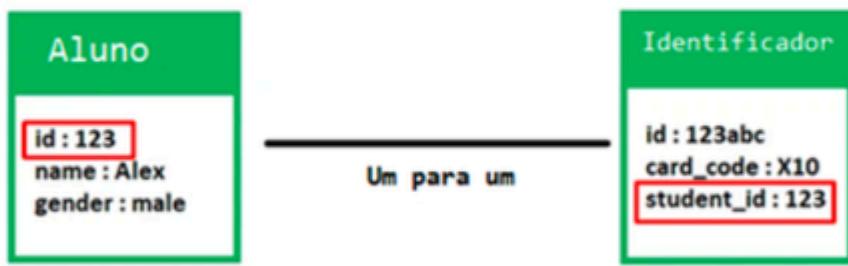
No modelo de dados do MongoDB, quando um campo puder ter apenas **um valor**, será considerado um **relacionamento um para um** entre os dados, que é o modelo de relacionamento

# Bancos de Dados Não Relacionais

A ampli

de dados mais simples. Para exemplificar, digamos que você esteja trabalhando com um sistema de gerenciamento de uma escola e tenha duas entidades: aluno e identificador (KHAN, 2022). Há uma relação de um para um entre essas duas entidades, já que um aluno pode ter apenas um ID exclusivo, ou seja, não pode consumir id múltiplo. Por outro lado, um id múltiplo não pode ser o mesmo para um aluno. Vamos analisar o diagrama da Figura 3 para ter uma compreensão mais clara.

Figura 3 | Exemplo de modelagem um para um



Fonte: adaptada de Khan (2022).

Podemos ver que **um aluno** tem apenas **um identificador** e **um identificador** é aplicável a apenas **um aluno**. A incorporação de dados conectados em um único documento pode reduzir o número de operações de leitura exigidas para obter dados. Em geral, você deve estruturar seu esquema para que o aplicativo receba todas as informações necessárias numa única operação de leitura.

### Trade-offs do padrão de subconjunto

Cabe observar que o uso de documentos menores contendo dados acessados com mais frequência reduz o tamanho geral do conjunto de trabalho. Esses documentos menores resultam em melhor desempenho de leitura e disponibilizam mais memória para o aplicativo.

No entanto, é importante entender seu aplicativo e a maneira como ele carrega os dados. Se você dividir seus dados em várias coleções incorretamente, seu aplicativo geralmente precisará fazer várias viagens ao banco de dados e contar com as operações JOIN para recuperar todos os dados necessários.

Além disso, dividir seus dados em várias coleções pequenas pode aumentar a manutenção necessária do banco de dados, pois existe a possibilidade de que seja difícil rastrear quais dados estão armazenados em uma determinada coleção.

## Modelando muitos para muitos



É importante lembrar que na modelagem de dados do MongoDB temos três tipos diferentes de relacionamento entre os dados, que são:

- Um para um.
- Um para muitos.
- Muitos para muitos.

Vamos estudar, agora, os dois últimos tipos de relacionamentos: um para muitos e muitos para muitos.

### Modelando um para muitos

Um relacionamento **um para muitos** é o relacionamento de modelagem de dados mais importante e mais usado no MongoDB, sendo subdividido em três categorias:

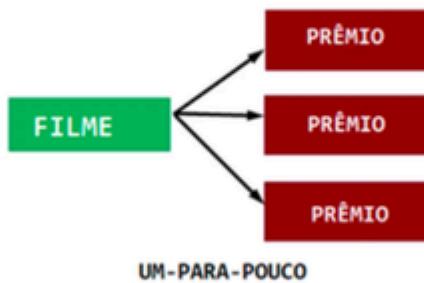
- Um para poucos/alguns.
- Um para muitos.
- Um para milhares.

As principais diferenças estão baseadas na quantidade. Vejamos, a seguir, um exemplo de relacionamento **um para poucos**. Para demonstrá-lo, vamos levar em consideração um filme que

# Bancos de Dados Não Relacionais

pode ter ganhado alguns prêmios, mas não muitos, como mostra a Figura 4.

Figura 4 | Esquema exemplificando o tipo de relacionamento um para poucos

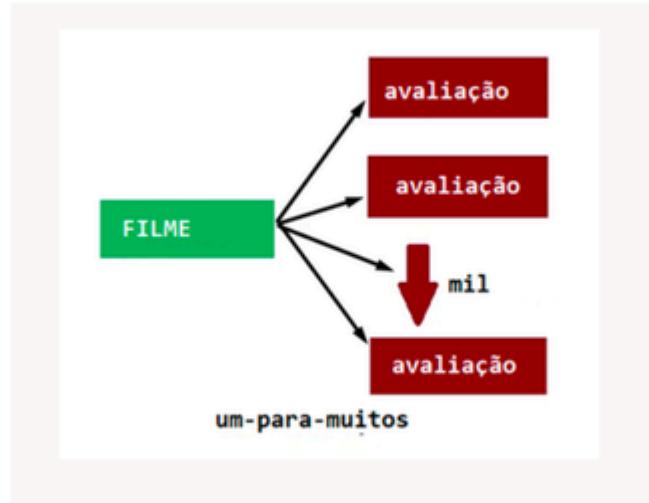


Fonte: adaptada de Khan (2022).

O segundo tipo é o relacionamento um para muitos, o qual se refere a um documento que pode ter relação com centenas ou mesmo milhares de outros documentos. Por exemplo, o mesmo filme do caso anterior poderá ter inúmeras avaliações, como é possível visualizar na Figura 5.

# Bancos de Dados Não Relacionais

Figura 5 | Esquema exemplificando o tipo de relacionamento um para muitos



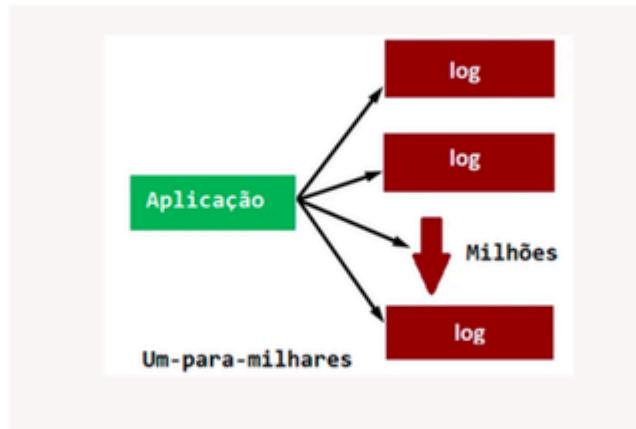
Fonte: adaptada de Khan (2022)

Entender o relacionamento do tipo um para milhares talvez seja um pouco mais complexo, mas, para uma assimilação mais efetiva, você pode compará-lo com os logs de eventos que coletam mensagens de log de máquinas diferentes. Se tentarmos ver os logs de acesso do nosso programa de aplicação do filme, ele poderá retornar milhões de logs, como no esquema da Figura 6.

# Bancos de Dados Não Relacionais

A ampli

Figura 6 | Esquema exemplificando o tipo de relacionamento um para milhares



Fonte: adaptada de Khan (2022).

## Modelando muitos para muitos

Como não existe um comando único para implementar um relacionamento muitos para muitos num banco de dados relacional, esse modelo de relacionamento se torna mais complexo que o relacionamento um para muitos. O mesmo vale quanto ao uso do MongoDB para implementá-lo. A capacidade de armazenar arrays em um documento, por outro lado, facilita a recuperação e a manutenção dos dados arquivados, além de fornecer as informações necessárias para vincular dois documentos num código.

Os relacionamentos **muitos para muitos** representam um tipo de relacionamento MongoDB no qual duas entidades num documento podem ter vários relacionamentos. É uma espécie de relação vice-versa entre os documentos. Nesse tipo de relacionamento, pode haver relacionamentos nos dois sentidos. Por exemplo, um filme pode ter muitos atores, e muitos atores podem fazer um filme. Além disso, muitos filmes podem ter muitos atores, assim como muitos atores podem fazer muitos filmes.

Esses são os diferentes tipos de relacionamentos na modelagem de dados. Em um banco de dados NoSQL, como o MongoDB, os relacionamentos um para muitos também são importantes porque a referência e a incorporação dependem deles.

Para concluir esta seção de estudos, vale destacar a importância de lembrar que existem dois tipos de relacionamentos no MongoDB: o incorporado e o feito como referência. Cada relacionamento tem seu próprio conjunto de vantagens e desvantagens, e as conexões ajudam no aprimoramento do desempenho.

O MongoDB implementa qualquer um dos relacionamentos – *um para um*, *um para muitos* ou *muitos para muitos* –, dependendo da situação. A instauração desses relacionamentos melhora

muito a consistência dos dados.

## Videoaula: Modelagem e manipulação no MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

A modelagem de dados do aplicativo para o MongoDB deve considerar vários fatores operacionais que afetam o desempenho desse software. Por exemplo, diferentes modelos de dados podem permitir consultas mais eficientes, aumentar a taxa de transferência de operações de inserção e atualização ou distribuir atividades para um cluster fragmentado com mais eficiência. Veremos, neste vídeo, mais detalhes a respeito de um assunto tão importante: fatores operacionais e modelos de dados!

## Saiba mais



# Bancos de Dados Não Relacionais



Caso você queira garantir mais informações sobre bancos de dados não relacionais, consulte o conteúdo indicado a seguir. Vale a pena conferir!

["O MongoDB 5.0 vem com dados na forma de séries temporais, mudanças na numeração e muito mais"](#)

## Referências



CREATE Documents. **MongoDB**, [s. d.]b. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/create-document-playground/>. Acesso em: 2 maio 2022.

KHAN, M. N. R. How to model your data in MongoDB. **Code Source**, 28 jan. 2022. Disponível em: <https://codesource.io/how-to-model-your-data-in-mongodb/>. Acesso em: 10 jun. 2022.

MODEL relationships between documents. **MongoDB**, 5 maio 2016. Disponível em: <https://www.mongodb.com/docs/manual/applications/data-models-relationships/>. Acesso em: 2 maio 2022.

WHAT is ObjectId in MongoDB. **GeeksforGeeks**, 1 nov. 2020. Disponível em:

<https://www.geeksforgeeks.org/what-is-objectid-in-mongodb/>. Acesso em: 2 maio 2022.

## Aula 4

Creating a collection no MongoDB

### Introdução



Nesta etapa de aprendizagem, vamos dar início ao estudo do processo de criar coleções no MongoDB. Como cada objeto no MongoDB é chamado de documento e todos os objetos juntos criam uma coleção, nos primeiros instantes desta aula você poderá se familiarizar com o recurso Creating a Collection no MongoDB, que basicamente é uma ferramenta para criar coleções. Em seguida, estudaremos a validação no Mongo DB e o MongoDB Compass, uma GUI que permite gerenciar dados do MongoDB por meio de uma exibição visual conveniente. Para finalizar, depois de definir as regras de validação, vamos apresentar uma das ferramentas disponíveis para testá-las. Preparado para começar seus estudos?

### Introdução ao Creating a Collection no MongoDB



O MongoDB armazena dados na forma de documentos, e todos os documentos semelhantes são armazenados numa coleção. As coleções funcionam como tabelas no banco de dados relacional, sendo capazes de armazenar documentos de diferentes tipos. A criação e a remoção de coleções no MongoDB podem ser feitas de maneiras específicas (W3SCHOOLS, 2019).

Logo, cada objeto no MongoDB é chamado de documento, e todos os objetos juntos criam uma coleção. Basicamente, a criação da coleção pode ser feita usando o db.createCollection(name, options). No entanto, normalmente você não precisará construir uma coleção própria. O MongoDB desempenha essa tarefa de criação automaticamente quando você começa a inserir alguns documentos para formar um banco de dados (MONGODB, [s. d.]). O Código 1, a seguir, indica a sintaxe que demonstrará como criar sua coleção no MongoDB.

Código 1 | Sintaxe para criar coleção no MongoDB

```
1 db.createCollection(collection_name, options)
```

Fonte: MongoDB ([s. d.]).

O método db.createCollection() abrange os parâmetros apresentados no Quadro 1, a seguir.

Quadro 1 | Parâmetros do método db.createCollection()

Parâmetro	Tipo	Descrição
name	string	O nome da coleção a ser criada.
options	documento	(Opcional) Especifique opções sobre tamanho de memória e indexação.

Fonte: MongoDB ([s. d.]).

# Bancos de Dados Não Relacionais



O parâmetro Options é opcional, portanto, você precisa especificar apenas o nome da coleção. A seguir, no Quadro 2, há uma lista de opções que você poderá utilizar.

Campo	Modelo	Descrição
capped	booleano	(Opcional) Se true, habilita uma coleção limitada. A coleção limitada é uma coleção de tamanho fixo que substitui automaticamente suas entradas mais antigas quando atinge seu tamanho máximo. <b>Se você especificar true, também precisará especificar o parâmetro size.</b>
autoIndexId	booleano	(Opcional) Se verdadeiro, cria automaticamente o índice no campo id.s. O valor-padrão é falso.
size	número	(Opcional) Especifica um tamanho máximo em bytes para uma coleção limitada. <b>Se capped for true, você também precisará especificar esse campo.</b>
Max	número	(Opcional) Especifica o número máximo de documentos permitidos na coleção limitada.

Fonte: MongoDB ([s. d.]).

Ao inserir o documento, primeiro o MongoDB verificará o campo de tamanho da coleção limitada e, em seguida, analisará o campo máximo. Vejamos, a seguir, alguns exemplos aplicados a coleções no MongoDB.

**Exemplo 1:** criar uma coleção com validação de documentos.

As coleções com validação compararam cada documento inserido ou atualizado com os critérios especificados na opção validator. Dependendo de validationLevel e validationAction, o MongoDB retorna um aviso ou se recusa a inserir ou atualizar o documento, caso este não atenda aos critérios especificados. O exemplo a seguir (Código 2) cria uma coleção contacts com um validador de esquema JSON.

Código 2 | Criar coleção contacts no MongoDB

```
1  db.createCollection( "contacts", {  
2      validator: { $jsonSchema: {  
3          bsonType: "object",  
4          required: [ "phone" ],  
5          properties: {  
6              phone: {  
7                  bsonType: "string",  
8                  description: "must be a string and is required"  
9              },  
10             email: {  
11                 bsonType : "string",  
12                 pattern : "@mongodb\\.com$",  
13                 description: "must be a string and match the regular  
14                 expression pattern"  
15             }  
16         } }  
17     }  
18  
19  
20  
21  
22
```

Fonte: MongoDB ([s. d.]).

Com o validador no lugar, a operação de inserção, apresentada pelo Código 3, falha na validação.

Código 3 | Operação de inserção de falhas na validação da criação de coleção

```
1 db.contacts.insertOne( { name: "Amanda", status: "Updated" } )
```

Fonte: MongoDB ([s. d.]).

O método retorna o erro, como demonstrado no Código 4, a seguir.

Código 4 | Método retornando erro

```
1  Uncaught:  
2  MongoServerError: Document failed validation  
3  Additional information: {  
4    failingDocumentId: ObjectId("61a8f4847a818411619e952e"),  
5    details: {  
6      operatorName: '$jsonSchema',  
7      schemaRulesNotSatisfied: [  
8        {  
9          operatorName: 'properties',  
10         propertiesNotSatisfied: [  
11           {  
12             propertyName: 'status',  
13             description: 'can only be one of the enum values',  
14             details: [ [Object] ]  
15           }  
16         ]  
17       ]  
18     }  
19   }
```

```
14  },
15  {
16    operatorName: 'required',
17    specifiedAs: { required: [ 'phone' ] },
18    missingProperties: [ 'phone' ]
19  }
20
21
22
23
24
25
```

Fonte: MongoDB ([s. d.]).

Para visualizar as especificações de validação de uma coleção, basta utilizar db.getCollectionInfos().

**Exemplo 2:** criar uma coleção automaticamente com o MongoDB.

O MongoDB pode criar coleções automaticamente à medida que os documentos são inseridos. Tomemos como exemplo uma situação em que você deseja inserir um documento usando insert() numa coleção chamada "movie". Você pode digitar o comando apresentado no Código 5, a seguir.

Código 5 | Inserir um documento usando insert() numa coleção

```
1 db.movie.insert({"name": "Avengers: Endgame"})
```

Fonte: W3Schools (2019).

A operação anterior criará automaticamente uma coleção, caso não exista uma coleção com o mesmo nome no momento. Além disso, se você deseja verificar um documento inserido, pode usar o método find() (W3SCHOOLS, 2019). O Código 6 demonstra a sintaxe desse procedimento.

Código 6 | Sintaxe para inserir um documento usando insert() numa coleção

```
1 db.collection_name.insert()
```

Fonte: W3Schools (2019).

Obteremos como resultado a saída indicada no Código 7, a seguir.

Código 7 | Resultado da inserção de um documento usando insert() numa coleção

```
1 db.movie.insert({"name": "Avengers: Endgame"})
2 WriteResult({"nInserted": 1 })
3
4 db.movie.find()
5 {"_id": ObjectId("5d0a8dgd45993456ef9f2910"), "name": "Avengers:
6 Endgame"}
```

Fonte: W3Schools (2019).

**Exemplo 3:** como visualizar todas as coleções existentes no banco de dados?

No MongoDB, podemos visualizar toda a coleção presente no banco de dados usando o comando exibido no Código 8, a seguir.

Código 8 | Visualizar todas as coleções existentes no banco de dados

```
1 | show collections
```

Fonte: Chhipa (2021).

Tal comando retorna uma lista com todas as coleções existentes no banco de dados, obtendo como resultado a saída apresentada no Código 9.

Código 9 | Saída com a lista das coleções existentes no banco de dados

1	show collections
2	myNewCollection1
3	myNewCollection2

Fonte: Chhipa (2021).

Neste bloco de estudos, você pôde entender como é possível criar uma coleção no MongoDB usando diferentes métodos, junto com exemplos. Aprender a utilizar o MongoDB é uma habilidade recomendada por muitos desenvolvedores da web, pois aumenta a gama de oportunidades de trabalho.

## Validação



Um aspecto muito importante possível de ser observado nos bancos de dados relacionais é o fato de que eles operam em esquemas fixos e rígidos, com campos de tipos de dados conhecidos. Bancos de dados orientados a documentos, como o MongoDB, são mais flexíveis a esse respeito, pois permitem remodelar a estrutura de seus documentos conforme a necessidade.

Muitos bancos de dados de documentos oferecem a possibilidade de que você defina regras responsáveis por ditar como as partes dos dados de seus documentos devem ser estruturadas, assim como concedem certo grau de liberdade para alterar essa organização, quando necessário.

O MongoDB tem um recurso chamado “validação de esquema” que permite aplicar restrições na estrutura de seus documentos. A validação do esquema é construída em torno do JSON Schema, que é um padrão aberto para descrição e validação da estrutura do documento JSON (MONGODB, [s. d.]).

## Pré-requisitos e premissas

Presume-se que você tenha uma instância do MongoDB em execução disponível e acessível. Se você não tiver acesso a essa funcionalidade, também poderá conquistá-la facilmente usando o Docker para executar um contêiner do MongoDB. Por conveniência, usaremos o MongoDB Compass, que é a GUI (ferramenta interativa para consultar, otimizar e analisar os dados) oficial do MongoDB.

## Executar um contêiner do Docker do MongoDB

# Bancos de Dados Não Relacionais

Você pode criar e executar um contêiner do Docker chamado “mongodb” executando o comando indicado no Código 10, a seguir.

Código 10 | Comando para criar e executar um contêiner do Docker

```
1 docker run --name mongodb -p 27017:27017 -d mongo
```

Fonte: Zafeiropoulos (2022).

Depois de criar o contêiner, você pode iniciá-lo usando, respectivamente, os comandos apresentados no Código 11.

Código 11 | Inicializando o contêiner do Docker

```
1 docker stop mongodb  
2 docker start mongodb
```

Fonte: Zafeiropoulos (2022).

## Obtenha o MongoDB Compass GUI e defina um banco de dados e uma coleção

Baixe o GUI Compass no Portal do MongoDB. Depois de instalá-lo, execute-o. Certifique-se de que o contêiner mongodb esteja funcionando e crie uma nova conexão usando uma string de conexão, como a descrita no Código 12.

Código 12 | String de conexão

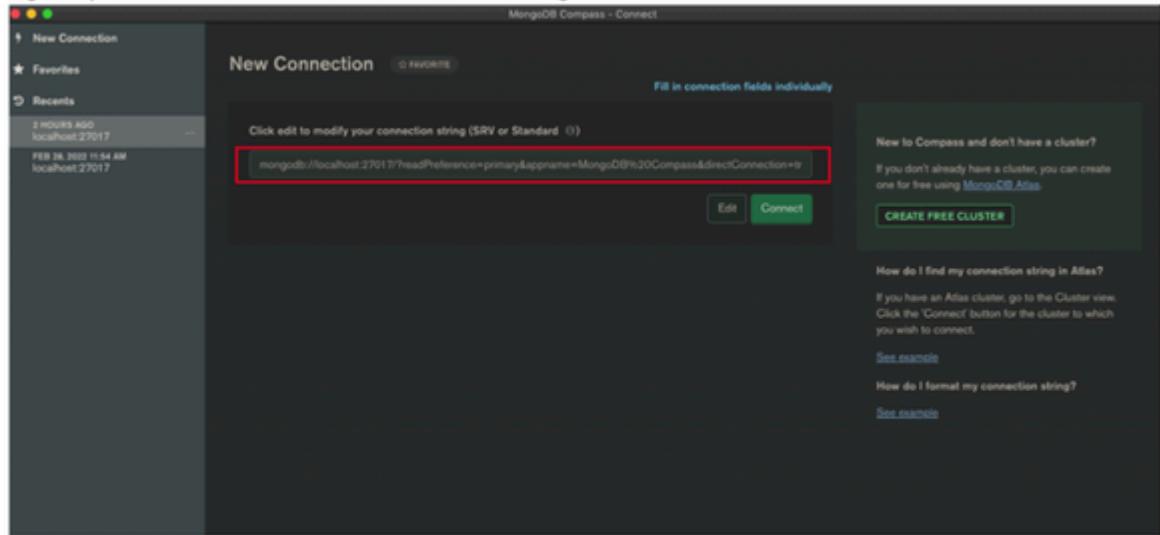
```
1  mongodb://localhost:27017/?  
2  readPreference=primary&appname=MongoDB%20Compass&directConnection=true&  
3  ssl=false
```

Fonte: Zafeiropoulos (2022).

Após conectar-se com sucesso a uma instância do Docker do MongoDB, como ilustra a Figura 1, a seguir, você pode criar um novo banco de dados e uma nova coleção. Nomeie-os como “gerenciamento de tickets” e “usuários”, respectivamente, como indica a Figura 2.

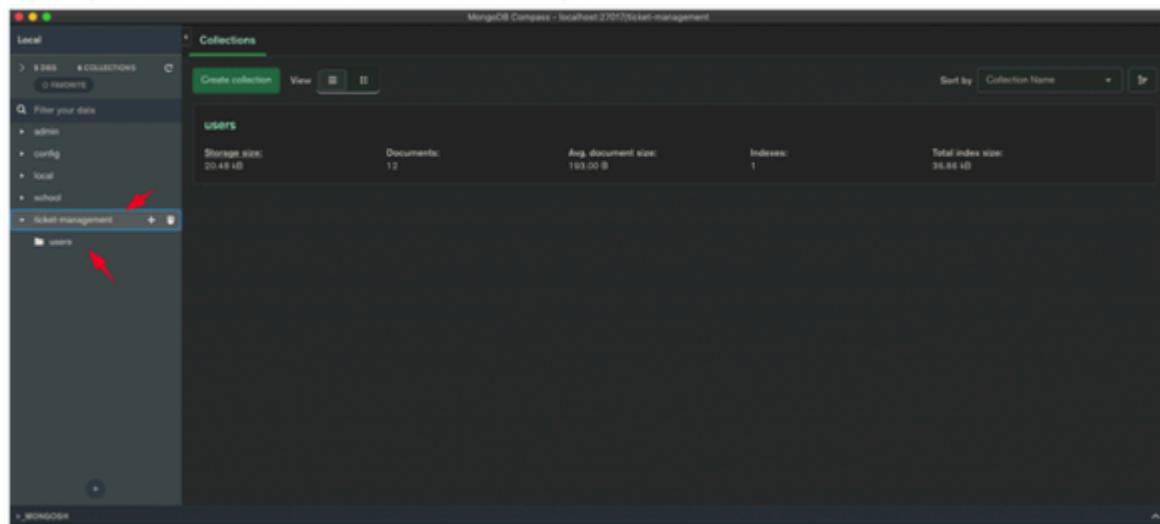
# Bancos de Dados Não Relacionais

Figura 1 | Conectando-se a uma instância do Docker do MongoDB



Fonte: Zafeiropoulos (2022).

Figura 2 | Criando um novo banco de dados e uma nova coleção



Fonte: Zafeiropoulos (2022).

A coleção de “usuários” armazenará documentos de usuários, e os documentos devem ser validados por suas regras de validação.

## Definir as propriedades de um documento MongoDB

Um documento MongoDB é um conjunto ordenado de pares chave-valor. Uma diferença fundamental é que um documento do MongoDB pode armazenar documentos de qualquer tamanho de pares de valores-chave, bem como documentos aninhados.

Contudo, no exemplo que estamos apresentando, queremos forçar a coleção de “usuários” a conter documentos que apresentem estritamente as mesmas propriedades (chaves). Isso é análogo aos campos (colunas) de uma tabela num banco de dados relacional. Portanto, desejamos que cada documento tenha exatamente as mesmas propriedades/campos (ZAFIROPOULOS, 2022).

## O mongo \_id

Antes de identificarmos os campos de nossa coleção de “usuários”, devemos relembrar que o MongoDB gera automaticamente uma propriedade/campo `_id` especial cada vez que um novo documento é inserido numa coleção. Sabendo disso, observe, no Código 13, a lista dos campos para a coleção “users”.

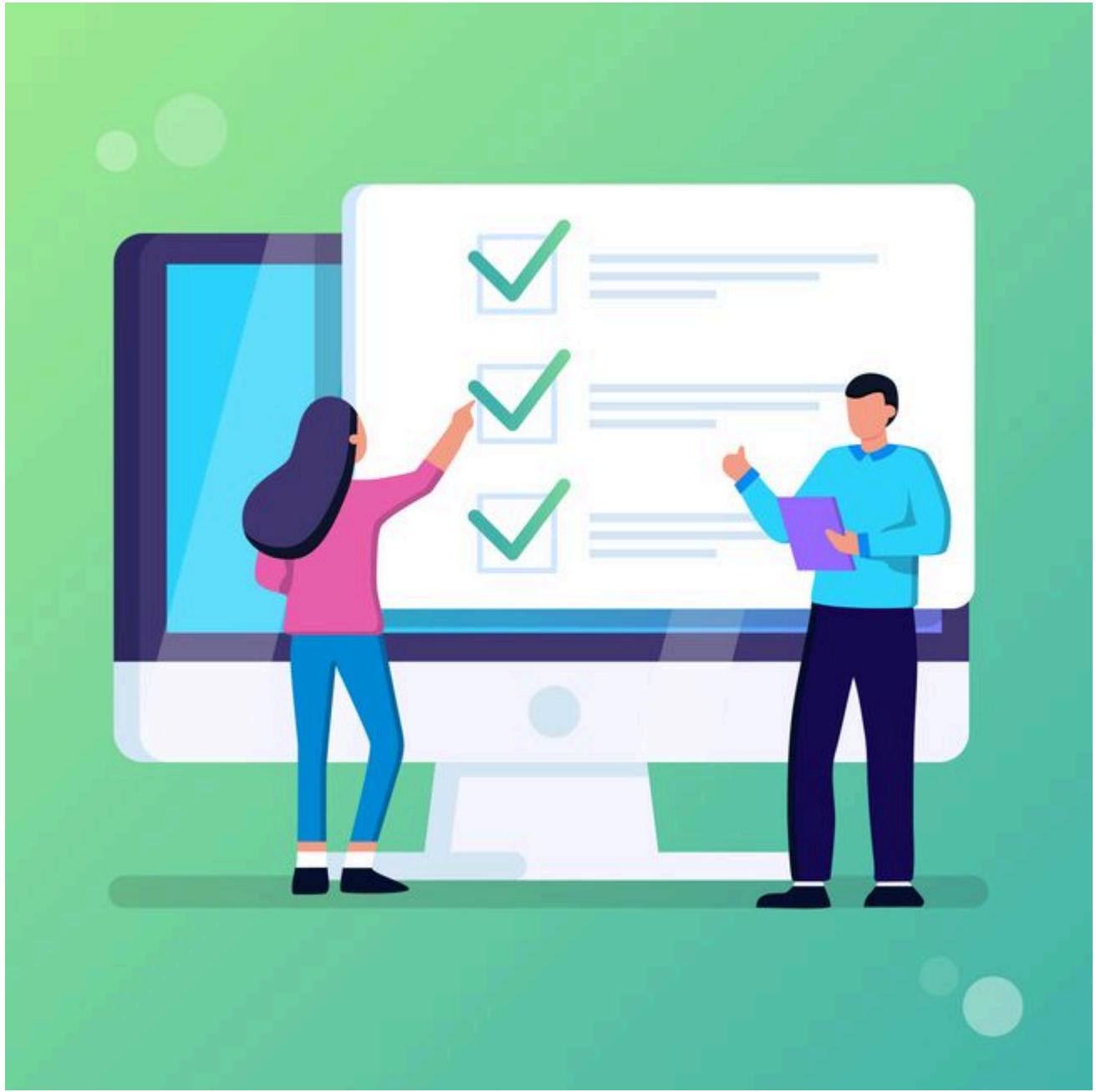
Código 13 | Lista dos campos para a coleção “users”

1	<code>_id</code>
2	<code>nome de usuário,</code>
3	<code>senha,</code>
4	<code>e-mail, data de</code>
5	<code>registro,</code>
6	<code>confirmado,</code>
7	<code>cancelado,</code>
8	<code>typeid,</code>
9	<code>countryid</code>

Fonte: Zafeiropoulos (2022).

O objetivo, nesse caso, é garantir, o máximo que pudermos, que todos os documentos destinados à inserção na coleção de “usuários” sejam compostos por tais campos. No próximo bloco de estudos, daremos continuidade a esse mesmo exemplo, evoluindo para os testes de validação.

## Testes de validação



Antes de ingressarmos, de fato, no estudo dos testes de validação, precisamos aplicar as regras de validação do esquema MongoDB. Então, vamos lá!

### Regras de validação do esquema MongoDB

Para que todos os documentos estejam em conformidade com os campos que determinamos no Bloco 2, usaremos um esquema específico do MongoDB. O conjunto de regras que vamos utilizar será definido a partir da aplicação de um arquivo JSON de acordo com os padrões BSON (ZAFIROPOULOS, 2022). Você pode ler mais sobre o esquema MongoDB e aprender como ele

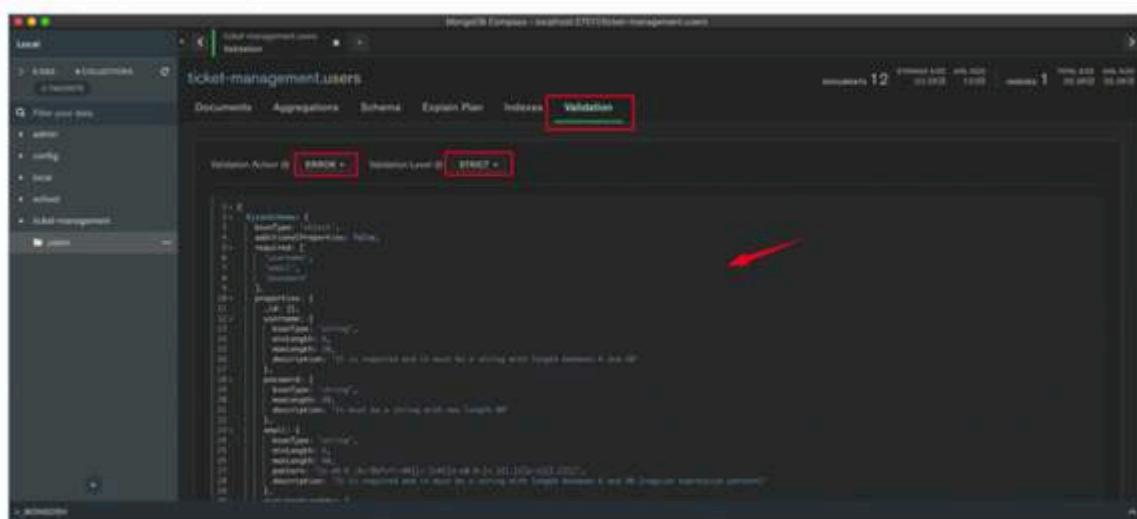
funciona por meio da documentação oficial. Para tanto, consulte os links que estarão listados na seção saiba *mais* desta aula.

Vamos finalmente determinar nosso esquema de validação do MongoDB. Listamos, a seguir, um resumo do que precisaremos definir.

- username, email, e password são obrigatórios e devem estar **presentes em cada documento**.
- username, email, e password devem ser do tipo **string**, e o comprimento de suas strings precisa estar entre os limites mínimo e máximo.
- email deve obedecer a um padrão regex específico (método formal de se especificar um padrão de texto).
- registrationdate deve ser do tipo data.
- confirmed e canceled devem ser do tipo boolean = true ou false.
- typeid e countryid devem ser do tipo int (inteiro) e seus valores precisam estar entre um número mínimo e máximo.

Pronto, as regras estão definidas! Como já criamos nossa coleção de “usuários” no Compass, vamos usar a respectiva GUI. Então, selecionaremos a coleção “users”, clicando na aba “Validation”, e implantaremos nosso esquema JSON (deixe as opções Validation Action e Validation Level para ERROR e STRICT, respectivamente), conforme indicado na Figura 3, a seguir. Mais à frente, no Código 14, você poderá visualizar o exemplo de regras de validação que utilizaremos.

Figura 3 | Regras de validação na GUI



Fonte: Zafeiropoulos (2022).

Código 14 | Esquema de validação de coleção MongoDB

```
1  {
2    $jsonSchema: {
3      bsonType: 'object',
4      additionalProperties: false,
5      required: [
6        'username',
7        'email',
8        'password'
9      ],
10     properties: {
11       _id: {},
12       username: {
13         bsonType: 'string',
14         minLength: 6,
15         maxLength: 20,
16         description: 'It is required and it must be a string with
17 length between 6 and 20'
18       },
19       password: {
20         bsonType: 'string',
21         maxLength: 80,
22         description: 'It must be a string with max length 80'
23     },
24     email: {
25       bsonType: 'string',
26       minLength: 6,
27       maxLength: 40,
28       pattern: '[a-zA-Z._%+!$&*^|~#%{}/-]+@[a-zA-Z-]{1,}([a-zA-Z]{2,22})',
29       description: 'It is required and it must be a string with
30 length between 6 and 40 (regular expression pattern)'
31     },
32     registrationdate: {
33       bsonType: 'date',
34       description: 'It must be a date'
35     },
36     confirmed: {
37       bsonType: 'bool',
38       description: 'It can only be true or false'
39     },
40     canceled: {
41       bsonType: 'bool',
42       description: 'It can only be true or false'
43     },
44     typeid: {
45       bsonType: 'int',
46       minimum: 1,
47       maximum: 4,
48       description: 'It must be an integer in [ 1, 5 ]'
49   }
```

```
50     },
51     countryid: {
52       bsonType: 'int',
53       minimum: 1,
54       maximum: 250,
55       description: 'It must be an integer in [ 1, 250 ]'
56     }
57   }
58 }
59 }
60 }
```

Fonte: Zafeiropoulos (2022).

## Verifique as regras de validação via CLIs mongosh e mongo shell

Depois de determinar as regras de validação, salvaremos o processo no Compass. Para isso, podemos usar o mongosh, já que o Compass fornece uma versão incorporada da CLI do mongosh. No entanto, primeiro será necessário ir ao shell do contêiner, como indica o Código 15, para efetuar tal modificação.

Código 15 | Shell do contêiner

```
1 docker ps
2 CONTAINER ID IMAGE COMMAND STATUS CRIADO PORTAS NOMES
3 11b9a599c13e mongodb "docker-entrypoint.s..." 3 meses atrás Até 4 horas
4 0.0.0.0:27017->27017/tcp mongodb
5 ...
6 docker exec -it mongodb bash
7 root@11b9a599c13e:/#
```

Fonte: Zafeiropoulos (2022).

Execute o mongosh de dentro dele, efetuando o mesmo procedimento apresentado no Código 16, a seguir.

Código 16 | Executando o mongosh dentro do shell do contêiner

```
1 root@11b9a599c13e:#
2 root@11b9a599c13e:~# mongosh
3 ID de log atual do Mongosh: 6229dc064130345cc3d542bf
4 Conectando-se                                     a:
5 mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
6 Usando o MongoDB: 5.0.5
7 Usando o Mongosh: 5.0.5
8 Para obter informações sobre o mongosh, consulte:
9 https://docs.mongodb.com/mongodb-shell/
10 Para ajudar a melhorar nossos produtos, dados de uso anônimos são
11 coletados e enviados ao MongoDB periodicamente ((
12 https://www.mongodb.com/legal/privacy-política).
13 Você pode desativar executando o comando disableTelemetry().
14 -----
15 -----
16 O servidor gerou estes avisos de inicialização ao inicializar:
17 2022-03-10T05:41:44.202+00:00: O uso do sistema de arquivos XFS é
18 altamente recomendado com o mecanismo de armazenamento WiredTiger.
19 Consulte http://dochub.mongodb.org/core/prodnotes-filesystem
20 2022-03-10T05:41:45.856+00:00: O controle de acesso não está
21 habilitado para o banco de dados. O acesso de leitura e gravação aos
22 dados é irrestrito
23 -----
24 Aviso: ~/mongorc.js encontrado, mas não ~/mongoshrc.js. ~/mongorc.js
25 não será carregado.
26 Você pode copiar ou renomear ~/mongorc.js para ~/mongoshrc.js.
27 teste>
```

Fonte: Zafeiropoulos (2022).

Agora, sim, mudaremos para o banco de dados ticket-management e executaremos o comando db.collectionInfos(), a fim de que seja possível, então, obter as informações das regras de validação para a coleção de “usuários”, como mostra o Código 17.

Código 17 | Validação usando a função db.getCollectionInfos()

```
1 test> use ticket-management
2 switched to db ticket-management
3 ticket-management>
4 ticket-management> db.getCollectionInfos({name: "users"})
5 [
6   {
7     name: 'users',
8     type: 'collection',
9     options: {
10       validator: [
11         '$jsonSchema': {
12           bsonType: 'object',
13           additionalProperties: false,
14           required: [ 'username', 'email', 'password' ],
15           properties: {
16             _id: {},
17             username: {
18               bsonType: 'string',
19               minLength: 6,
20               maxLength: 20,
21               description: 'It is required and it must be a string with
22 length between 6 and 20'
23             },
24             password: {
25               bsonType: 'string',
26               maxLength: 80,
27               description: 'It must be a string with max length 80'
28             },
29             email: {
30               bsonType: 'string',
31               minLength: 6,
32               maxLength: 40,
```

```

33     pattern: '[a-zA-Z._%+!$&^~#%()/-]+@[a-zA-Z-
34 ]+.){1,}([a-zA-Z]{2,22})',
35     description: 'It is required and it must be a string with
36 length between 6 and 40 (regular expression pattern)'
37   },
38   registrationdate: { bsonType: 'date', description: 'It must
39 be a date' },
40   confirmed: {
41     bsonType: 'bool',
42     description: 'It can only be true or false'
43   },
44   canceled: {
45     bsonType: 'bool',
46     description: 'It can only be true or false'
47   },
48   typeid: {
49     bsonType: 'int',
50     minimum: 1,
51     maximum: 4,
52     description: 'It must be an integer in [ 1, 5 ]'
53   },
54   countryid: {
55     bsonType: 'int',
56     minimum: 1,
57     maximum: 250,
58     description: 'It must be an integer in [ 1, 250 ]'
59 }
60   }
61 },
62 ],
63 validationLevel: 'strict',
64 validationAction: 'error'
65 },
66 info: {
67   readOnly: false,
68   uuid: UUID("b033d158-5ee1-4187-9e29-70be69ed97bb")
69 },
70 idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
71 }
72 ]
73 <ticket-management>

```

Fonte: Zafeiropoulos (2022).

Dessa vez, as regras de validação do exemplo foram claramente apresentadas.  
**Testando as regras de validação**

Depois de definir as regras de validação, será preciso usar a ferramenta mongosh CLI para testar. Nesse sentido, podemos tentar inserir alguns documentos que não atendam aos requisitos das regras de validação, na intenção de confirmar sua falha (ZAFEIROPOULOS, 2022).

## 1. Teste utilizando o mongosh

Vamos começar inserindo um documento vazio, como no Código 18, a seguir.

Código 18 | Teste utilizando o mongosh

```
1 ticket-management> db.users.insertOne({})
2 Uncaught:
3 MongoServerError: Document failed validation
4 Additional information: {
5   failingDocumentId: ObjectId("622af142daae2661d9e8c62d"),
6   details: {
7     operatorName: '$jsonSchema',
8     schemaRulesNotSatisfied: [
9       {
10         operatorName: 'required',
11         specifiedAs: { required: [ 'username', 'email', 'password' ] }
12       },
13       missingProperties: [ 'email', 'password', 'username' ]
14     ]
15   }
16 }
17 }
18 ticket-management>
```

Fonte: Zafeiropoulos (2022).

Em seguida, faremos outra tentativa, porém agora com um documento que contenha um e-mail inválido, como mostra o Código 19.

Código 19 | Teste utilizando o mongosh com e-mail inválido

```
1 ticket-management> db.users.insertOne(  
2 ... {  
3 ..... "username": "Panos1",  
4 ..... "password": "mypassword",  
5 ..... "email": "email.com",  
6 ..... "registrationdate": ISODate("2022-03-11T11:11:55.353Z"),  
7 ..... "typeid": 4,  
8 ..... "confirmed": false,  
9 ..... "canceled": false,  
10 ..... "countryid": 28  
11 ..... }  
12 ... )  
13 Uncaught:  
14 MongoServerError: Document failed validation  
15 Additional information: {  
16   failingDocumentId: ObjectId("622af390daae2661d9e8c62f"),  
17   details: {  
18     operatorName: '$jsonSchema',  
19     schemaRulesNotSatisfied: [  
20       {  
21         operatorName: 'properties',  
22         propertiesNotSatisfied: [ { propertyName: 'email', details: [  
23           [Object] ] } ]  
24       }  
25     ]  
26   }  
27 }  
28 ticket-management>
```

Fonte: Zafeiropoulos (2022).

Você pode continuar tentando inserir documentos com valores inválidos. No Código 20, por exemplo, utilizou-se um valor do campo typeid (valor igual a 0) quando o valor deveria ser 1, no mínimo, como especificado nas regras.

# Bancos de Dados Não Relacionais

A ampli

Código 20 | Teste utilizando o mongosh com o campo typeid inválido

```

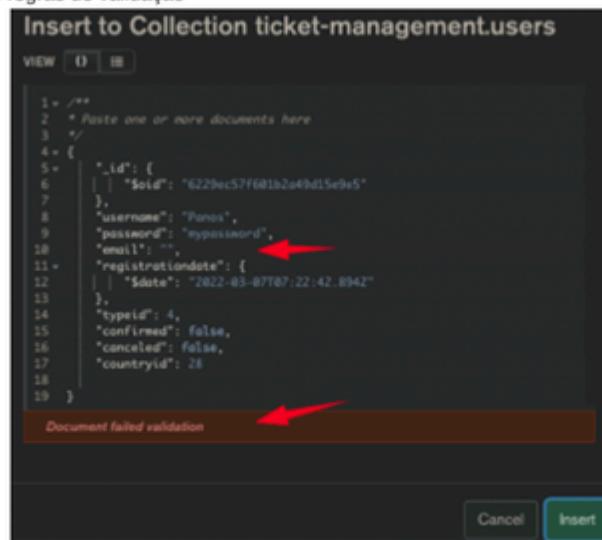
1 ticket-management> db.users.insertOne(
2 ... {
3 ..... "username": "Panos1",
4 ..... "password": "mypassword",
5 ..... "email": "panos1@email.com",
6 ..... "registrationdate": ISODate(),
7 ..... "typeid": 0,
8 ..... "confirmed": false,
9 ..... "canceled": false,
10 ..... "countryid": 28
11 .... }
12 ...
13 Uncaught:
14 MongoServerError: Document failed validation
15 Additional information: {
16   failingDocumentId: ObjectId("622af5c3daae2661d9e8c635"),
17   details: {
18     operatorName: '$jsonSchema',
19     schemaRulesNotSatisfied: [
20       {
21         operatorName: 'properties',
22         propertiesNotSatisfied: [ { propertyName: 'typeid', details: [
23           [Object] ] } ]
24       }
25     ]
26   }
27 }
28 ticket-management>

```

Fonte: Zafeiropoulos (2022).

Ao tentar inserir documentos que não estejam de acordo com as regras de validação estabelecidas, sempre receberemos avisos de “falha”, como ilustra a Figura 4.

Figura 4 | Avisos de falhas de regras de validação



Fonte: Zafeiropoulos (2022).

Utilizar as regras de validação do MongoDB pode ser uma opção útil no que se refere à capacidade de definir esquemas, pois fornece como benefício um certo nível de flexibilidade para a nossa rotina de trabalho.

## Videoaula: Creating a collection no MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Com o MongoDB Compass, ou só Compass, é possível acessar a maioria dos recursos que o mecanismo de banco de dados MongoDB oferece por meio de uma exibição visual intuitiva. Você pode examinar bancos de dados, coleções e documentos individuais, criar consultas interativamente, manipular documentos existentes e projetar pipelines de agregação a partir de uma interface dedicada. Neste vídeo, vamos instalar o MongoDB Compass e nos familiarizar com a execução de algumas administrações de banco de dados usando a ferramenta gráfica.

## Saiba mais



Caso você queira garantir mais informações sobre bancos de dados não relacionais, consulte os conteúdos indicados a seguir. Vale a pena conferir!

- Validação no MongoDB.

["Definir regras de validação para seu esquema"](#)

["Validação de esquema"](#)

["Definição"](#)

["Tipos de BSON"](#)

Essas páginas estão em inglês. Caso tenha dificuldade com essa língua, recomenda-se o uso do recurso de tradução do seu *browser*.

---

## Referências



CHHIPA, K. MongoDB: check the existence of the fields in the specified collection.

**GeeksforGeeks**, 22 nov. 2021. Disponível em: <https://www.geeksforgeeks.org/mongodb-check-the-existence-of-the-fields-in-the-specified-collection/>. Acesso em: 14 jun. 2022.

CREATE Documents. **MongoDB**, [s. d.]. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/create-document-playground/>. Acesso em: 2 maio 2022.

DOCKER and MongoDB. **MongoDB**, 21 ago. 2021. Disponível em:

<https://www.mongodb.com/compatibility/docker>. Acesso em: 2 maio 2022.

DOWNLOAD MongoDB Compass. **MongoDB**, 25 out. 2018. Disponível em:

<https://www.mongodb.com/try/download/compass>. Acesso em: 2 maio 2022.

MODEL relationships between documents. **MongoDB**, 5 maio 2016. Disponível em:

<https://www.mongodb.com/docs/manual/applications/data-models-relationships/>. Acesso em: 2 maio 2022.

MONGODB Create Collection. **W3Schools**, 19 jun. 2019. Disponível em:

<https://www.w3schools.in/mongodb/create-collection>. Acesso em: 14 jun. 2022.

ZAFEIROPOULOS, P. MongoDB schema validation rules. **Better Programming**, 10 mar. 2022.

Disponível em: <https://betterprogramming.pub/mongodb-schema-validation-rules-8a1afc6ea67b>. Acesso em: 14 jun. 2022.

## Aula 5

Revisão da unidade

# Bancos de Dados Não Relacionais

Aprendendo a utilizar as técnicas de manipulação de dados em um banco de dados não relacional a partir do MongoDB



O MongoDB é um banco de dados NoSQL que permite armazenar e consultar documentos no estilo JSON com alguns recursos inteligentes. Isso significa que você pode armazenar objetos com dados aninhados numa coleção (as coleções são como tabelas para o MongoDB).

Geralmente o MongoDB é mais rápido que os bancos de dados relacionais ou SQL tradicionais para armazenamentos transacionais, mas não possui o poder de consulta para uso analítico (CHRISTOPHER, 2020).

E por que utilizar o MongoDB? Existem vários motivos, mas, dentre os principais, podemos destacar o bom funcionamento na nuvem (escala, tolerância a falhas, segurança), a agilidade quanto à inicialização para uso (possui muita documentação), sua ampla utilização por parte de inúmeras organizações de diferentes escalas em todo o mundo e, por fim, o alto investimento feito para que o MongoDB mantenha um desenvolvimento contínuo, um aspecto tão importante quanto todos os anteriores.

O Mongo oferece diversas possibilidades de procedimentos. A partir desse software, podemos, por exemplo: **criar documentos** numa coleção usando os operadores CRUD do MongoDB a partir de um MongoDB Playground:

- Usamos o método insertOne() para inserir um documento.

- Usamos o método `insertMany()` para inserir mais de um documento.

As operações CRUD, acrônimo dos termos em inglês *Create* (Criar), *Read* (Ler), *Update* (Atualizar) e *Delete* (Apagar), são processos básicos porém essenciais, os quais o ajudarão a interagir facilmente com o servidor MongoDB (ROUT, 2021).

O uso do **`ObjectId`** para o campo `_id` tem como resultado os seguintes benefícios adicionais: no mongoshell, podemos acessar o tempo de criação do `ObjectId` utilizando o método `ObjectId.getTimestamp()`; além disso, classificar um campo `_id` que armazena valores `ObjectId` é aproximadamente equivalente à classificação por tempo de criação. É importante entender, também, que no MongoDB cada documento armazenado numa coleção requer um campo `_id` exclusivo, que atua como chave primária. Se um documento inserido omitir o campo `_id`, o driver MongoDB automaticamente gerará um `ObjectId` para o campo `_id`.

O MongoDB possui, ainda, o **`Atlas MongoDB`**, que é um banco de dados em nuvem totalmente gerenciável, desenvolvido pelas mesmas pessoas que criaram o MongoDB. O Atlas lida com toda a complexidade de implantação, gerenciamento e recuperação de suas implantações no provedor de serviços de nuvem de sua escolha (AWS, Azure ou GCP) (MONGODB, 2016a).

Um dos aspectos mais importantes de serem assimilados com relação ao banco de dados do MongoDB é o tipo de relacionamento. Nesse sentido, teremos dois métodos possíveis para a sua criação:

- **Relações referenciadas:** também são conhecidas como o formulário normalizado no qual todos os documentos relacionados são mantidos separados.
- **Relações incorporadas:** também são conhecidas como a forma desnormalizada de dados, em que os documentos são simplesmente desnormalizados pela incorporação dos dados no documento principal.

No momento em que você for projetar modelos de dados, sempre considere o uso dos dados pelo aplicativo (ou seja, consultas, atualizações e processamento dos dados), bem como a estrutura inerente aos próprios dados. O principal desafio na modelagem de dados é equilibrar as necessidades do aplicativo, as características de desempenho do mecanismo de banco de dados e os padrões de recuperação de dados (MONGODB, 2016b).

## Videoaula: Revisão da unidade



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

# Bancos de Dados Não Relacionais

MongoDB é um banco de dados de documentos de uso geral projetado para o desenvolvimento de aplicativos modernos e para a nuvem. Sua arquitetura escalável permite que você, em sua atividade profissional, atenda à crescente demanda de sistemas projetados, adicionando mais nós para compartilhar a carga. Vamos revisar, nesta etapa de aprendizagem, alguns dos principais conceitos e termos com os quais você se deparou ao estudar o MongoDB. Vamos lá!

## Estudo de caso



O MongoDB é altamente flexível e permite combinar e armazenar vários tipos de dados. Também armazena e lida com quantidades maiores de dados do que os bancos de dados relacionais tradicionais. Além disso, usa um formato de armazenamento de documentos chamado BSON, que é uma forma binária de JSON (JavaScript Object Notation), capaz de acomodar mais tipos de dados.

Trata-se de um software que basicamente funciona armazenando objetos de dados em coleções e documentos em vez das tabelas e linhas comumente usadas em bancos de dados relacionais tradicionais. As coleções compreendem conjuntos de documentos que são equivalentes a tabelas num banco de dados relacional. Os documentos consistem em pares chave-valor, os quais representam a unidade básica de dados no MongoDB.

Pensando no funcionamento do MongoDB, temos a possibilidade de fazer buscas textuais, o que desencadeia algumas facilidades para os desenvolvedores. Um índice de texto pode ser uma string ou uma matriz de elementos de string. Para realizar uma pesquisa de texto, a coleção deve conter um índice de texto. Uma coleção pode possuir apenas um índice de texto, e um único índice de texto pode ser aplicado a vários campos.

# Bancos de Dados Não Relacionais

Uma pesquisa também pode ser realizada numa coleção com um índice de texto usando o operador \$text, o qual tokeniza cada string de pesquisa com espaço em branco e trata todas as pontuações, exceto “-” e “\”, como delimitadores. Depois que a string de pesquisa é tokenizada, o operador executa a operação lógica OR nos tokens.

A ideia deste estudo de caso é fazer com que você explique como utilizar esse tipo de busca e demonstre algumas de suas configurações para os novos contratados da sua empresa, os quais trabalharão com você, mas ainda não possuem tanto conhecimento sobre buscas textuais no MongoDB.

Diante disso, você deve preparar um treinamento para o processo de busca textual, lembrando-se, também, de destacar a funcionalidade do *stemming* e do *stop words*, uma vez que o recurso de pesquisa de texto é um dos mais importantes do MongoDB.

## Refita

MongoDB é um programa de banco de dados orientado a documentos multiplataforma disponível na fonte. Classificado como um programa de banco de dados NoSQL, o MongoDB usa documentos semelhantes a JSON com esquemas opcionais.

Trata-se de um banco de dados distribuído, baseado em documentos e de uso geral, criado para desenvolvedores de aplicativos modernos e para a era da nuvem. Vale destacar que o MongoDB é escrito em C++.

## Base de dados

O banco de dados é um contêiner físico para coleções que obtém seu próprio conjunto de arquivos no sistema de arquivos. Um único servidor MongoDB normalmente possui vários bancos de dados.

## Coleção

Collection é um grupo de documentos do MongoDB. É o equivalente a uma tabela RDBMS. Uma coleção existe num único banco de dados. As coleções não impõem um esquema. Os documentos de uma coleção podem ter campos diferentes. Normalmente todos os documentos numa coleção são de finalidade semelhante ou relacionada.

## Documento

Um documento é um conjunto de pares chave-valor. Os documentos têm esquema dinâmico, o que significa que documentos de uma mesma coleção não precisam ter o mesmo conjunto de campos ou estrutura, e os campos comuns nos documentos de uma coleção podem conter diferentes tipos de dados.

## Onde usar o MongoDB?

- Big Data.
- Gerenciamento e entrega de conteúdo.
- Infraestrutura móvel e social.
- Gerenciamento de dados do usuário.
- Centro de dados.

Para aprender mais detalhes sobre o MongoDB, você pode conferir os conteúdos fornecidos no site do próprio software.

## Videoaula: Resolução do estudo de caso



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

A busca textual é uma abordagem que tem a finalidade de achar, numa base de textos, um *match* para um determinado resultado procurado. O objetivo é encontrar uma palavra ou um conjunto de termos em meio a uma grande quantidade de documentos. Com o tempo, foram introduzidas outras formas de achar o dado desejado por aproximação (*stemming*) e de ignorar as palavras que teriam correspondência com um número considerável de documentos, as quais são chamadas de palavras-coringa (*stop words*).

Para que a aproximação dos possíveis candidatos a resultados da palavra pesquisada ocorra, as ferramentas que realizam o full-text-search – como o MongoDB – implementam a técnica de *stemming*.

E o que é *stemming*? Essa técnica consiste em analisar palavras, dentro do seu respectivo idioma, para tentar reduzi-las a uma raiz, a qual pode ser utilizada na aquisição de resultados similares. Para que você possa entender de forma mais efetiva, acompanhe o conjunto de palavras descrito no Quadro 1, a seguir.

Quadro 1 | Exemplo de aplicação do stemming

Palavra	Stem
Pedro	Peter
Pedra	Peter
Ensaiar	Ensai
Ensaio	Ensai

Fonte: Fachine (2020).

A fim de evitar resultados não pretendidos, o MongoDB faz uso de uma lista de palavras indesejadas por idioma. Palavras como *por*, *para*, *de*, entre outras, são o que chamamos de **stop words**.

Vamos, então, dar continuidade ao exemplo anterior. Para tanto, criaremos uma base de dados e, em seguida, vamos inserir tais dados numa coleção, como mostra o Código 1, a seguir.

Código 1 | Exemplo de database para inserir dados numa coleção

```
1 > use redspark // criando uma nova database
2 > db.jobs.insert([
3   { message: 'Solicitação de desenvolvimento de aplicativo mobile',
4     author: 'joão' },
5   { message: 'Solicitação de desenvolvimento de front-end', author:
6     'Joao' },
7   { message: 'Solicitação de desenvolvimento de back-end', author:
8     'Pedro' }
9 ])
```

Fonte: Fachine (2020).

Construiremos, agora, o nosso índice textual, indexando somente o autor, por enquanto, como indica o Código 2.

Código 2 | Criando o índice textual

```
1 | > db.jobs.createIndex({ author: 'text' }, { default_language: 'pt' })
```

Fonte: Fachine (2020).

O primeiro parâmetro ao qual temos acesso é um objeto que contém o campo a ser indexado (o autor) e o segundo, um objeto de configuração. Para o exemplo em questão, estamos passando pt porque desejamos que o índice utilize o **stop words** e o **stemming** no idioma português.

### Realizando a primeira busca

Agora faremos uma pesquisa com o nome “João”. A sintaxe básica para efetuar essa busca textual está descrita no Código 3, a seguir.

Código 3 | Sintaxe para realizar a busca textual

```
1 > db.jobs.find({ $text: { $search: 'joao' } })
```

Fonte: Fachine (2020).

Podemos notar, nesse caso, que conseguimos recuperar dois documentos com a busca, como mostra o Código 4.

Código 4 | Busca textual

```
1 { "_id" : ObjectId("5e582d2ad5cadfff4c390ab3"), "message" :  
2 "Solicitação de desenvolvimento de front-end", "author" : "Joao" }  
3 { "_id" : ObjectId("5e582d2ad5cadfff4c390ab2"), "message" :  
4 "Solicitação de desenvolvimento de aplicativo mobile", "author" :  
"joão" }
```

Fonte: Fachine (2020).

Com base nesse resultado, podemos verificar dois aspectos:

- Independentemente do *case sensitivity*, conseguimos o resultado esperado.
- Independentemente dos *diacritics* (que são os acidentes nas palavras, como os acentos), também foi possível recuperá-los.

Contudo, isso se deve porque o MongoDB tem por padrão o *case-insensitive* e *diacritics-insensitive*. Ou seja, não há distinção entre as letras maiúsculas ou minúsculas, ou mesmo entre a presença ou ausência de acentos, como nos casos a seguir:

- A, B e C / a, b e c.
- á, É, ó e Ú / a, E, o e U.

## Colocando o *stemming* em prática

Neste momento, realizaremos uma nova busca pela palavra “pedra”.

Código 5 | Busca textual pela palavra “pedra”

```
1 | > db.jobs.find({ $text: { $search: 'pedra' } })
```

Fonte: Fachine (2020).

Perceberemos que por causa do stemming, o resultado obtido com o autor será Pedro, como indicado no Código 6.

Código 6 | Resultado da busca textual

```
1 {   "_id" : ObjectId("5e582d2ad5cadfff4c390ab4"), "message" :  
2 "Solicitação de desenvolvimento de back-end", "author" : "Pedro" }
```

Fonte: Fachine (2020).

Talvez num primeiro momento o **stemming** seja difícil de entender. Pode parecer um recurso previsível, porém nem sempre será. Tal complexidade é agravada simplesmente pelo fato de que apenas *matches* completos são compreendidos como resultados numa **full-text search**.

A seguir, faremos uma análise mais minuciosa sobre para verificar por que o exemplo abordado anteriormente funcionou. Em seguida, trataremos de outro caso no qual não será possível garantir o mesmo sucesso. É importante saber que todo o processo está relacionado à linguagem de processamento de texto chamada *Snowball*, que é a verdadeira responsável pelo **stemming**.

### Caso 1

No exemplo aplicado para a busca da palavra **pedra**. Quando visualizamos o Código 5, apresentado anteriormente, notamos que na palavra “pedra” o *stem* relacionado é *pedr*. Logo, é totalmente possível obter **pedro** com base nesse *stem*. É por isso que tivemos o resultado esperado.

### Caso 2

Mas se considerássemos outro documento cujo autor tivesse o nome Paullo e, ao realizar a busca, escrevêssemos Paulo, com apenas um L? Então não teríamos o resultado desejado. Claro, esse fato ocorreria porque ao analisarmos o *stem* retornado pela *Snowball*, veríamos que:

- A resultante do *stemming* de Paulo é *Paul*.
- A resultante do *stemming* de Paullo é *Paull*.

É nesse contexto que podemos destacar a questão das buscas com *matches* exatos. Escrever Paul não é o mesmo que escrever Paullo nem Paull. Ainda que saibamos que um esteja contido em ambos os casos, mesmo assim não haverá um *match* perfeito. Dessa forma, nesse exemplo, nenhum resultado será apresentado.

Outro aspecto importante de ser considerado a respeito do *stemming* e do *stop words* é o uso do *none*. Isso pode ser feito caso você queira que seu banco devolva apenas *matches* exatos de palavras. Para tanto, basta passar o idioma do índice como *none*, como mostra o Código 7.

Código 7 | Aplicando *none* para devolver apenas *matches* exatos

```
1 > db.jobs.createIndex({ author: 'text' }, { default_language: 'none' })
```

Fonte: Fachine (2020).

Como aprendemos, o uso do full-text search com o MongoDB envolve diversas peculiaridades. É claro que se trata de uma ótima ferramenta, mas, assim como fazemos durante o uso de qualquer outro recurso, é necessário, antes de tudo, promover uma análise profunda do caso em que será empregada, para que seja aplicada cuidadosamente.

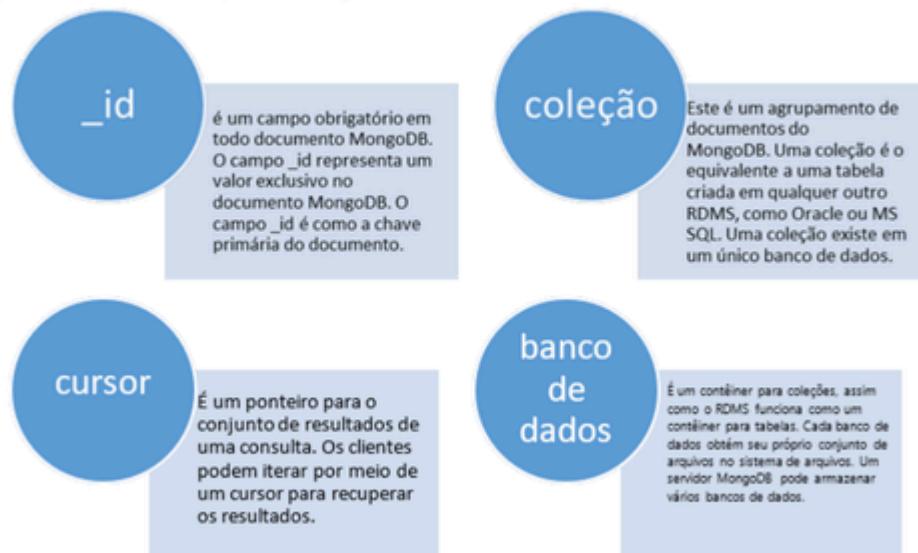
## Resumo visual

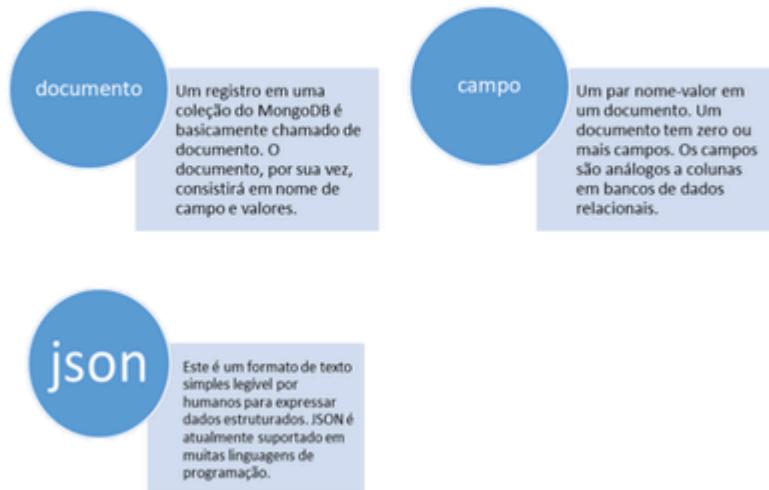
# Bancos de Dados Não Relacionais



Objeto de aprendizagem: infográfico.

Título: Principais componentes da arquitetura MongoDB.





Fonte: elaborado pela autora.

## Referências



CHRISTOPHER, A. A walkthrough of MongoDB. **Analytics Vidhya**, 12 out. 2020. Disponível em: <https://medium.com/analytics-vidhya/a-walkthrough-of-mongodb-aef402594144#:~:text=So%20let's%20have%20another%20go,are%20like%20tables%20for%20MongoDB>). Acesso em: 17 jun. 2022.

CREATE Documents. **MongoDB**, [s. d.]. Disponível em:

<https://www.mongodb.com/docs/mongodb-vscode/create-document-playground/>. Acesso em: 2 maio 2022.

FACHINE, B. MongoDB: entendendo full-text search. **Redspark**, 28 fev. 2020. Disponível em:

<https://www.redspark.io/mongodb-entendendo-full-text-search/>. Acesso em: 17 jun. 2022.

MODEL relationships between documents. **MongoDB**, 5 maio 2016b. Disponível em:

<https://www.mongodb.com/docs/manual/applications/data-models-relationships/>. Acesso em: 2 maio 2022.

MONGODB Atlas: The multi-cloud developer data platform. **MongoDB**, 28 jun. 2016a. Disponível em: <https://www.mongodb.com/atlas>. Acesso em: 17 jun. 2022.

ROUT, A. R. Spring Boot – CRUD Operations using MongoDB. **GeeksforGeeks**, 28 dez. 2021. Disponível em: <https://www.geeksforgeeks.org/spring-boot-crud-operations-using-mongodb/>. Acesso em: 2 maio 2022.

## Unidade 3

Map Reduce e Transactions em Ambiente NoSQL

### Aula 1

Introdução MapReduce

#### Introdução

# Bancos de Dados Não Relacionais



No início desta unidade, de MapReduce e Transactions em ambiente NoSQL, vamos conhecer o que é a tecnologia MapReduce, o contexto no qual ela foi criada e para qual finalidade.

Com isso, entendendo o propósito da tecnologia, mapearemos alguns dos casos em que podemos aplicá-la no nosso cotidiano - seja no nosso emprego ou criando nossas próprias soluções.

Ao final desta aula, com toda essa fundamentação, vamos instalar juntos o MapReduce para podermos fazer uso dessa ferramenta.

## Introdução ao MapReduce



Para falar sobre MapReduce temos que falar, também, sobre o Hadoop.

Hadoop é uma plataforma, construída em Java, que se utiliza de conceitos de computação paralela e distribuída, com o uso de *clusters* (arquitetura de computação em que se utiliza de uma ou várias máquinas agrupadas), para lidar com processamento massivo de dados. Hoje, nossos computadores são cada vez mais potentes, em todos os âmbitos:

1. Processamento: chips menores e mais potentes (Lei de Moore).
2. Memória: cada vez mais barata.
3. Comunicação: dispomos de bandas maiores para troca de dados.

Então, por que devemos estudar esse campo da computação?

Figura 1 | Criação de dados a cada minuto em 2021



Fonte: adaptada de Visual Capitalist.

Como vemos na imagem acima, justamente por essa enorme quantidade de dados massivos, sendo gerados em formatos não estruturados e em uma velocidade cada vez mais acelerada, que chamamos de *Big Data*, é o que responde nossa pergunta.

Dado esse contexto, os pesquisadores tinham como motivação o desenvolvimento de uma tecnologia que lidasse com:

1. Estrutura de arquivos capaz de atuar com dados massivos.
2. Paradigma que comporte escalabilidade elástica.
3. A não necessidade de implementar todos os conceitos que os bancos de dados relacionais utilizam.
4. Infraestrutura tolerante a falhas e que permita computação paralela.

Com este cenário, tivemos mudanças nos paradigmas arquiteturais na criação da solução, que tem como requisito:

1. Estrutura de computação paralela.
2. Esquemas de particionamentos, onde cada nó do *cluster* tenha recursos computacionais independentes (memória, disco, rede, CPU, etc).
3. Tolerância a falhas.
4. Escalabilidade horizontal e elástica.

Quando falamos sobre escalabilidade de infraestrutura, temos duas categorias principais: escalabilidade vertical e horizontal. Quando escalamos verticalmente, estamos aumentando o poder de computação de uma máquina específica, adicionando, por exemplo, mais disco ou memória. Já na escalabilidade horizontal, ao invés de aumentar o poder de uma máquina em específico, adicionamos uma nova máquina ao nosso *cluster*. Com isso, começamos a lidar com o hardware como *commodity*, onde podemos adicionar ou remover máquinas de um *cluster* sob demanda.

Neste contexto, então, surgiu o Hadoop - e o MapReduce. Ele foi criado por um time de desenvolvedores e pesquisadores da Yahoo, com raízes em trabalhos acadêmicos que foram desenvolvidos na Google. Posteriormente, foi criada a empresa Hortonworks (mais tarde comprada pela Cloudera).

O Hadoop surgiu com duas estruturas: o Hadoop HDFS (*Hadoop Distributed File System*), baseado no artigo de 2003, *The Google file system* (O sistema de arquivos da Google, em tradução livre) e com o Hadoop MapReduce, inspirado no artigo, escrito em 2004, *MapReduce: simplified data processing on large cluster* (MapReduce: processamento de dados simplificado em grandes *clusters*, em tradução livre).

O Hadoop HDFS dispõe de uma arquitetura de persistência dos arquivos que permite cobrir todos os pontos de requisitos levantados anteriormente, oferecendo tolerância a falhas por particionar os arquivos em múltiplos discos, e de maneira replicada. Já o MapReduce fornece um paradigma de processamento que, usando os nós do *cluster*, faz o processamento de grandes cargas de dados de maneira paralela, por meio de um processo chamado de Map; depois, agrupa os resultados intermediários em um resultado final, por meio de um segundo processo chamado Reduce.

## Principais aplicações do cotidiano



O MapReduce, como vimos no bloco anterior, tem seu comportamento atrelado a duas funções:

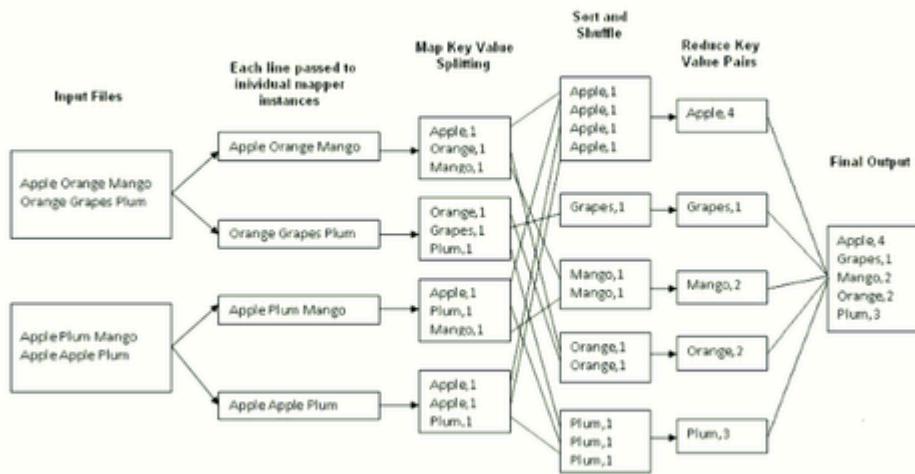
- Map: com os dados originais, fazemos um particionamento para que seja processado de maneira paralela, reduzindo o problema maior em subproblemas.
- Reduce: com os resultados parciais, oriundos de cada processo da etapa de Map, fazemos a agregação para ter o resultado final.

Esta abordagem, de quebrar o problema maior em problemas menores, que podem ser resolvidos de maneira independente, segue a ideia do “dividir para conquistar”, um dos paradigmas de programação.

# Bancos de Dados Não Relacionais

 ampli

Figura 2 | Fluxo de processo do MapReduce



Fonte: Wikimedia Commons.

Como vemos no exemplo acima, podemos fazer uso do MapReduce para computar o número de ocorrências de uma determinada palavra (ou item) dentro de um conjunto. Neste cenário da imagem, temos 2 arquivos de entrada, cada arquivo contendo 2 linhas, onde estão preenchidos nomes de frutas. Em um primeiro mapeamento, são desmembrados os arquivos originais em linhas. Estas linhas, por sua vez, são divididas em mapeamentos de chave/valor, onde cada chave será o nome da fruta e o valor que, neste primeiro momento, será 1 (pois como chegamos a nível de palavra, ela é apenas uma ocorrência). Após esse mapeamento chave/valor, fazemos uma ordenação lexicográfica (por ordem alfabética) e em sequência fazemos o Reduce, que é a união de todos os elementos iguais e a soma de suas ocorrências (repete-se a chave e soma-se o valor dos elementos com chaves iguais). E, como resultado final, temos um novo Reduce, unindo todas as palavras (chaves) seguidas pelo número de ocorrências (valores).

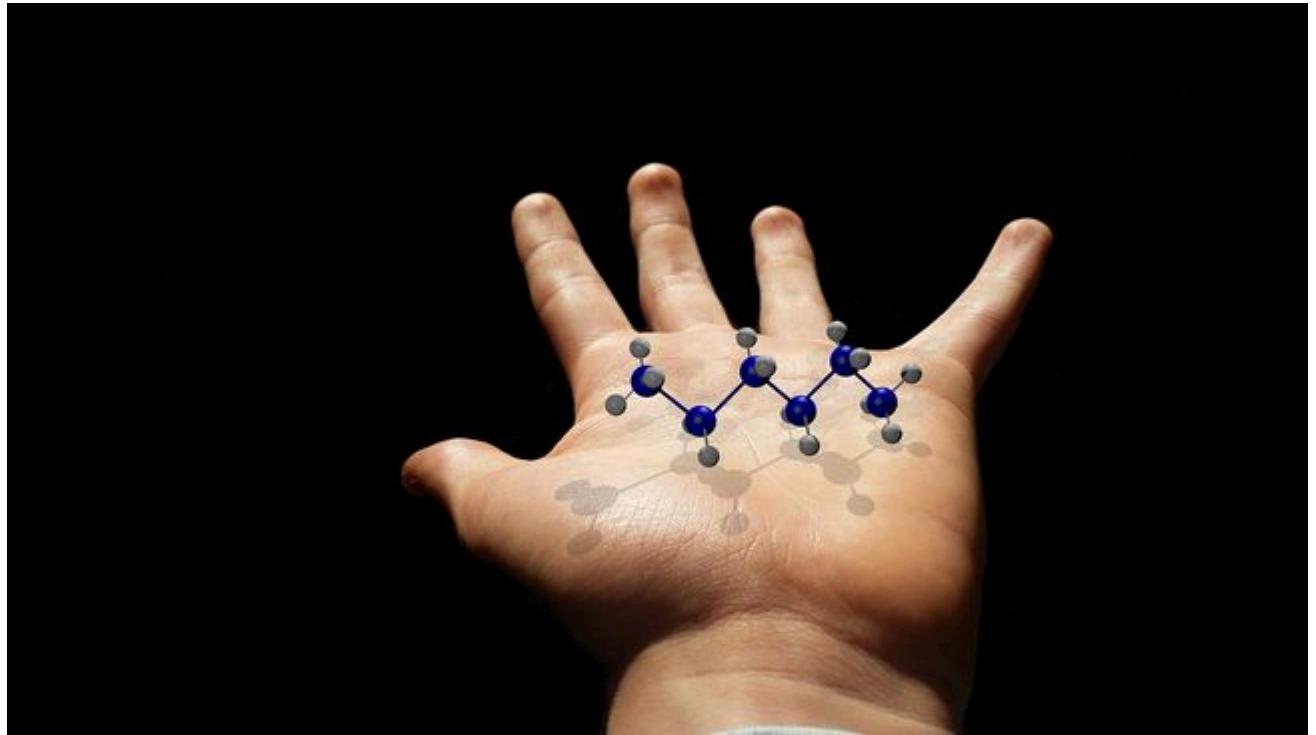
Os ganhos trazidos pelo uso do MapReduce se dão na possibilidade, como vemos acima, de conseguir fazer múltiplos processamentos em paralelo, trazendo a possibilidade de que tenhamos uma vazão de resultados mais eficiente do que em um processamento sequencial, além de poder utilizar N máquinas com uma configuração mediana, ao invés de termos que comprar 1 máquina com configuração muito mais elevada. Essa possibilidade de escalar horizontalmente, de maneira elástica (adicionam-se e removem-se máquinas de *clusters* sob demanda), faz com que tenhamos uma maior eficiência de investimentos em infraestrutura para a tecnologia das empresas, universidades e centros de pesquisa.

Com essa capacidade, podemos utilizar o MapReduce, assim como outras tecnologias de *Big Data*, em diversas aplicações, como:

- Redes sociais: processamento das interações entre os indivíduos, sugestão de novos conteúdos, publicidade.
- *Streaming* de músicas e vídeos: análise do comportamento de navegação de um usuário e sugestão de novos conteúdos para serem consumidos, reconhecimento de músicas que o usuário pulou e apresentação de uma diferente em sequência ou, se o usuário tiver sinalizado positivamente (como uma curtida), indicação de outra de mesmo gênero logo em sequência.
- Sistemas antifraude: processar em tempo real se um padrão de comportamento do usuário no aplicativo do banco é similar ao histórico dele ou se é um comportamento anômalo, validação facial na abertura de contas e autenticação de documentos.
- Câmeras de segurança: análise de imagens para detectar pessoas desaparecidas ou foragidas no meio de multidões, reconhecer objetos abandonados em regiões inadequadas (maleta em um aeroporto) e, com isso, aumentar a segurança.
- Carros autônomos: processar em tempo real os dados coletados pelas câmeras e sensores do veículo, identificando obstáculos, sinalização e faixas da via e mantendo a velocidade adequada, além de evitar acidentes.
- Astronomia e astrofísica: comportar para armazenamento e análise imagens e dados coletados de satélites e sensores que, via de regra, têm uma quantidade muito grande de informações.
- Varejo: processar o comportamento de consumo do cliente, analisando o que ele costuma comprar, com qual periodicidade e os itens que ele acaba de adicionar ao carrinho, podendo sugerir, assim, uma venda cruzada com itens semelhantes ou relacionados.
- Mercado financeiro: analisar dados correlacionados ao mercado em tempo real, como movimentação do preço das ações, *commodities*, notícias vinculadas com a empresa analisada e comportamentos de padrão em múltiplos mercados.

Reparam que, citando apenas algumas aplicações (poderíamos expandir para os setores de saúde, agronegócio, seguros, manutenção de equipamentos além de tantas outras), vemos que o uso do Hadoop com o MapReduce tem um enorme valor comercial e leque de cenários onde podemos usá-lo. Em todos esses casos é importante conseguir processar um grande volume de dados em um curto espaço de tempo. No caso do carro, se não tivermos uso de tecnologias eficientes, podemos provocar um acidente. No varejo, perder oportunidades de vendas. No *streaming*, ao não apresentar sugestões que o usuário goste, podemos perder uma assinatura.

## Instalação do MapReduce



Para realizar a instalação do MapReduce, assim como do Hadoop, vamos seguir os seguintes passos recomendados pela Apache Foundation.

O que é necessário ter disponível como pré-requisito?

O Apache Hadoop tem suporte, como plataforma de desenvolvimento e produtiva, para os sistemas operacionais GNU / Linux e, também, para Windows.

Vamos abordar aqui a instalação em sistemas operacionais baseados em Linux, por ser acessível a todos os usuários, tendo inúmeras distribuições gratuitas e com ampla comunidade - como as distribuições do sistema operacional Linux: Ubuntu e Debian.

Os pré-requisitos, quando vamos efetuar a instalação em Linux, é a ferramenta ssh e uma sdk (*software development toolkit*) do Java, em suas versões 8 ou seguintes.

Com estes pré-requisitos, ssh e Java SDK, realizaremos o *download* de uma distribuição Hadoop em um dos provedores oficiais associados à Apache Hadoop. As versões, assim como seus *links* para *download* podem ser encontradas no portal da Apache.

Atualmente, o Apache Hadoop se encontra disponível, em sua *release* mais recente, na versão 3.3.3.

Comando 1 | Fazendo o download da versão mais recente

```
1 $ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.3/  
hadoop-3.3.3.tar.gz
```

Fonte: Elaborado pelo autor.

Ao baixar o arquivo hadoop-3.3.3.tar.gz, teremos o arquivo compactado, contendo todos os componentes principais do Hadoop - como o HDFS, o MapReduce e o Yarn, um gerenciador de recursos para otimizar o desempenho e gestão de ambiente do Hadoop.

Agora, vamos descompactar o arquivo hadoop-3.3.3.tar.gz:

Comando 2 | Descompactando o arquivo

```
1 $ tar -zxvf hadoop-3.3.3.tar.gz
```

Fonte: Elaborado pelo autor.

Agora teremos a pasta com o nome da distribuição, hadoop-3.3.3.

Navegando pelo repositório recém-criado, da distribuição, navegaremos até a pasta etc/hadoop e vamos editar o arquivo hadoop-env.sh com as seguintes parametrizações:

Comando 3 | Definindo variáveis de ambiente

```
1 # set to the root of your Java installation  
    export JAVA_HOME=/usr/java/latest
```

Fonte: Hadoop: Setting up a Single Node Cluster, Apache Hadoop.

Com esta parametrização realizada, o Apache Hadoop terá acesso à SDK do Java para sua execução.

Para testar o correto funcionamento podemos digitar o comando:

Comando 4 | Teste da instalação

```
1 $ bin/hadoop
```

Fonte: Hadoop: Setting up a Single Node Cluster, Apache Hadoop.

Se a instalação tiver sido executada com sucesso, teremos exibida em nosso terminal a documentação para o uso do Hadoop.

Com isso, finalizamos a instalação do Apache Hadoop, tendo acesso às suas ferramentas de Hadoop Distributed File System (Hadoop HDFS), MapReduce e Yet Another Resource Negotiator (Hadoop YARN).

Na nossa próxima aula, exploraremos o uso da ferramenta.

## Videoaula: Introdução MapReduce



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta aula falaremos sobre o MapReduce, componente contido no Apache Hadoop. Exploraremos em qual contexto tecnológico os desenvolvedores se encontravam quando surgiu a ferramenta, para quais propósitos ela foi desenhada e como foi sua criação. Além disso, vamos

# Bancos de Dados Não Relacionais

mostrar por que o MapReduce - e o Hadoop - são ferramentas tão poderosas para termos em nossa caixa de habilidades, mostrando seus ganhos e aplicações.

## Saiba mais



Para conhecer um pouco mais sobre o framework MapReduce e a ferramenta Apache Hadoop como um todo, é recomendável uma visita ao portal da própria ferramenta. A seguir, indicamos páginas de interesse do Apache Hadoop:

["Apache Hadoop"](#)

["Hadoop: Configurando um cluster de nó único"](#)

["Repositório do Apache Hadoop no Github"](#)

Esses conteúdos estão originalmente em língua inglesa. Caso tenha dificuldade com esse idioma, recomenda-se o uso do recurso de tradução do seu *browser*.

---

## Referências



APACHE HADOOP. Hadoop: Setting up a Single Node Cluster. **Apache Hadoop**. Disponível em: [https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone\\_Operation](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone_Operation). Acesso em: 19 jun. 2022.

VISUAL CAPITALIST. From Amazon to Zoom: What Happens in an Internet Minute In 2021? **Visual Capitalist**, 2021. Disponível em: <https://www.visualcapitalist.com/from-amazon-to-zoom-what-happens-in-an-internet-minute-in-2021/>. Acesso em: 18 jun. 2022.

## Aula 2

Interfaces do usuário

### Introdução



Na aula de hoje daremos sequência aos nossos estudos sobre a tecnologia Hadoop MapReduce, abordando as principais categorias de interfaces de usuários para carga útil, gestão de ambiente e de entradas e saídas para nossas tarefas.

Na aula passada, fizemos a instalação do Hadoop 3.3.3. em uma máquina com Sistema Operacional Linux. Utilizando essa instalação, que foi feita de maneira local e *standalone*, realizaremos experimentos iniciais com um código de exemplo da Apache.

Ao final desta aula, seremos capazes de implementar o MapReduce em Java, executar experimentos e propor modificações que julgarmos adequadas - dado que teremos acesso ao código.

O material desta aula está baseado na documentação oficial da Apache Foundation, que está disponível nas referências para que você se aprofunde nos estudos.

## Carga útil



Na última aula aprendemos sobre o MapReduce: um componente para processamento massivo de dados que faz uso de recursos de computação distribuída para analisar cargas de dados em paralelo.

Vimos que o MapReduce está integrado ao Hadoop e estudamos o cenário de desafios onde a tecnologia foi desenvolvida, os problemas para os quais foi pensada e diversas aplicações nas quais podemos ter ganhos de negócios.

Nesta aula, falaremos mais sobre como configurar e executar uma tarefa no MapReduce.

Recapitulando sobre o MapReduce, como estudado na aula anterior, seu comportamento de execução está atrelado a duas principais funções:

- Map: com os dados originais, fazemos um particionamento para que os dados sejam processados de maneira paralela, reduzindo o problema maior em subproblemas.
- Reduce: com os resultados parciais calculados dos subproblemas, oriundos de cada processo da etapa de Map, fazemos a agregação para termos o resultado final.

Para cada uma dessas capacidades, de Map e de Reduce, temos interfaces do usuário para que façamos uso e ganho das implementações já realizadas pelo Hadoop MapReduce, aproveitando das competências de paralelização, monitoramento e tolerância a falhas.

Quando pensamos nessas interfaces do usuário, podemos agrupar em algumas grandes categorias:

- *Payload* (Carga Útil): aqui é onde estão contidas as funcionalidades centrais do Hadoop MapReduce, como as interfaces *Partitioner* (Particionamento), *Counter* (Contagem) e, as principais, as interfaces *Mapping* e *Reducer* (Mapeamento e Redução).
- *Job Configuration* (Configuração das tarefas): expõe uma interface *Job* (Tarefa), que permite a configuração da tarefa.
- *Task Execution and Environment* (Ambiente e execução das tarefas): permite a gestão de memória na execução da tarefa e as parametrizações para o *Map*, *Shuffle* e *Reduce*.
- *Job Submission and Monitoring* (Envio e monitoramento das tarefas): com a interface *Job* temos acesso para interagir com o *Resource Manager* (Gestor de recursos), onde podemos submeter as tarefas a serem executadas, monitorar o progresso, ter status do cluster de execução do Hadoop MapReduce, acessar logs e relatórios das tarefas.
- *Job Input* (Entradas das tarefas): fornece uma interface, *InputFormat* (formato de entrada), para especificar e validar as tarefas de entradas que são recebidas pelo programa.
- *Job Output* (Saídas das tarefas): fornece uma interface, a *OutputFormat* (formato de saída), para, assim como a *InputFormat*, especificar e validar as saídas geradas pelo programa.

Como podemos observar, o *framework* Hadoop MapReduce nos fornece um amplo leque de funcionalidades já previamente implementadas, das quais podemos fazer uso via as Interfaces que são disponibilizadas. Isso tudo nos traz diversos ganhos:

1. Velocidade de entrega de projetos.
2. Uso da tecnologia por um maior número de desenvolvedores (não é necessário que o programador saiba implementar o algoritmo do zero).
3. Desempenho eficiente (evita o risco de implementações com baixa qualidade técnica, que não tenham eficiência para processamento paralelo).
4. Manutenção no código (por ser padronizado, não se tem um alto dispêndio de tempo para que um desenvolvedor comprehenda a implementação feita).

Percebemos, então, que o MapReduce, do Hadoop, é muito eficiente - tanto no custo de processamento quanto no de criação e manutenção de sistemas. No próximo bloco, detalharemos mais cada uma dessas seis grandes categorias de interfaces de usuário e mergulharemos mais a fundo nas principais.

## Configuração e execução de tarefas do ambiente



Vimos que as interfaces de usuário do Hadoop Map Reduce podem ser classificadas em seis grandes categorias, sendo elas:

- *Payload* (Carga Útil).
- *Job configuration* (configuração das tarefas).
- *Task execution and environment* (ambiente e execução das tarefas).
- *Job submission and monitoring* (envio e monitoramento das tarefas).
- *Job input* (entradas das tarefas).
- *Job output* (saídas das tarefas).

Esta categorização é a mesma que a Apache Foundation, no projeto do Hadoop MapReduce, define.

Aprofundaremos-nos agora nas duas primeiras categorias, especialmente na Carga útil (*Payload*), por ser a categoria central do *framework*.

### ***Payload* (carga útil)**

Nesta categoria, temos os principais métodos do algoritmo em si, que são as interfaces *Mapper* e *Reducer*.

A ***Mapper*** faz o mapeamento da entrada, no formato chave e valor, para um novo conjunto intermediário de pares de chave e valor. Os ***Maps*** são tarefas unitárias que transformarão os

# Bancos de Dados Não Relacionais

registros de entrada em registros intermediários. Estes registros intermediários não precisam, necessariamente, ser do mesmo tipo que os registros de entrada. Cada entrada pode mapear para N pares de saída, com N sendo maior ou igual a 0.

Cada um destes valores intermediários, que está associado a uma dada chave de saída, será posteriormente agrupado pelo *framework* e será enviado para o método de Redução computar o resultado e a saída final.

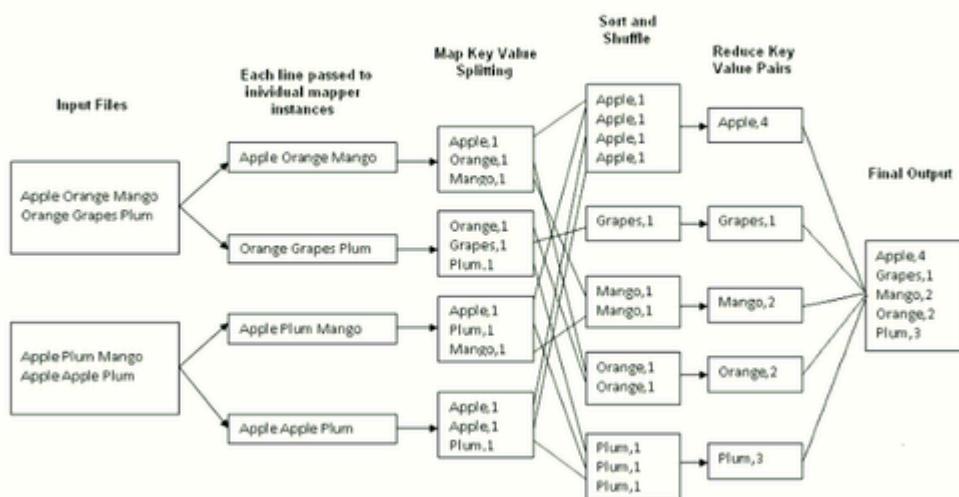
A quantidade de mapeamentos é geralmente determinada baseando-se no tamanho das entradas.

O *Reducer* faz a redução dos conjuntos intermediários de pares de chave e valor que compartilham uma mesma chave para um conjunto menor de valores. O número de tarefas de redução que serão computadas é definida pelo usuário via parâmetro, na interface de *Job*.

O processo do *Reducer* tem, de maneira geral, três principais fases: *shuffle* (embaralhamento), *sort* (ordenação) e redução. As fases de *shuffle* e de *sort* são realizadas de maneira simultânea; enquanto os mapeamentos são recebidos pelo *Reducer*, ele vai realizando também a mescla das duplas de chaves e valores.

Esses processos podem ser visualizados na imagem abaixo:

Figura 1 | Fluxo de processo do MapReduce



Fonte: Wikimedia Commons

Além do *Mapper* e do *Reducer*, temos também as capacidades *Partitioner* (particionador) e *Counter* (contador).

O *Partitioner* é chamado para fazer o particionamento do espaço das chaves (no nosso exemplo acima, das palavras). A chave é utilizada para controlar o particionamento dos dados. O número

de partições vai ser igual ao número de reduções que o trabalho executará. Desta forma, ele controla também para qual das tarefas de redução uma chave intermediária será enviada. Já o **Counter** é um recurso focado nas estatísticas do processamento. O *Counter* pode ser chamado pelos demais métodos, de mapeamento e redução, para fazer o relatório dos números e métricas de execução.

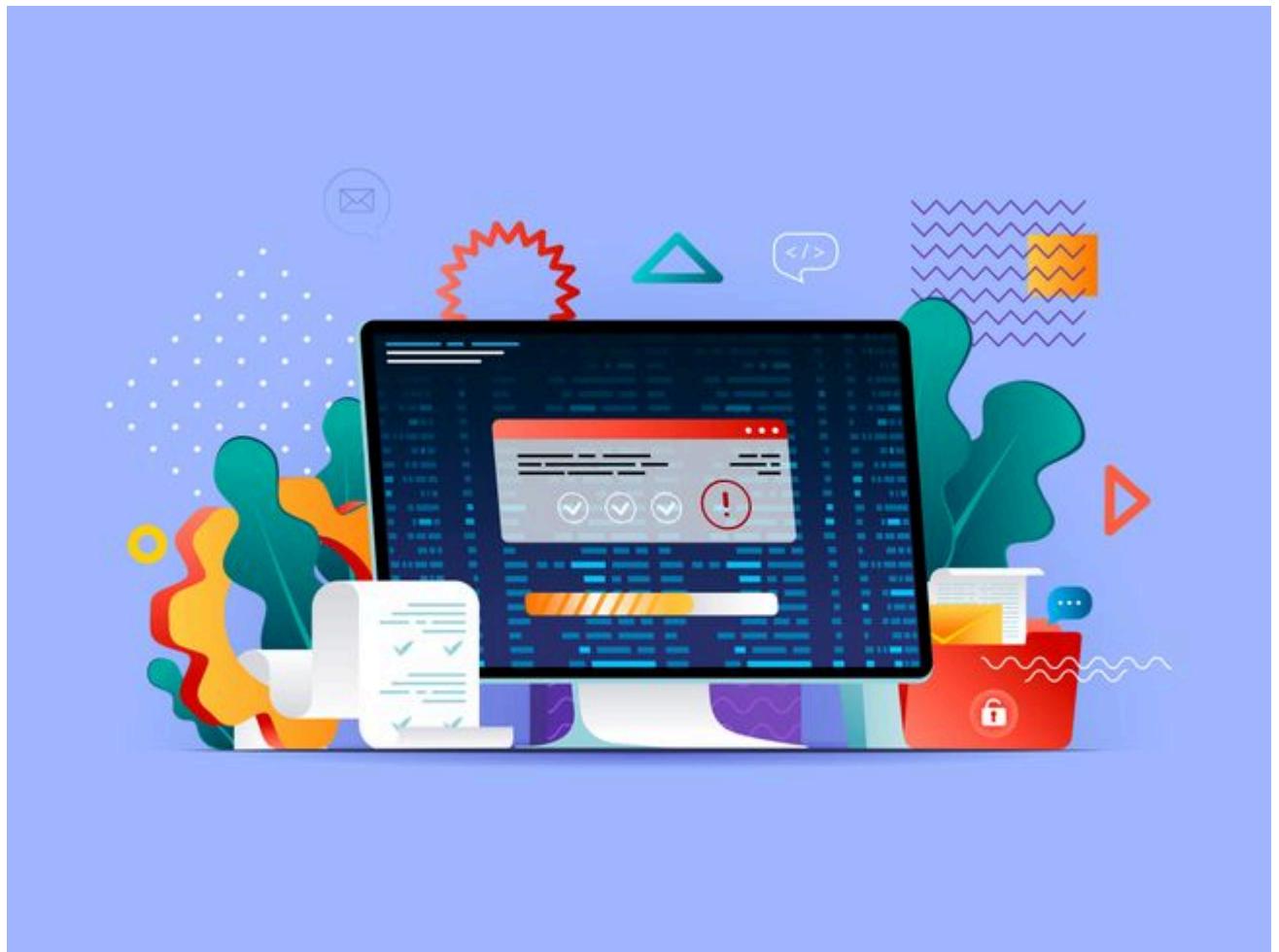
### **Job Configuration (Configuração das tarefas)**

Nesta categoria temos a Interface **Job**.

A *Job* apresenta a configuração para as tarefas que serão executadas no MapReduce. Ela é a principal interface para o usuário descrever a estrutura de execução da tarefa.

Teremos um amplo leque de parâmetros, alguns sendo de mais simples definição, como o `Job.setNumReduceTasks(int)`, e outros terão uma maior complexidade para ter uma definição, como o `Configuration.set(JobContext.NUM_MAPS, int)`. Vale a pena ler as bibliografias, caso queira se aprofundar nessas documentações.

## Envio, entrada e saída de trabalho



Dando continuidade aos estudos do Bloco 2, vamos falar mais sobre as quatro últimas categorias.

## ***Task execution and environment* (ambiente e execução das tarefas)**

Permite a gestão de memória na execução da tarefa e as parametrizações para o *Map*, *Shuffle* e *Reduce*.

## ***Job submission and monitoring* (envio e monitoramento das tarefas)**

A *Job*, já vista nas seções anteriores, é também a principal interface pela qual se interage com o *Resource Manager*. O fluxo de envio de uma tarefa envolve:

- Verificar e validar as especificações de entrada e saída da tarefa.
- Calcular os valores de *InputSplit* para a tarefa.
- Configurar as informações necessárias para o *DistributedCache* da tarefa.
- Copiar o .jar e a configuração da tarefa para o diretório do sistema MapReduce no *FileSystem*.
- Enviar as tarefas para o *ResourceManager* e monitorar seu status.

## ***Job Input* (entradas das tarefas)**

Fornece uma interface, a *InputFormat* (formato de entrada), para especificar e validar as tarefas de entradas que são recebidas pelo programa. A estrutura do MapReduce depende do *InputFormat* para:

- Validar as especificações de entrada da tarefa.
- Dividir o arquivo de entrada em instâncias lógicas *InputSplit*, onde cada uma é direcionada para um *Mapper* individual.
- Fornecer o *RecordReader* usado para coletar os registros de entrada do *InputSplit* para processamento pelo *Mapper*.

Os *InputSplit* representam os dados que serão processados por *Mappers* individualmente.

## ***Job Output* (saídas das tarefas)**

Fornece uma interface, a *OutputFormat* (formato de saída), para, assim como a *InputFormat*, especificar e validar as saídas geradas pelo programa. A estrutura do MapReduce depende do *OutputFormat* para:

- Validar as especificações de saída da tarefa, como a não existência do diretório de saída da nossa tarefa.
- Fornecer a implementação *RecordWriter* usada para a gravação dos arquivos de saída da tarefa. Os arquivos de saída são armazenados em um *FileSystem*.

Agora que vimos sobre as Interfaces de Usuário que são disponibilizadas para nosso uso, vamos fazer um experimento prático simples do Hadoop MapReduce, que instalamos na última aula. O experimento será baseado no tutorial de MapReduce (APACHE HADOOP, 2021).

Nosso experimento receberá dois arquivos de entrada e devolverá a contagem de ocorrências para cada palavra contida nos arquivos.

Código-Fonte 1 | Programa WordCount.java

```
1 import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
                        ) throws IOException, InterruptedException {
            StringTokenizer itr = new
StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());

                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
        extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
                          Context context
                          ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Fonte: Apache Hadoop, MapReduce Tutorial. Apache Hadoop.

O código acima implementa o nosso programa de contagem de palavras e foi disponibilizado no Tutorial do Hadoop MapReduce, que se encontra nas referências.

Com o arquivo já criado, vamos compilar o mesmo:

Comando 1 | Descompactando o arquivo

```
1 $ bin/hadoop com.sun.tools.javac.Main WordCount.java  
2 $ jar cf wc.jar WordCount*.class
```

Fonte: Apache Hadoop, MapReduce Tutorial. Apache Hadoop.

Agora criaremos um repositório chamado *input*, que será onde vamos disponibilizar os arquivos de entrada do nosso programa. Faremos dois arquivos com os seguintes conteúdos:

Arquivo: **arquivo01**

Conteúdo: aula de hadoop map reduce

Arquivo: **arquivo02**

Conteúdo: meu primeiro experimento de hadoop map reduce com map e com reduce

Com estes dois arquivos criados na pasta *input*, vamos rodar nosso programa:

# Bancos de Dados Não Relacionais ampli

Comando 2 | Executando o programa

```
1 $ bin/hadoop jar wc.jar WordCount input output
```

Fonte: Apache Hadoop, MapReduce Tutorial. Apache Hadoop.

Após a execução com sucesso, basta checarmos o resultado:

Figura 2 | Testando Hadoop: Saída

```
douglas@douglas-VirtualBox:~/Documents/hadoop/output$ ls
part-r-00000 _SUCCESS
douglas@douglas-VirtualBox:~/Documents/hadoop/output$ cat part-r-00000
aula      1
com       2
de        2
e         1
experimento    1
hadoop     2
map       3
meu       1
primeiro    1
reduce     3
douglas@douglas-VirtualBox:~/Documents/hadoop/output$
```

Fonte: elaborada pelo autor.

Pronto! Agora, além de sabermos com mais detalhes sobre o funcionamento do MapReduce e suas interfaces no Hadoop, também fizemos nosso primeiro experimento prático com a ferramenta.

## Videoaula: Interfaces do usuário



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta aula tratamos das interfaces de usuário do MapReduce, componente contido no Apache Hadoop. Exploramos como essas interfaces são mapeadas e documentadas em categorias e teremos uma visão de cada uma delas, com maior foco na principal, que se chama Payload. Além disso, teremos um experimento prático executando o algoritmo do MapReduce em nossas máquinas, com a instalação feita na última aula.

## Saiba mais



Para conhecer um pouco mais sobre o *framework* MapReduce, é recomendável fazer uma visita ao portal da Apache Hadoop. A seguir, indicamos uma página de interesse dessa ferramenta: ["Tutorial do MapReduce"](#).

Esse conteúdo está originalmente em língua inglesa. Caso tenha dificuldade com esse idioma, recomenda-se o uso do recurso de tradução do seu *browser*.

---

## Referências



APACHE HADOOP. MapReduce Tutorial. Apache Hadoop. Disponível em:  
[https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#MapReduce\\_-\\_User\\_Interfaces](https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#MapReduce_-_User_Interfaces). Acesso em: 23 jun. 2022

## Aula 3

Introdução ao transactions NoSQL

### Introdução



Na aula de hoje, daremos sequência aos nossos estudos e nos aprofundaremos nos bancos de dados - mais especificamente nos bancos de dados não relacionais (também conhecidos como NoSQL).

Destacaremos bases de dados e mostraremos que, para alguns cenários, os bancos não relacionais têm uma aderência maior, por questões de desempenho ou até mesmo pelas características dos dados.

Depois, conheceremos as grandes categorias de bancos de dados NoSQL e discutiremos sobre as propriedades ACID.

Ao final deste material, estarão disponíveis referências para que você se aprofunde nos estudos dos tópicos abordados.

## Introdução ao banco de dados NoSQL



Enquanto os estudos da computação e dos processamentos de dados evoluíam, foram necessárias pesquisas e desenvolvimento de soluções para realizar o armazenamento dos dados de insumos para os programas e também para seus resultados. Além do armazenamento em si, diversos problemas traziam demandas específicas, como recuperação (busca) de registros e atualização de um dado armazenado.

Foram, então, sendo geradas diversas abordagens do que podemos chamar de bancos de dados.

Tivemos (e temos, em aplicações específicas) grande uso de armazenamentos em arquivos, que são estruturas mais simples. Estes arquivos - que podem literalmente ser um arquivo de texto simples - tinham abordagens que traziam seus prós e contras: se um arquivo com registros de tamanho fixo é utilizado, tem-se a vantagem de fácil pesquisa entre os registros. Se se deseja acessar um dado registro, é apenas necessário resgatar o número de bytes que um registro tem e multiplicar por  $N-1$ , onde  $N$  é a posição do registro. Nesse cenário, em um arquivo onde o registro tenha 100 bytes, e deseja-se acessar o primeiro registro, sabe-se que este registro se inicia no byte 0 do arquivo. Se se deseja acessar o registro de número 10, começa-se lendo o arquivo da posição de byte 900. Chamamos essa manipulação de *offset*. Outra abordagem de arquivo é ter o tamanho variável, mas utilizar um delimitador entre os registros e os campos, como vírgula ou ponto-e-vírgula. A vantagem da primeira abordagem, como vimos, é a facilidade de se acessar um registro em determinada posição. A vantagem da segunda é uma maior otimização de memória consumida (podemos ter nomes com 50 caracteres e outros com 20,

enquanto na primeira abordagem todos teriam um tamanho específico e o espaço restante com algum símbolo coringa seria preenchido, como, por exemplo, com espaços).

Outras abordagens diversas foram desenvolvidas, como os bancos de dados orientados a objetos. Porém, a que mais se consolidou, foi a solução conhecida como bancos de dados relacionais.

Nos bancos de dados relacionais, temos uma estrutura de modelagem do banco onde podemos estruturar tabelas (também conhecidas como entidades) e relacionamentos. Existem formas de se modelar buscando deixar a estrutura mais robusta e consistente. Uma destas formas de modelação são as conhecidas Formas Normais, para normalização das bases.

Diversos SGBDs (Sistema de Gerenciamento de Banco de Dados), em inglês DBMS (*Data Base Management System*), foram desenvolvidos e consolidados no mercado nesta abordagem Relacional, como o MySQL, Oracle, PostgreSQL e o Microsoft SQL Server. Todos estes SGBDs utilizam como linguagem padrão o SQL (do inglês, *Structured Query Language*).

Ocorre que os bancos de dados relacionais trazem consigo algumas amarrações, como cumprir as características ACID (que discutiremos nos próximos blocos) e demandam de um *schema* (estrutura) bem definido. Estas características são fundamentais e trazem ganhos para muitos cenários, como para armazenar informações transacionais financeiras de uma instituição bancária, por exemplo.

Outros cenários, na contramão, já não fazem questão de garantir o ACID e preferem trocar esta amarração por um incremento de performance computacional no processamento. Algumas também podem lidar com dados que são, intrinsecamente, desestruturados - o que torna inviável ou deveras custoso modelar o *schema* da base de dados que pode, inclusive, ser mutável ao longo do tempo.

Para lidar com esses problemas supracitados temos uma gama de tecnologias conhecidas como NoSQL (de “No SQL” ou “Não SQL”), que detalharemos no próximo bloco.

## Entendendo as transações em NoSQL



Como vimos na seção anterior, os bancos de dados NoSQL são uma categoria criada em contraposição aos bancos de dados relacionais, que utilizam a modelagem em forma de tabelas (também conhecidas como entidades) e seus relacionamentos.

Já que falamos sobre o que são os bancos de dados “SQL” relacionais, falaremos sobre os NoSQL.

Quando citei que o NoSQL é uma contraposição aos relacionais é porque, de fato, o NoSQL é uma grande categoria onde temos agrupado algumas categorias de bancos de dados bem distintas entre si.

Temos quatro grandes categorias, mapeadas nas pesquisas acadêmicas mais comuns, de bancos de dados não relacionais. As categorias são: bancos de dados baseados em documentos, bancos de dados colunares, bancos de dados baseados em chave-valor e banco de dados baseados em grafo.

Cada categoria tem características muito específicas e distintas entre si, sendo adequado seus usos em cenários que tenham semelhança da natureza do problema com as soluções fornecidas.

A seguir, descrevemos brevemente cada categoria e fornecemos um exemplo de cenário onde o banco de dados NoSQL pode facilmente ser utilizado:

## **Bancos de dados baseados em documentos**

É um banco de dados onde seus registros ficam armazenados no formato de documentos na estrutura JSON, com seus campos e respectivos valores associados. Este tipo de banco de dados NoSQL tem se tornado muito frequente por ter uma estrutura extremamente flexível e adaptável. Cada registro - documento - pode ter uma estrutura variável, que se adapta ao dado armazenado. Por essa flexibilidade, alguns desenvolvedores gostam de usar bancos dessa

# Bancos de Dados Não Relacionais

natureza mesmo em cenários onde o adequado seria usar um banco relacional, o que deve ser evitado.

Exemplo de tecnologia: MongoDB.

Exemplo de aplicação: cadastros de livros com campos dinâmicos (o número de autores pode ser variável, pode haver informações de tradutores e afins).

## Bancos de dados colunares

São bancos que podem crescer não somente verticalmente (por meio das linhas) como também horizontalmente (por meio das colunas). É uma estrutura que suporta diversas aplicações focadas em *Big Data*, como a Hadoop HBase.

Exemplo de tecnologia: Cassandra.

Exemplo de aplicação: lista de reprodução em ferramentas de *streaming* de música.

## Bancos de dados baseados em chave-valor

São bancos de dados com uma representação mais simplificada, onde cada registro de baseia simplesmente em uma associação de uma chave com um valor atribuído a ela.

Exemplo de tecnologia: Redis.

Exemplo de aplicação: cache de dados.

## Bancos de dados baseados em grafos

Grafo é um conceito da matemática. Nele, de maneira simplificada, podemos representar relacionamentos, por meio de vértices que podem se conectar - ou não - via arestas. Nos bancos de dados, os vértices representam os objetos armazenados e as arestas, os seus relacionamentos.

Exemplo de tecnologia: Neo4J.

Exemplo de aplicação: redes sociais (vértice sendo o usuário e aresta representando sua conexão)

## Propriedade ACID



Como vimos nas seções anteriores, o principal paradigma que dominou os bancos de dados por muito tempo (e que se mantém forte) é a de bancos de dados relacionais.

Vimos também que, apesar disso, os bancos de dados NoSQL (não relacionais) têm crescido e tomado grande relevância. Apesar de os colocarmos em uma grande caixa, NoSQL, estes produtos dispõem de peculiaridades e formas de funcionamento específicas.

Vimos também que o que faz, em um dado projeto, escolhermos entre um banco relacional e um NoSQL é a natureza do problema - no caso, se o banco de dados tem uma estrutura pouco definida ou se preferimos trocar aderência das transações ao ACID por desempenho computacional.

Mas, o que é a tal de ACID?

ACID é um conjunto de propriedades relacionadas às transações de bancos de dados. A sigla deriva da abreviação de quatro características: Atomicidade, Consistência, Isolamento e Durabilidade.

Os bancos de dados relacionais, por padrão, são aderentes a estas propriedades e isto tem grande valia no que tange à confiabilidade dos dados e aplicação em contextos transacionais, como movimentações financeiras em instituições bancárias.

## Atomicidade

Esta característica diz que uma dada transação deve ser atômica, ou seja, indivisível. A aplicação dessa regra nos garante que uma dada transação tem que ser integralmente executada com sucesso ou, apresentando alguma falha, deve ser desfeita sem impactar a base de dados. Com

isso, a base não reflete resultados parciais ou intermediários das transações: ou ela reflete o resultado da execução completa ou mantém os dados originais pré-transação.

## Consistência

Esta propriedade diz que toda transação deve levar de um estado consistente para outro também consistente. Ou seja, após uma transação, toda a estrutura da base de dados deve estar aderente às regras definidas para a base. Repare que, com esta definição, caso tenha alguma transação que fira a consistência definida, ela incorrerá em falha e, ao usar a atomicidade, será desfeita integralmente, o que levará para um estado consistente (o estado pré-transação). Além disso, temos que, quando uma operação realiza alterações nos dados, as próximas operações deverão refletir e utilizar os dados já atualizados.

## Isolamento

O isolamento tem suas raízes na gestão de concorrência. Seu foco é garantir que transações em paralelo não possam causar interferências cruzadas. Com isso, mantém o determinismo nas chamadas em paralelo, garantindo que os resultados (das operações realizadas sobre os dados) serão os mesmos que ocorreriam em uma execução sequencial.

## Durabilidade

Esta propriedade preza por garantir que os resultados de uma transação sejam persistidos de maneira a ter tolerância a falhas posteriores, como em quedas de energia, pane no sistema ou reinício da máquina. Com isso, a ideia é que os dados sejam persistidos em uma memória não volátil. Exemplo de memória volátil: memória RAM. Exemplo de memória não volátil: disco rígido (HD).

Nos bancos de dados NoSQL, de maneira geral, as transações não garantem o ACID. Com isso, podem ter sua execução parcial (quebra do princípio de atomicidade), uma consistência eventual (quebra da consistência), não determinismo em caso de concorrência (quebra do isolamento) e podem não ser tolerantes às falhas (falta de durabilidade). Já nas transações SQL essas características, por padrão, são integralmente atendidas. Como contraponto, nos bancos de dados relacionais costuma-se ter um desempenho inferior ao NoSQL, justamente pelo *overhead* gerado pela gestão a respeito dessas propriedades.

Por isso, ao desenhar sua arquitetura para atender um problema, seja de negócios ou científico, é importante entender quais são as características que as transações devem obedecer e, com isso, escolher a ferramenta mais adequada para solucioná-lo (seja SQL ou, se for NoSQL, saber escolher uma de suas subcategorias).

Uma observação pertinente é que algumas das tecnologias de bancos de dados NoSQL tem evoluído durante os últimos anos com foco em tentar garantir aderência ao ACID.

## Videoaula: Introdução ao transactions NoSQL



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo

# Bancos de Dados Não Relacionais

para assistir mesmo sem conexão à internet.

Nesta aula abordamos os bancos de dados NoSQL, sua diferença perante a estrutura dos bancos relacionais e, dentro do NoSQL, suas subcategorias.

Além disso, vemos o que representam as propriedades ACID e como elas impactam a escolha de ferramental para lidar com um problema de bases de dados.

## Saiba mais



Para conhecer um pouco mais sobre os bancos de dados NoSQL é pertinente conhecer seus tipos e as propriedades ACID e Teorema CAP. A seguir, indicamos páginas de interesse:

["ACID"](#) (este é um conteúdo de uma enclopédia virtual que pode ajudá-lo em consultas pontuais)

["O que são transações de ACID"](#) (este conteúdo está em inglês, se preferir utilize o recurso de tradução do seu browser para estudá-lo)

["Bancos de Dados NoSQL"](#)

["Teorema CAP"](#)

## Referências



APACHE HADOOP. Hadoop: Setting up a Single Node Cluster. **Apache Hadoop**. Disponível em: [https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone\\_Operation](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone_Operation). Acesso em: 19 jun. 2022.

CIFERRI, C. D. A. **Bancos de Dados NoSQL**. ICMS/USP, [s. d.]. Disponível em: <http://wiki.icmc.usp.br/images/1/18/SCC0542012017noSQL.pdf>. Acesso em: 22 jul. 2022.

IBM CLOUD EDUCATION. IBM. O que é o Teorema CAP?. [S.I.]. IBM, 2019. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/cap-theorem>. Acesso em: 19 jul. 2022.

MONGODB. MongoDB. What are ACID transactions?. [S.I.]. MongoDB, s. d.. Disponível em: <https://www.mongodb.com/basics/acid-transactions>. Acesso em: 19 jul. 2022.

VISUAL CAPITALIST. From Amazon to Zoom: What Happens in an Internet Minute In 2021? **Visual Capitalist**, 2021. Disponível em <https://www.visualcapitalist.com/from-amazon-to-zoom-what-happens-in-an-internet-minute-in-2021/>. Acesso em 18 jun. 2022.

WIKIPEDIA. Wikipedia, a encyclopédia livre. ACID. [S.I.]. Wikipedia, s. d.. Disponível em: <https://pt.wikipedia.org/wiki/ACID>. Acesso em: 19 jul. 2022

## Aula 4

Transações no MongoDB

### Introdução

# Bancos de Dados Não Relacionais

A ampli



Na aula de hoje seguiremos com nossos estudos de bancos de dados não-relacionais e nos aprofundaremos nas transações de um banco não-relacional de documento - mais especificamente, no banco de dados MongoDB - e como elas trazem ganhos para nossas aplicações.

Começaremos contextualizando o assunto de transações atômicas no MongoDB. Em seguida, conheceremos suas características, os custos e pontos negativos que vêm também associados ao uso desta funcionalidade. Ao final, veremos casos práticos, onde podemos utilizar as transações do MongoDB graças às suas propriedades ACID.

Ao final deste material, estarão disponíveis referências para que você se aprofunde nos estudos nos tópicos abordados.

Bons estudos!

## Transações no MongoDB - introdução

# Bancos de Dados Não Relacionais



Como estudado na última aula, temos diversas características e pontos a serem analisados quando vamos definir nossas tecnologias para armazenamento e gestão de bases de dados. Além da natureza e estrutura daquele dado, que pode nos dizer se devemos abordar o problema com um banco de dados não-relacional, de documento, chave-valor, de grafo ou colunar, temos também que considerar se é necessário cumprir os requisitos e propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

Vimos que as propriedades ACID nos trazem uma previsibilidade e governança com maior robustez sobre os nossos dados, o que é importante para diversos cenários empresariais e científicos.

Tradicionalmente, os bancos de dados relacionais nos garantem estas propriedades de maneira automática. Já os bancos de dados não relacionais, também chamados de NoSQL, em suas origens, não trazem consigo as propriedades ACID.

Pelo fato de os bancos de dados NoSQL originalmente serem orientados a cenários específicos e também relacionados a questões de performance, não era um grande empecilho a ausência das propriedades ACID. Porém, com a popularização de algumas ferramentas, foram realizados estudos pelos seus desenvolvedores e pela comunidade para incrementar as tecnologias, a fim de torná-las “bancos de dados de propósito geral”, quebrando silos originais de escopos.

Uma das tecnologias que melhor se desenvolveu nessa linha foi o banco de dados orientado a documentos MongoDB.

# Bancos de Dados Não Relacionais

A ampli

O MongoDB, que é publicado via uma combinação de Licença Apache e uma licença GNU, é desenvolvido pela empresa privada homônima, MongoDB Inc. Além do desenvolvimento, correções e manutenções na ferramenta, a MongoDB Inc. presta também serviços de suporte comerciais para seus clientes, o que a torna uma solução competitiva no mercado.

Hoje, o MongoDB é o quinto sistema gerenciador de bancos de dados mais popular do mundo, e o primeiro quando analisamos apenas bancos de dados não relacionais, como podemos ver na imagem a seguir.

Figura 1 | Ranking de uso dos bancos de dados

Classificação			SGBD	Modelo de banco de dados	Pontuação		
julho de 2022	Junho de 2022	julho de 2021			julho de 2022	Junho de 2022	julho de 2021
1.	1.	1.	Oráculo	Relacional , Multimodelo	1280,30	-7,44	+17,63
2.	2.	2.	MySQL	Relacional , Multimodelo	1194,87	+5,66	-33,51
3.	3.	3.	Microsoft SQL Server	Relacional , Multimodelo	942,13	+8,30	-39,83
4.	4.	4.	PostgreSQL	Relacional , Multimodelo	615,87	-4,97	+38,72
5.	5.	5.	MongoDB	Documento , Multimodelo	472,98	-7,74	-23,18
6.	6.	6.	Redis	Valor-chave , vários modelos	173,62	-1,69	+5,32
7.	7.	7.	IBM DB2	Relacional , Multimodelo	161,22	+2,03	-3,94
8.	8.	8.	Elasticsearch	Motor de busca , multimodelo	154,33	-1,67	-1,43
9.	9.	↑11.	Microsoft Access	Relacional	145,09	+3,27	+31,64
10.	10.	↓9.	SQLite	Relacional	136,68	+1,24	+6,47
11.	11.	↓10.	Cassandra	Coluna larga	114,40	-1,05	+0,40
12.	12.	12.	MariaDB	Relacional , Multimodelo	112,52	+0,94	+14,54
13.	13.	↑25.	Floco de neve	Relacional	99,15	+2,73	+59,11
14.	14.	↓13.	Splunk	Motor de busca	98,21	+2,64	+8,15
Banco de Dados SQL do Microsoft Azure				Relacional , Multimodelo	84,89	-1,12	+9,68
15.	15.	15.	Microsoft Azure				

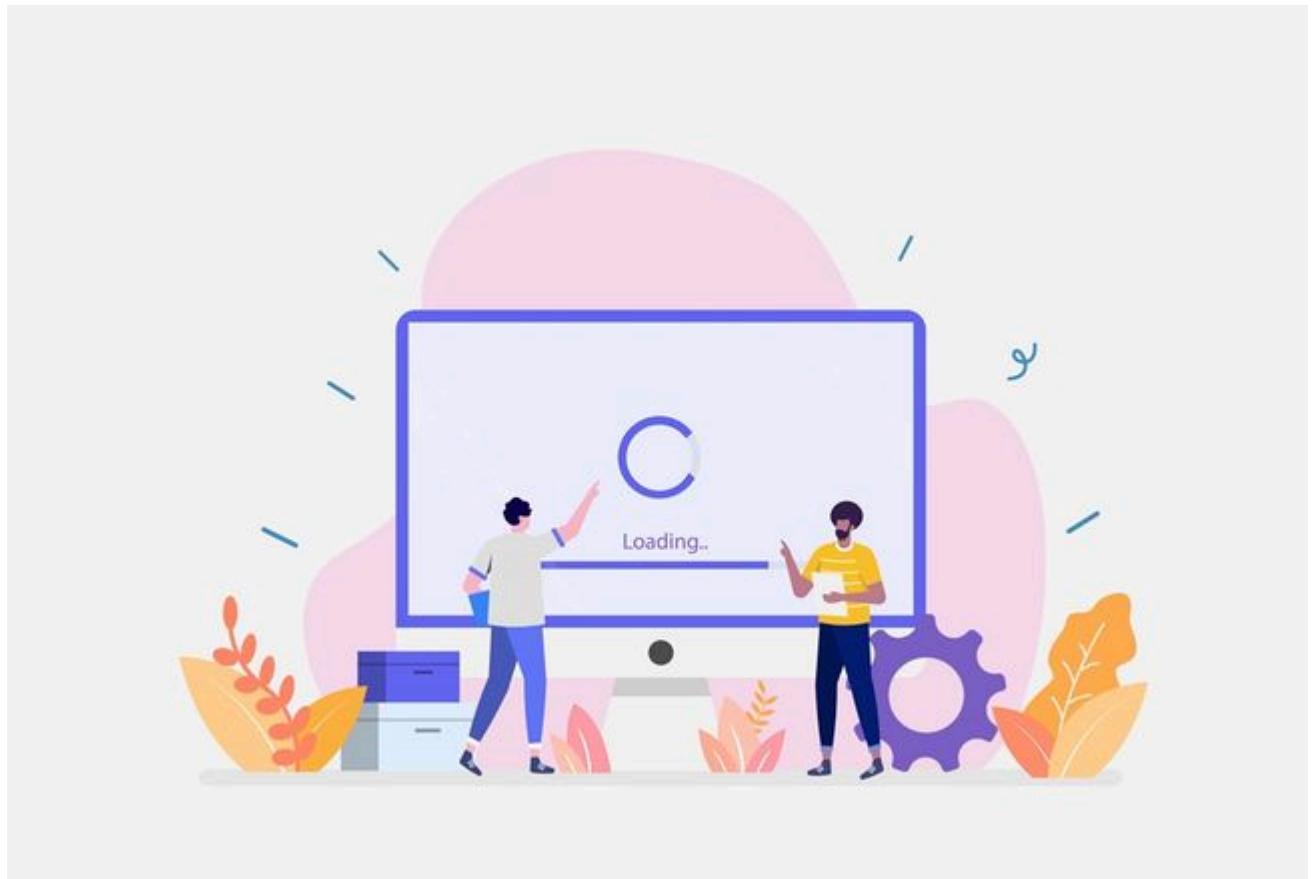
Fonte: DB-ENGINES.

O MongoDB foi lançado no início de 2009 e, nessa linha de buscar aperfeiçoar as capacidades dos SGBDs (Sistemas Gerenciadores de Bancos de Dados) não relacionais para se tornarem mais robustos, foi lançada em 2018 uma versão com as transações ACID.

No MongoDB, as transações que ocorrem em um único documento são atômicas, o que elimina algumas complexidades de projeto. Além disso, para situações em que se exijam atomicidade de leitura e escrita em múltiplos documentos, o MongoDB fornece suporte ao que chama de *multi-document transactions* (transações em múltiplos documentos, em tradução livre). Com esta propriedade, permite-se que os sistemas que utilizem o MongoDB como base de dados possam executar operações atômicas em múltiplos documentos, na linha do “tudo ou nada”, como vimos na característica de Atomicidade.

Para o uso da ferramenta na criação de aplicações que demandem dessas funcionalidades de transações atômicas em múltiplos documentos, a MongoDB Inc. recomenda o uso do MongoDB na versão 4.2 ou mais recente.

## Transações no MongoDB - transação completa



Como vimos na seção anterior, para cenários que tenham como requisito a atomicidade em operações de leitura e escrita em múltiplos documentos, seja em uma ou múltiplas *collections*, podemos utilizar as *multi-document transactions* (transações em múltiplos documentos, em tradução livre).

Por garantir a atomicidade, temos que:

- Quando uma transação é confirmada, todas as alterações realizadas nos dados são armazenadas e visíveis para leitura. Ou seja, não corremos o risco de que parte das alterações seja realizada enquanto outras são desfeitas por falha na execução da transação. Até que a transação seja confirmada integralmente, nenhuma das alterações parciais dos dados, realizadas pela transação, poderá ser visível ou acessível fora do escopo da transação.
- Quando uma transação é abortada por uma falha de operação, indisponibilidade de infraestrutura ou mesmo por possíveis interrupções forçadas, todas as alterações parciais por ela realizadas são descartadas e não sensibilizam os dados iniciais, não sendo essas alterações parciais visíveis em momento algum. Se, em uma dada transação, qualquer operação parcial gerar uma falha, por exemplo, todas as demais alterações nos dados

feitas pelas outras operações serão descartadas, sem nunca terem sido visíveis fora deste escopo da transação.

Vale ressaltar que, via de regra, para operacionalizar essa funcionalidade de transações atômicas em múltiplos documentos, é gerado um aumento do custo computacional para realizar a orquestração e a gestão do processo. Este aumento de custos computacionais é também conhecido como *overhead* (sobrecarga, em tradução livre). Com isso, é esperado que tenhamos um desempenho computacional inferior ao que temos no MongoDB sem o uso dessa funcionalidade. Esse desempenho inferior pode ser traduzido em um aumento de infraestrutura necessária e/ou em um aumento no tempo estimado para executar as atividades. Portanto, não devemos usar essa funcionalidade apenas para fugir da tarefa de fazer uma boa modelagem de dados, pois, para diversos cenários, com uma eficiente e adequada modelagem podemos reduzir a necessidade de transações atômicas em múltiplos documentos.

As transações atômicas em múltiplos documentos podem ser usadas em várias operações, *collections*, bases de dados e documentos. Baseando-nos na própria documentação da MongoDB, para transações, é possível:

- Especificar operações de leitura e escrita em *collections* existentes.
- Criar *collections* e índices em uma transação.
- Que as *collections* usadas em uma determinada transação estejam em diferentes bases de dados.

Por outro lado, não é possível:

- Fazer escrita em *capped collections* (um tipo específico de *collection* que busca maior performance de escrita).
- Ler ou escrever nas *collections* de config, admin, ou *local databases*.
- Escrever em *collections* system.\*, que são reservadas para uso interno do MongoDB, como para atribuição de papéis, usuários e suas credenciais, versão para suporte interno e construções de índices (MONGODB, 2022).

Além de outras características mais específicas, que você pode conhecer nas referências bibliográficas, se julgar interessante.

## Exemplos de transação



Nas transações atômicas para múltiplos documentos, podemos agrupar diversas instruções dentro da mesma transação. Com isso, podemos criar uma transação atômica para múltiplos documentos que contenham *updates* (atualizações), *inserts* (inserções) e *deletes* (remoções) unidos, de maneira atômica.

As possibilidades geradas por essa funcionalidade têm aplicações em múltiplos casos de uso do mercado e da academia de ensino.

Um exemplo de situação em que temos benefícios com estas transações atômicas para múltiplos documentos seria em movimentações financeiras.

Imagine que você dispõe de uma conta bancária no Banco do Aluno, assim como seu amigo Josué. Durante uma viagem que vocês fizeram nas férias de seus empregos, o Josué pagou por toda a hospedagem, pois teria desconto por um programa de benefícios de que ele é membro. Apesar de o Josué ter efetuado o pagamento da hospedagem, vocês dividirão o valor que ficou em R\$ 350,00 para cada pessoa. E aqui entram as transações do MongoDB. Quando você fizer a transferência para o Josué, o Banco do Aluno precisará executar algumas operações:

- Checar se sua conta dispõe de saldo.
- Checar se sua conta tem alguma restrição de movimentação.
- Confirmar a existência da conta do Josué.
- Debitar R\$ 350,00 de sua conta.
- Creditar R\$ 350,00 na conta do Josué.

Imagine que o débito de R\$ 350,00 tenha sido realizado com sucesso, mas o crédito na conta do Josué apresente falha. Com o uso das transações, que são atômicas, por não ter conseguido realizar todas as instruções, todo o processo será desfeito e sua conta continuará intacta, sem que o débito de R\$ 350,00 seja perceptível em sua conta. Caso não tivéssemos essa atomicidade, você teria os R\$ 350,00 retirados de sua conta e o Josué não teria o crédito feito, com o dinheiro permanecendo “no limbo”.

Outro exemplo em que as transações atômicas são úteis é no controle de estoque. Para fazer uma compra em um sistema de *e-commerce*, o sistema fará uma reserva do item comprado e:

- Reduzirá os itens comprados do estoque.
- Processará o pagamento.
- Aguardará confirmação da operadora de que o pagamento foi realizado com sucesso.
- Acionará o operador logístico para realizar a coleta dos produtos a serem entregues.

Imagine o cenário onde o pagamento seja recusado. Nesse caso, precisamos desfazer a redução de estoque que foi realizada. Utilizando transações atômicas, este problema é automaticamente resolvido, pois um item só será sensibilizado de fato na base, caso todos os demais também tenham sido executados com sucesso.

Segue um exemplo de código em JavaScript, retirado do portal da Microsoft, exemplificando uma transação atômica no MongoDB:

Exemplo 1 | Executando uma transação atômica com duas atualizações de registros

```
// insert data into friends collection
db.getMongo().getDB("users").friends.insert({name:"Tom"})
db.getMongo().getDB("users").friends.insert({name:"Mike"})
// start transaction
var session = db.getMongo().startSession();
var friendsCollection = session.getDatabase("users").friends;
session.startTransaction();
// operations in transaction
try {
    friendsCollection.updateOne({ name: "Tom" }, { $set: {
        friendOf: "Mike" } });
    friendsCollection.updateOne({ name: "Mike" }, { $set: {
        friendOf: "Tom" } });
} catch (error) {
    // abort transaction on error
    session.abortTransaction();
    throw error;
}

// commit transaction
session.commitTransaction();
```

Fonte: Microsoft (2022).

Reparam que esses casos são muito cotidianos no nosso mercado - transferências financeiras, compras em e-commerces - e que todos eles demandam confiança e robustez na sua execução.

# Bancos de Dados Não Relacionais

A ampli

Se não tivéssemos as garantias ACID no MongoDB, com suas transações, dificilmente poderíamos operar esses casos de uso com a tecnologia, pois abriríamos margem para grandes prejuízos financeiros.

## Videoaula: Transações no MongoDB



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta aula falaremos sobre como algumas tecnologias de bancos de dados NoSQL evoluíram para suportar as propriedades ACID, ampliando seu escopo de casos de uso. Para isso, aprenderemos como funcionam as transações atômicas em múltiplos documentos no banco de dados MongoDB. Trataremos também dos contras que o uso desta funcionalidade gera no desempenho de nossas aplicações.

## Saiba mais



Para conhecer um pouco mais sobre o MongoDB e suas transações, é pertinente conhecer a documentação da ferramenta:

## ["Transactions"](#)

(este conteúdo está em inglês, se preferir utilize o recurso de tradução do seu *browser* para estudá-lo)

---

## Referências



DB-ENGINES. DB-Engines Ranking **DB-Engines**, 2022. Disponível em <https://db-engines.com/en/ranking>. Acesso em 01 jul. 2022.

MONGODB. System Collections **MongoDB**, 2022. Disponível em <https://www.mongodb.com/docs/manual/reference/system-collections/>. Acesso em 08 jul. 2022.

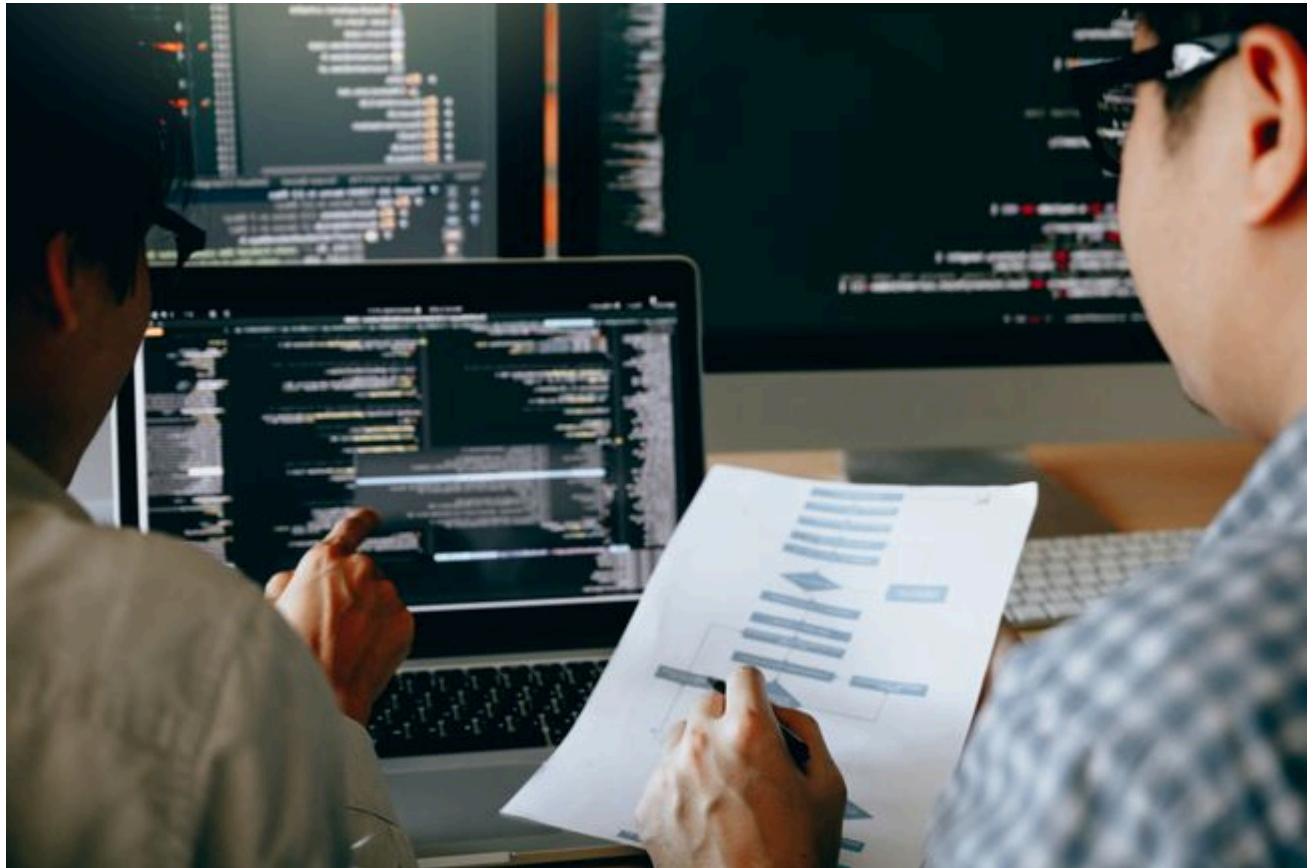
MICROSOFT. Usar as transações com vários documentos na API do Azure Cosmos DB para MongoDB **Microsoft**, 2022. Disponível em <https://docs.microsoft.com/pt-br/azure/cosmos-db/mongodb/use-multi-document-transactions>. Acesso em 01 jul. 2022.

## Aula 5

Revisão da unidade

**MapReduce e transactions em ambiente NoSQL**

# Bancos de Dados Não Relacionais



Nesta unidade você pôde ampliar os conhecimentos que vêm sendo desenvolvidos na disciplina, sobre bancos de dados não relacionais, e ainda se aprofundar em novos conteúdos como o MapReduce.

Vimos juntos que o MapReduce é uma ferramenta, contida no Apache Hadoop, que tem como princípio trabalhar com o processamento de grandes bases de dados em paralelo. Ancorado nos fundamentos da Computação Distribuída, o Hadoop surgiu na empresa Yahoo com a criação dos componentes MapReduce e HDFS (*Hadoop Distributed File System*), baseados em duas publicações científicas realizadas por cientistas da Google. O HDFS criou uma estrutura de armazenamento e organização de sistemas de arquivos em clusters de computação distribuída; já o MapReduce se utiliza do paradigma “dividir para conquistar”, quebrando os problemas a serem processados em subproblemas, em um processo chamado de Map. Com os mapeamentos executados, é realizado o processamento e são agregados os resultados preliminares até chegarmos ao resultado final. Este processo de agregação dos resultados é chamado de Reduce.

Estas tecnologias e abordagens de computação distribuídas se tornam importantes com o nosso contexto atual, onde a empresa em que você trabalha pode lidar com uma quantidade enorme de dados e que precisam ser armazenados e processados de maneira rápida e, preferencialmente, barata.

Após estudar sobre o MapReduce, você aprendeu sobre os bancos de dados e as diferenças entre os SGBDs (Sistemas Gerenciadores de Bases de Dados) relacionais, com suas estruturas

# Bancos de Dados Não Relacionais

de tabela e relacionamentos, e os bancos de dados ditos não relacionais, ou NoSQL. Entendemos que NoSQL não se refere a apenas um tipo de banco de dados, mas a uma ampla gama de tecnologias que podem ser agrupadas em 4 grandes grupos: bancos de dados orientados a documentos, bancos de dados orientados a grafos, bancos de dados colunares e bancos de dados chave-valor. Você viu que cada uma dessas categorias tem suas especificidades próprias e tem aderência facilitada a determinados contextos, como redes sociais (bancos orientados a grafos) ou cache de sistemas (bancos chave-valor).

Você viu que os bancos de dados NoSQL costumam ter, como uma das principais vantagens, um desempenho mais rápido dos que os bancos de dados relacionais (como MySQL, MS SQL Server e PostgreSQL), mas que, em contrapartida, não necessariamente atendem às propriedades ACID, que visam a garantir Atomicidade, Consistência, Isolamento e Durabilidade para as transações, aumentando nossa confiança para o uso em sistemas transacionais - como controle de vendas, estoque e movimentações financeiras.

Apesar de NoSQL não necessariamente dispor das propriedades ACID, você viu que algumas tecnologias, focadas em expandir a ferramenta para o uso de propósito geral, foram atrás de criar mecanismos para trabalhar e garantir sua aderência às propriedades. A principal tecnologia de banco de dados usada hoje (sendo a quinta maior ferramenta de banco de dados, e a maior NoSQL) é a MongoDB, um banco de dados não relacional orientado a documentos. Sobre isto, você aprendeu sobre as transações multi documentos atômicos do MongoDB e seu funcionamento, além de ver exemplos de código e aplicações.

Com estes conhecimentos que você desenvolveu, consegue ter um leque de ferramentas para trabalhar com cenários de armazenamento e processamento de grandes cargas de dados em suas empresas e projetos pessoais ou acadêmicos.

## Videoaula: Revisão da unidade



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta aula de revisão da unidade você relembrará os itens que aprendemos nas últimas aulas. Passaremos pelo MapReduce e seu framework tradicional, o Hadoop MapReduce, entendendo a aplicação, seus casos de uso e por que ele é uma ferramenta tão importante no nosso dia-a-dia profissional. Depois, você verá sobre as diferenças dos bancos relacionais e os bancos NoSQL, discutindo sobre suas características, aderência às propriedades ACID e, por fim, as transações atômicas em múltiplos documentos no MongoDB.

## Estudo de caso



Para consolidar o seu aprendizado, faremos um estudo de caso.

Vamos imaginar que você trabalha na empresa Livro de Rostos. A Livro de Rostos é um grande conglomerado de empresas de tecnologias, com foco principal na internet.

Algumas das empresas da Livro de Rostos são a Ostragrama, uma rede social focada no compartilhamento de fotos e vídeos de curta duração, e a Pantanal, uma empresa de comércio eletrônico que também atua como um *marketplace* (ou seja, ela vende seus próprios produtos e disponibiliza o canal para que parceiros possam vender seus produtos).

Você, como o responsável pela Arquitetura de Sistemas da Livro de Rostos, foi convocado pelo superintendente para projetar as estruturas de armazenamento e processamento dos dados das empresas do conglomerado.

Cada empresa tem seus próprios contextos e especificidades. Como focaremos na Ostragrama e na Pantanal, o superintendente lhe trouxe uma lista de requisitos que devem ser atendidos na sua proposta. Os principais requisitos, são:

**Para a Ostragrama:**

1. A. O volume de usuários a ser suportado é de cerca de 2 bilhões de contas, ao redor do mundo.
2. B. Um usuário A pode seguir um usuário B.
3. C. O fato de um usuário A seguir o usuário B não implica, necessariamente, que o usuário B também siga o usuário A.
4. D. Quando entrar no menu de **seguidores** do usuário A, o sistema deve retornar em até 1 segundo a lista integral de seguidores do usuário A.

5. Quando entrar no menu de usuários **seguidos** pelo usuário A, o sistema deve retornar em até 1 segundo a lista integral de usuários seguidos pelo usuário A.
6. F. O sistema deve dispor da funcionalidade de monitoramento de tendências.
7. G. A funcionalidade de monitoramento de tendências consiste em: receber uma lista de usuários e um intervalo de 2 datas. A funcionalidade processará todas as publicações e comentários textuais das contas listadas no período e retornará, de maneira ordenada decrescente, a lista de palavras utilizadas seguidas pela quantidade de ocorrências somadas de todas as contas no período.

#### Para a Pantanal:

1. A. É necessário que o sistema faça controle de cache das sessões ativas no e-commerce.
2. B. É necessário um banco de dados para as informações cadastrais.
3. C. Cada cadastro pode ter quantos números de telefone, e-mail e endereços de entrega quanto o usuário quiser, sendo como restrição apenas um número maior que 1 e menor do que 1024.
4. O sistema precisa controlar as vendas: ao ser realizada uma venda, o estoque deve ser decrescido, o faturamento incrementado pelo valor do produto e o sistema de logístico acionado para a entrega dos produtos físicos. Caso em algum momento da transação haja uma falha, a transação deve ser desfeita e nenhuma base deve ser impactada (ou seja, deve-se manter os estados dos dados excluindo as mudanças que seriam ocasionadas com a transação).

Buscando seu conhecimento técnico, o superintendente pergunta quais tecnologias devem ser usadas na Ostragrama e na Pantanal. Além disso, ele solicita uma breve descrição racional das escolhas efetuadas.

#### Refletá

Neste estudo de caso, você deve levar em conta diversos aspectos:

- Existe alguma tecnologia que tem aderência com cada contexto apresentado?
- A tecnologia atende aos requisitos de tempo de processamento?
- A tecnologia garante o bom funcionamento da operação?

## Videoaula: Resolução do estudo de caso



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

No nosso estudo de caso, você encontrou um cenário prático que traz uma série de requisitos comuns no dia-a-dia profissional.

Vamos começar com a cobertura dos requisitos da empresa Ostragrama.

De acordo com o item A, teremos que trabalhar com uma estrutura distribuída e que tenha alta escala para lidar com os dados. É possível trabalhar com esse volume em bancos de dados relacionais, mas já é um indício de que NoSQL pode vir a ser uma solução interessante.

Pelos itens B e C, vemos que existe um relacionamento entre os usuários, e o relacionamento é direcionado (ou seja, A segue B é diferente de B segue A). Os itens D e E nos mostram que existe uma grande demanda de desempenho quanto às consultas dos relacionamentos desses usuários. Estes itens - B, C, D e E - nos mostram que uma solução adequada é a de se fazer uso de bancos de dados orientados a grafos.

Os itens F e G nos mostram que iremos fazer um grande processamento massivo de dados.

Como você viu nas aulas da Unidade, o cenário encaixa muito bem com o uso do MapReduce, da Apache Hadoop. O *framework* é nativamente preparado para lidar com o processamento distribuído em um *cluster*, além de dispor de uma abordagem de resolução de problemas - o MapReduce -, que lida muito bem com esse processamento - ainda mais com a tarefa de contagem de palavras.

Com todos os itens mapeados acima, no que se refere ao Ostragrama, podemos indicar então o Neo4J como banco de dados não relacional, orientado a grafo, para fazer a persistência dos dados e lidar com as consultas de relacionamento em rede. Para os processamentos de monitoramento de tendências, o recomendado é o uso do MapReduce.

Já na empresa Pantanal, para o item A, podemos utilizar um banco de dados chave-valor, por ser simples, leve e atender muito bem esse cenário de armazenamento de cache.

Com os itens B e C, vemos que, para os cadastros, não temos uma estrutura tão rígida de campos, dado que podemos ter um número variável (e com grande discrepância) de números de telefone, endereços de entrega e e-mails para cada usuário. Até conseguimos mapear isto numa estrutura relacional, mas essa característica tem uma grande aderência com bancos de dados em documento.

Para o item D, precisamos garantir as propriedades ACID. Por ser um sistema transacional, todos estes itens - Atomicidade, Consistência, Isolamento e Durabilidade - se fazem necessários. A ausência de consistência e isolamento comprometem a qualidade dos dados durante as transações, além de ausência da atomicidade dificultar o desfazimento de uma transação falha, podendo gerar sérios prejuízos, tanto estruturais de dados quanto financeiros, para a Pantanal. Dados os itens B, C e D, podemos recomendar o uso de MongoDB para os armazenamentos de dados cadastrais e transacionais no controle das vendas. Já com o item A podemos utilizar o Redis.

## Resumo visual

# Bancos de Dados Não Relacionais



Para auxiliá-lo na escolha de qual tecnologia de banco de dados (relacional e NoSQL) utilizar, montei um breve fluxograma com algumas perguntas que podem nortear qual tecnologia usar. A ideia não é necessariamente inferir com 100% de precisão (afinal, existem casos em que mais de uma técnica é aderente ao problema) mas, sim, nortear sua escolha e fornecer uma opção inicial de como abordar o problema.

# Bancos de Dados Não Relacionais

A ampli

Figura 1 | Fluxograma



Fonte: elaborada pelo autor.

## Referências



VISUAL CAPITALIST. From Amazon to Zoom: What Happens in an Internet Minute In 2021? **Visual Capitalist**, 2021. Disponível em <https://www.visualcapitalist.com/from-amazon-to-zoom-what-happens-in-an-internet-minute-in-2021/>. Acesso em 18 jun. 2022.

APACHE HADOOP. Hadoop: Setting up a Single Node Cluster. **Apache Hadoop**. Disponível em: [https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone\\_Operation](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone_Operation). Acesso em: 19 jun. 2022

## Unidade 4

### Migração de Banco de Dados Relacional para Não Relacional

#### Aula 1

Conceitos de bancos de dados relacionais e não relacionais

#### Introdução

# Bancos de Dados Não Relacionais

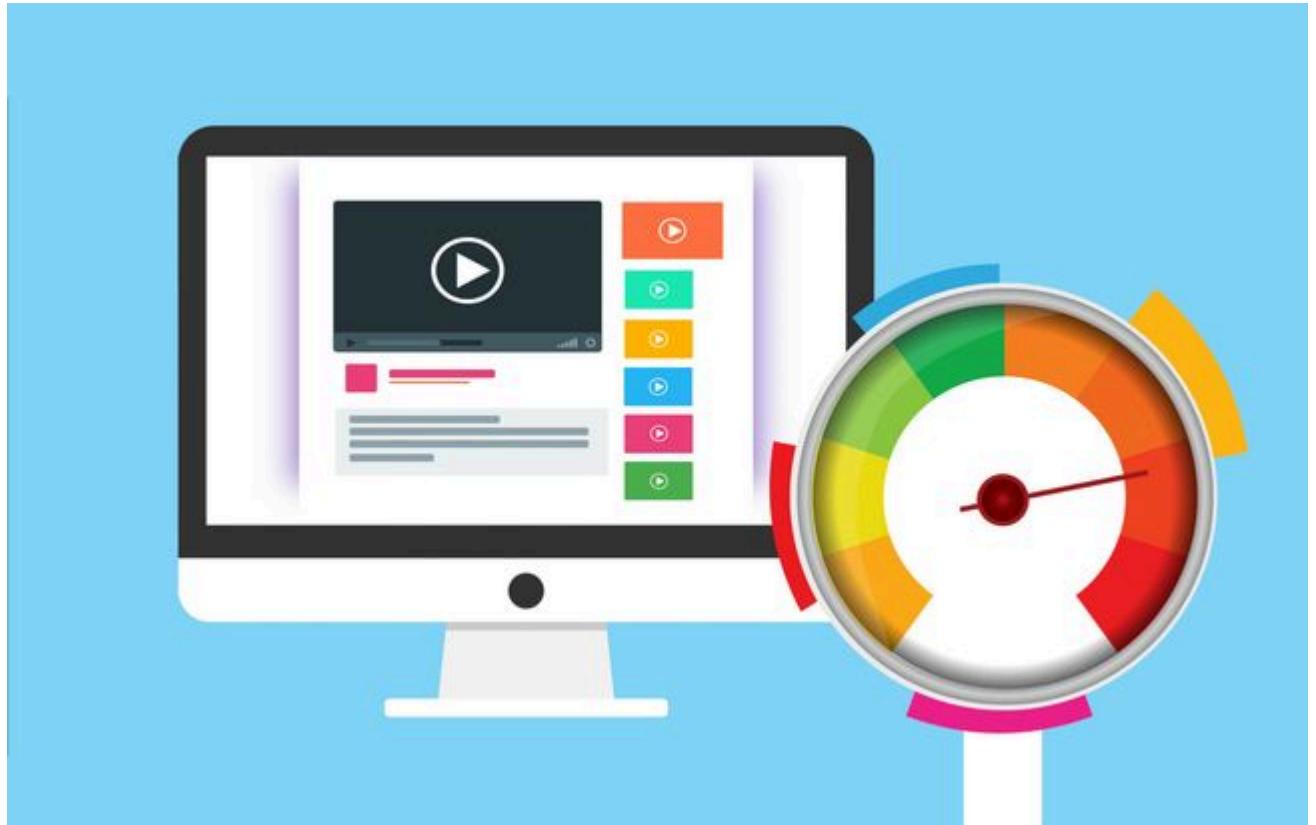


Começaremos nossos estudos conceituando e diferenciando bases de dados relacionais e NoSQL. Veremos que, além de diferenças em suas estruturas, também temos atualmente diversos tipos de bancos de dados disponíveis.

Nos sistemas de gerenciamento de **banco de dados relacionais**, veremos que os dados são armazenados de forma estruturada, porém possuem limitações como a escalabilidade. Já o **NoSQL** trabalha com dados não estruturados e sem esquema e, desta forma, podem ser armazenados em várias coleções e nós.

A transferência de dados, por fim, é o processo no qual os dados são transferidos entre sistemas. Diversos motivos, como problemas de custo, melhores recursos e serviços rápidos, podem fazer com que uma empresa transfira dados de uma plataforma para outra.

## Conceitos fundamentais sobre bases de dados relacionais e NoSQL



As organizações, de um modo geral, utilizam um volume grande de ativos, por exemplo, imóveis. No entanto, talvez o ativo mais valioso seja a informação, os dados. Logo, os dados são a espinha dorsal de todos os projetos que envolvem tecnologia.

Geralmente temos os dados armazenados em um Sistema de Gerenciamento de Banco de Dados (SGBD), cuja interação e comunicação com o SGBD, é realizada a partir do uso de uma determinada linguagem que ele entenda, desse modo foi criada uma linguagem que é utilizada para interagir com os sistemas de banco de dados, que chamamos de *Structured Query Language* (SQL) (METWALLI, 2022).

Porém, nas últimas décadas, tanto o volume como a velocidade dos dados cresceram exponencialmente, e tivemos como consequência, uma grande diversidade de tecnologias de banco de dados que passaram a serem desenvolvidas a fim de atender a grande demanda do mercado. Esses bancos de dados possuem normalmente implementações distribuídas, pois o volume de dados que são gerenciados excede muito a capacidade de armazenamento de um único nó (PLOETZ *et al.*, 2018). E, assim, surgiu o *banco de dados não relacional*. E o NoSQL é a linguagem utilizada para interagir com bancos de dados não relacionais.

Como mencionado, hoje possuímos diversos tipos de bancos de dados disponíveis, alguns com mais popularidade. No SQL, temos: MySQL, Oracle, Microsoft SQL Server e o PostgreSQL. Já no NoSQL, temos o MongoDB e Cassandra e muitos outros, porém menos populares no mercado (PLOETZ *et al.*, 2018).

Esses bancos de dados são amplamente implantados nas organizações, e possuem diferentes modelos de dados, que vão desde o modelo de documento do MongoDB, ou modelo coluna-família do Cassandra.

## Consistência *versus* disponibilidade

A **consistência** nos estudos de um banco de dados diz respeito à confiabilidade do desempenho de suas funções. Logo, um sistema dito consistente realiza determinadas leituras obtendo como retorno o valor da última gravação e as leituras em uma determinada época retornam o valor idêntico, independentemente de onde foram iniciadas (PLOETZ et al., 2018).

A **disponibilidade** de um banco de dados está relacionada à capacidade do sistema de concluir uma operação. A disponibilidade trata-se de um espectro, que nada mais é um sistema que pode apresentar-se indisponível para gravações enquanto ele estiver disponível para leituras; como também pode apresentar-se indisponível para realizar operações administrativas enquanto estiver disponível para operacionalizar os dados (PLOETZ et al., 2018).

Mas é importante saber que consistência e disponibilidade podem ser conflituosas, no sentido de que um sistema quanto for altamente disponível precisará permitir que as operações sejam bem-sucedidas, mesmo se alguns nós do sistema estiverem inabilitados. Contudo, como não conhecemos se os nós ainda estão de fato acessíveis, não há então qualquer garantia se as operações deixaram o sistema em um estado consistente. Desse modo, um sistema consistente deverá certificar que todos os nós tenham dados para que uma chave específica seja não só alcançável, mas como também estejam na operação (PLOETZ et al., 2018).

## Análise comparativa entre SBGDS relacionais e NoSQL



O *Relational Database Management System (RDBMS)* é um banco de dados relacional. É a forma padrão para sistemas de gerenciamento de banco de dados relacional (em português, SGBDs).

# Bancos de Dados Não Relacionais

Os dados são armazenados na forma de linhas e colunas, e suas principais características englobam:

1. Bancos de dados SQL - baseados em tabelas.
1. Armazenamento de dados em linhas e colunas.
1. Uma instância exclusiva de dados para as categorias definidas pelas colunas contida em cada linha.
1. Chave primária do recurso, para identificar exclusivamente as linhas.

As principais limitações do banco de dados relacionais são:

- **Escalabilidade:** obter a escalabilidade tornou-se atualmente um grande desafio para os bancos de dados relacionais, isso porque estes foram pensados em um período em que os dados eram armazenados em pequenos tamanhos e projetados para serem executados em um único servidor, para que pudesse manter sua integridade como também evitar os possíveis problemas da computação distribuída. Com esse modelo, caso um sistema necessite ser redimensionado, será necessário adquirir um hardware com maior poder de processamento, porém com mais complexidade, o que se tornará um processo oneroso de se manter (ALLEN, 2022).
- **Complexidade:** No banco de dados tipo SQL, os dados devem necessariamente caber em tabelas. Se seus dados não couberem em tabelas, você precisará projetar sua estrutura de banco de dados, que será complexa e novamente difícil de manusear.

No **NoSQL**, comumente referido como "**Not Only SQL**", os dados não estruturados e sem esquema podem ser armazenados em várias coleções e nós. Segundo o portal LoginRadius (2022), seus principais benefícios são:

- **Altamente e facilmente escalável**

Bancos de dados relacionais são escaláveis, e projetados para que forneçam dados em um formato devidamente estruturado. No momento em que temos uma carga aumentando no banco de dados relacional, podemos redimensionar este mesmo banco de dados acrescentando potência ao hardware do servidor, mas para isso é necessário termos servidores maiores, porém eles são mais caros. Já os bancos de dados não relacionais, são elaborados de modo que sejam expandidos em escala horizontal, o que significa dizer que podemos dimensioná-los adicionando mais máquinas em seu *pool* de recursos (LOGINRADIUS, 2022).

- **Servidores NoSQL são mais baratos**

Os bancos de dados não relacionais (NoSQL) exigem um custo menor no seu gerenciamento se comparado aos sistemas relacionais. Eles também suportam inúmeros recursos que vão desde o reparo automático, facilidade na distribuição de dados e modelagens mais simples (LOGINRADIUS, 2022).

- Menor custo de servidor e código aberto

Os bancos de dados NoSQL possuem código aberto, o que barateia e muito. Sua implementação também é fácil, e utilizar também servidores populares para gerenciar os dados e transações, enquanto que os bancos de dados relacionais possuem licenças onerosas e utilizam servidores e sistemas de armazenamento muito grande, a depender do volume de dados (LOGINRADIUS, 2022).

- Não possui esquema ou modelo de dados fixos

Não é necessário apresentar um esquema predefinido para inserir os dados no banco de dados NoSQL, dessa forma o modelo de dados poderá sofrer alterações a todo instante (LOGINRADIUS, 2022).

- Suporte ao cache integrado

No banco de dados não relacional há o suporte de cache na memória do sistema, o que melhora o desempenho da saída dos dados. No banco de dados relacional isto é realizado utilizando uma infraestrutura a parte (LOGINRADIUS, 2022).

## Ferramentas para migração de banco de dados



A maioria das empresas transfere seus dados para plataformas mais eficientes para obter melhores recursos com o intuito de facilitar suas operações comerciais diárias. Motivos como problemas de custo, mais recursos e serviços rápidos podem fazer com que uma empresa transfira dados de uma plataforma para outra.

A migração de dados é o processo de transferência de dados entre sistemas, cujos sistemas podem ser do tipo armazenamento de dados ou formatos de arquivo. Na migração temos os dados de um sistema antigo sendo enviados para outro sistema mais novo através de um padrão de mapeamento específico. Estes padrões incluem designs que atuam como um tradutor dos dados antigos para o formato do novo sistema, o que garante uma migração mais segura (SOFTWARE TESTING HELP, 2022).

A migração de dados pode ser motivada por diversos fatores, sendo os mais comuns, como afirma o Software Testing Help (2022):

- Migração de aplicativos.
- Atividades de manutenção ou atualização.
- Substituições de equipamentos de armazenamento/servidor.
- Migração ou realocação do *data center*.
- Consolidação de sites.

A migração de dados para ser realizada manualmente demoraria muito tempo, para isso tivemos a automatização deste processo com o auxílio de ferramentas específicas para este propósito. A migração de dados programática vai compreender desde a extração de dados do sistema antigo, o seu carregamento para o novo sistema e verificação dos dados a fim de certificar que os dados foram de fato transferidos.

Podemos classificar em três os tipos de ferramentas de migração de dados, de acordo com Phaujdar (2021):

- **Localmente:** soluções in loco é a melhor opção quando os requisitos de conformidade proibirem as ferramentas de migração de dados multilocatários ou baseadas em nuvem. Esta ferramenta proporciona baixa latência e total controle desde do aplicativo até as camadas físicas.
- **Self-Scripted:** são boas soluções em projetos de pequeno porte ou quando necessário efetuar rápidas correções, e/ou quando um destino ou origem específica não é suportado por outras ferramentas. Essas ferramentas são baratas se os requisitos forem simples o suficiente. No entanto, as soluções com *scripts* automáticos exigem amplo conhecimento de codificação que pode desviar os engenheiros das tarefas mais urgentes. Se o código não estiver bem documentado, pode ser difícil fazer alterações.
- **Baseado em nuvem:** estas ferramentas são aplicadas quando for necessário aumentar e/ou reduzir a escala a fim de atender as exigências dos dados dinâmicos. Isso é ideal para analistas de negócios e cientistas de dados em diferentes locais que precisam de acesso a ferramentas comuns e *data warehouses*. Essas ferramentas são escaláveis e ágeis o suficiente para lidar com as necessidades de negócios em constante mudança. Isso é evidente por meio de seu armazenamento sob demanda e poder de computação, que pode lidar com um aumento na demanda causado por eventos temporários ou intermitentes.

## Videoaula: Conceitos de bancos de dados relacionais e não relacionais



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Uma estratégia abrangente de migração de dados evita uma experiência abaixo do ideal, que cria mais problemas do que resolve. Uma estratégia incompleta pode fazer com que um projeto de migração de dados falhe. A seleção de uma ferramenta adequada é uma parte importante dessa estratégia e deve ser baseada nos requisitos de negócios da organização. Veremos quais as principais ferramentas de migração no mercado e alguns fatores a se ter em mente antes de decidir qual delas utilizar.

### Saiba mais



Para entender melhor os bancos de dados não relacionais, seguem alguns conteúdos, que abordam com mais detalhes esse assunto. Vale a pena conferir:

["Vantagens de um Banco de Dados NoSQL, MongoDB"](#)

["Quando utilizar RDBMS ou NoSQL?"](#)

["SQL, NoSQL, NewSQL: Qual banco de dados usar?"](#)

["Comparar ferramentas de migração de dados do SQL"](#)

["Como transferir dados de um banco de dados para outro banco de dados"](#)

## Referências



ALLEN, M. Relational Databases Are Not Designed For Scale. **Mark Logic**, 2022. Disponível em: <https://www.marklogic.com/blog/relational-databases-scale/>. Acesso em: 06 jul. 2022.

LEAVITT, S. Comparar ferramentas de migração de dados do SQL. In: Microsoft. **Microsoft. [S.I.]**. Disponível em: <https://docs.microsoft.com/pt-br/sql/sql-server/migrate/dma-azure-migrate-compare-migration-tools?view=sql-server-ver16>. Acesso em: 20 jul. 2022.

LOGINRADIUS. RDBMS vs NoSQL. **LoginRadius**, 2022. Disponível em: <https://www.loginradius.com/blog/engineering/relational-database-management-system-rdbms-vs-nosql/#:~:text=NoSql%20database%20implementation%20is%20easy,than%20the%20cost%20of%20RDBMS>. Acesso em: 27 jun. 2022.

METWALLI, S. A. SQL vs. NoSQL: Which Should You Choose? **Builtin**, 2022. Disponível em: <https://builtin.com/data-science/sql-vs-nosql>. Acesso em: 27 jun. 2022.

MUNIZ, F. Vantagens de um Banco de Dados NoSQL, MongoDB. In:4-Linux. **4-Linux.** [S.I.]. 21 mar. 2018. Disponível em: <https://blog.4linux.com.br/vantagens-de-um-banco-de-dados-nosql-mongodb/>. Acesso em: 20 jul. 2022.

PEREIRA, T. Quando utilizar RDBMS ou NoSQL?. In:Data Science Academy. **Data Science Academy Blog.** [S.I.]. 19 mai. 2016. Disponível em: <https://blog.dsacademy.com.br/quando-utilizar-rdbms-ou-nosql/>. Acesso em: 20 jul. 2022.

PHAUJDAR, A. 11 Best Data Migration Tools for 2022. **Hevo**, 2021. Disponível em: <https://hevodata.com/learn/data-migration-tools/>. Acesso em: 27 jun. 2022.

PLOETZ, A.; KANDHARE, D.; KADAMBI, S.; WU, X. *Seven NoSQL Databases in a Week*. Birmingham: Packt, 2018. Disponível em: [https://www.packtpub.com/product/seven-nosql-databases-in-a-week/9781787288867?\\_ga=2.155181944.1968772252.1656419861-906940344.1655237593](https://www.packtpub.com/product/seven-nosql-databases-in-a-week/9781787288867?_ga=2.155181944.1968772252.1656419861-906940344.1655237593). Acesso em: 27 jun. 2022.

RAFAEL, B. SQL, NoSQL, NewSQL: Qual banco de dados usar?. In:Geek Hunter. **Geek Hunter.** [S.I.]. 24 out. 2019. Disponível em: <https://blog.geekhunter.com.br/sql-nosql-newsql-qual-banco-de-dados-usar/#NewSQL>. Acesso em: 20 jul. 2022.

SMALLCOMBE, M. SQL vs NoSQL: 5 Critical Differences. **Integrate.io**, 2021. Disponível em: <https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable,data%20like%20documents%20or%20JSON>. Acesso em: 27 jun. 2022.

SOFTWARE TESTING HELP. 13 Best Data Migration Tools For Complete Data Integrity [2022 LIST]. **Software Testing Help**, 2022. Disponível em: <https://www.softwaretestinghelp.com/data-migration-tools/>. Acesso em: 27 jun. 2022.

## Aula 2

Tipos de bases de dados

### Introdução



Vamos iniciar abordando os bancos de dados relacionais (SQL), sendo eles formados por agrupamentos de informações ou dados logicamente modelados, que apresentam um esquema formado por colunas e tabelas.

Em seguida, veremos que uma característica comum dos bancos de dados NoSQL é o seu armazenamento de valor-chave, que é um banco de dados em que as coleções são dicionários inerentes, nos quais cada entrada está associada a uma chave exclusiva da coleção. Embora seja semelhante ao banco de dados relacional, os armazenamentos de valores-chave diferem de maneira fundamental das tabelas relacionais, pois não dependem de um esquema fixo para uma determinada coleção.

Por último, conheceremos o teorema CAP, apresentado por Eric Brewer em 2000. Segundo ele,

“um registro de leitura/gravação sequencialmente consistente que eventualmente responde a todas as solicitações não pode ser realizado em um sistema assíncrono propenso a partições de rede” (GESSERT, 2016, [s. p.])

## Bases de dados relacionais



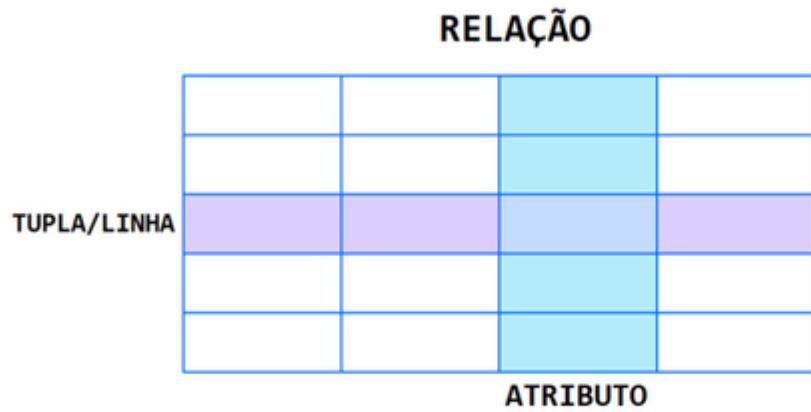
Os bancos de dados são agrupamentos de informações, e ou mesmo uma coleção de dados, independentemente de como ou onde está sendo armazenado. Um exemplo seriam os arquivos com as informações de folha de pagamento, ou mesmo os formulários de pacientes em hospitais. Antes de termos os computadores, tínhamos os bancos de dados físicos que eram os únicos armazenamentos de informações, e não eram nada seguros.

Com o passar dos anos, por volta do século 20, o advento da ciência da computação proporcionou computadores com grande capacidade de processamento e de armazenamento tanto local como externo. No fim dos anos 60, Edgar F. Codd apresentou seu modelo de gerenciamento de banco de dados relacional. Este permitia registros individuais em tabelas, abrindo diferentes possibilidades, como os relacionamentos ditos “muitos para muitos” entre pontos de dados, e relacionamentos “um para muitos”. Este modelo trouxe uma maior flexibilidade para os projetos de estruturas de banco de dados e os Sistemas de Gerenciamento de Banco de Dados (SGBDs) puderam atender diversa necessidades de negócios.

## Modelagem de dados

Vejamos mais de perto como um modelo pode organizar os dados. Temos que as relações são os elementos mais importantes no modelo relacional. Uma relação é um conjunto de tuplas (que são as linhas da tabela), que compartilham um conjunto de atributos ou colunas, e podem ter atributos como *name*, *subjects*, *start\_date*, etc., como demonstrado na Figura 1:

Figura 1 | Elementos do modelo relacional



Fonte: adaptada de Drake (2020).

A coluna é a menor estrutura organizacional em um banco de dados relacional e representa os diversos itens que definem os registros em uma tabela, por isso recebe um nome mais formal, “atributos”. Podemos pensar em cada tupla como uma instância única, e que podem ser coisas como funcionários de uma empresa, ou resultados de testes de laboratório.

### Diagramas Entidade-Relacionamento (ER)

Um diagrama ER

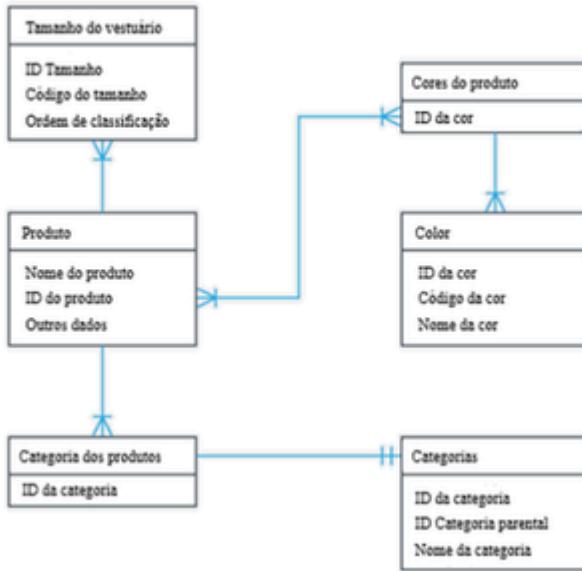
“mostra os relacionamentos de conjuntos de entidades armazenados em um banco de dados” (AJAY, 2020, [s. p.]),

como demonstrado na Figura 2.

# Bancos de Dados Não Relacionais

A ampli

Figura 2 | Diagrama ER para ilustrar a estrutura lógica de uma base de dados relacional



Fonte: adaptada de Ajay (2020)

**Entidades:** São representações do mundo real que podem ser desde um lugar a uma pessoa. São representados por uma caixa retangular.

**Atributos:** Representam as características da entidade por meio do símbolo de elipse, podendo ser nome, idade, entre outros.

**Relacionamentos:** Descreve a associação entre duas entidades, sendo representado pelo símbolo de diamante. Temos três tipos de relacionamentos: um a um; um para muitos e muitos para muitos, como demonstrados nos exemplos a seguir sob a notação de Peter Chen (HEUSER, 2009).

**Um para um:** “uma entidade está associada a outra entidade” (AJAY, 2020, [s. p.])

como representado na Figura 3.

# Bancos de Dados Não Relacionais

Figura 3 | Exemplo de diagrama um para um



**Legenda:** cada aluno está associado a um curso.

Fonte: adaptada de Ajay (2020).

**Um para muitos:** “uma entidade está associada a muitas outras entidades” (AJAY, 2020, [s. p.])

como vemos na Figura 4.

Figura 4 | Exemplo de diagrama um para muitos



**Legenda:** um curso que está associado/inscrito a todos os estudantes que cursam uma determinada área de ensino (TI por exemplo) em uma determinada universidade/instituição

Fonte: adaptada de Ajay (2020).

Fonte: adaptada de Ajay (2020).

**Muitos para Muitos:** “muitas entidades estão associadas a muitas outras entidades” (AJAY, 2020, [s. p.])

como na Figura 5.

# Bancos de Dados Não Relacionais

Figura 5 | Exemplo de diagrama um para muitos



**Legenda:** os estudantes que estão associados a vários cursos da área de tecnologia

Fonte: adaptada de Ajay (2020).

Fonte: adaptada de Ajay (2020).

Hoje podemos dizer que todos os principais provedores de nuvem agora oferecem serviços populares de banco de dados relacional gerenciado, por exemplo: Amazon RDS, Google Cloud SQL, Banco de dados do Azure para PostgreSQL.

## Bases de dados NoSQL

# Bancos de Dados Não Relacionais



Antes de sabermos as tendências atuais de pesquisa em linguagens e sistemas de banco de dados não relacional, é importante destacar alguns dos conceitos técnicos de tais sistemas. Na verdade, é o seu afastamento de traços relacionais bem compreendidos que fomentou novas pesquisas e desenvolvimentos nesta área. Os aspectos que vamos estudar estão principalmente centrados em modelos de dados.

Os bancos de dados NoSQL permitem que os desenvolvedores armazenem grandes quantidades de dados não estruturados, dando-lhes muita flexibilidade, eles se diferenciam porque fornecem um mecanismo para armazenar e recuperar dados não estruturados. Como os bancos de dados NoSQL foram projetados para resolver problemas de escalabilidade dos bancos de dados SQL, eles são livres de esquemas e baseados em sistemas distribuídos, tornando-os mais fáceis de dimensionar e fragmentar.

Uma característica que você verá entre os bancos de dados NoSQL é a tipologia do armazenamento chamado de **valor-chave**, este é um banco de dados cujas coleções são dicionários específicos nos quais cada entrada fica associada a uma chave restrita da coleção. Embora possa parecer com um banco de dados relacional, este tipo se diferencia no modelo das tabelas, pois não dependem de um esquema fixo para uma determinada coleção (PIVERT, 2018).

Outro ponto é o suporte nativo de seu pequeno conjunto de operações, porém não suportam operações de junção, mas depende da desnormalização de dados para alcançar resultados

semelhantes ao banco de dados relacional, porém geram custos de armazenamento e manutenção (PIVERT, 2018).

## Tipos de NoSQL

Existem quatro principais tipos de bancos de dados NoSQL, que são:

- **Armazenamentos de valor-chave:** é um banco de dados que usa chaves diferentes onde cada uma está associada a apenas um valor em uma coleção, não apresentam linguagem de consulta, e os comandos para armazenar, recuperar e deletar dados são respectivamente o GET, PUT e DELETE. Sua estrutura simplificada torna o armazenamento de valor-chave rápido, fácil de acessar, escalável e portátil (RASANAYAGAM, 2020).

Exemplos: Dínamo e Riak.

- **Gráfico orientado a colunas:** muito empregado no gerenciamento de *data warehouses*, em inteligência de negócios e detecção de fraudes. Como os dados estão disponíveis em uma coluna, esse tipo acaba oferecendo alto desempenho nas consultas de agregação (RASANAYAGAM, 2020).

Exemplo: HBase; Cassandra e Hipertabela.

- **Baseado em gráficos:** neste os dados são retratados semelhante a um gráfico de entidades onde cada nó do gráfico é um pedaço de dados. As arestas simbolizam a relação entre os nós. Cada aresta e nó tem seu identificador único. Este tipo é voltado para redes sociais, logística e dados espaciais (RASANAYAGAM, 2020).

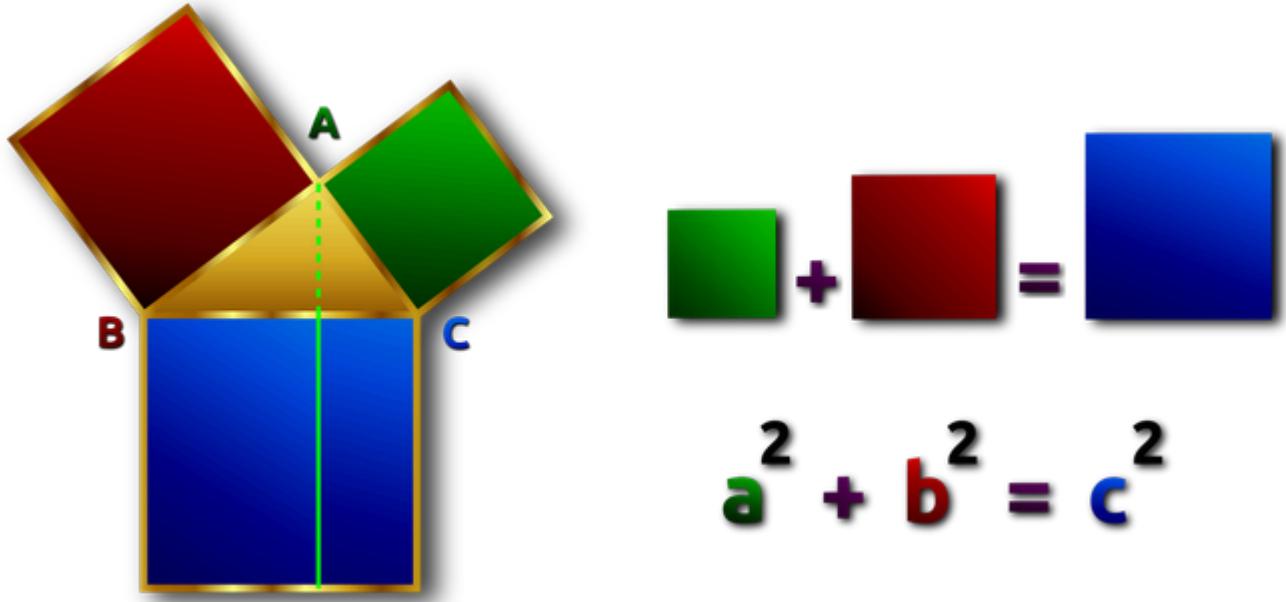
Exemplo: Neo4j; OrienteDB; FlockDB.

- **Orientado a documentos:** é bem aplicado para usos de análises em tempo real, como em e-commerce, sistema de gerenciamento de conteúdo. Não é indicado aplicar em transações complexas com várias operações ou consultas (RASANAYAGAM, 2020).

Exemplo: CouchDB; MongoDB e Riak.

## Teorema de CAP

# Bancos de Dados Não Relacionais



O Teorema CAP foi apresentado por Eric Brewer em 2000. É um dos resultados mais influentes no campo da computação distribuída, porque coloca um limite superior no que pode possivelmente ser realizado por um sistema distribuído. Ele afirma que este sistema que apresenta as propriedades Consistência, Disponibilidade e Tolerância pode garantir no máximo duas das três propriedades ao mesmo tempo (GESSERT, 2016).:

- **Consistência (C):** as leituras e gravações são sempre executadas atomicamente e são estritamente consistentes. Simplificadamente temos que todos os clientes possuem a mesma visão dos dados a todo o momento.
- **Disponibilidade (A):** cada nó com êxito no sistema poderá aceitar solicitações de leitura e escrita de clientes e possivelmente retornará com uma resposta sem mensagem de erro.
- **Tolerância à partição (P):** O sistema preserva as garantias de consistência e disponibilidade diante de alguma perda de mensagens entre os nós ou durante alguma falha parcial do sistema.

Brewer argumenta que um sistema pode estar disponível e consistente em operação normal, mas, na presença de uma partição do sistema, não é possível pois se houver a perda do contato com os outros nós deverá ter que decidir entre continuar processando solicitações a fim de preservar a disponibilidade de clientes ou rejeitar as solicitações dos clientes para sustentar as garantias de consistência (GESSERT, 2016). Importante observar que neste caso a primeira opção viola a consistência, porque conduz a leituras obsoletas e gravações conflitantes, enquanto que a segunda opção prejudica a disponibilidade (GESSERT, 2016).

Teoricamente a tolerância à partição é uma obrigatoriedade, porque supõe que o sistema opere em um armazenamento de dados distribuído, que por natureza, seja com partições de rede. Falhas de rede poderão acontecer, logo é importante oferecer qualquer tipo de serviço confiável, desse modo é indispensável a tolerância de partição – o P em CAP (JOHNSON, 2020). Porém, a

obrigatoriedade da partição, pede-se que opte também por garantir consistência ou disponibilidade, de modo que:

- A alta consistência tem o custo de menor disponibilidade.
- A alta disponibilidade tem o custo de menor consistência.

Cabe observar que a consistência no CAP significa ter as informações mais atualizadas o que difere da consistência do ACID que se refere a um evento de banco de dados diferente, no qual significa que qualquer nova transação no banco de dados não o corromperá (JOHNSON, 2020). Enquanto que para alguns fazer a escolha entre consistência e disponibilidade seja realmente uma questão filosófica, ela dificilmente é feita na prática. Onde temos que a teoria diz para você ter apenas dois dos três componentes, no dia a dia nem sempre será o caso.

Eric Brewer esclareceu algumas confusões em torno do teorema, resumindo-o em uma declaração:

*"O objetivo do CAP moderno deve ser maximizar as combinações de consistência e disponibilidade que façam sentido para a aplicação específica. Essa abordagem incorpora planos para operação durante uma partição e para recuperação posterior, ajudando os designers a pensar sobre o CAP além de suas limitações historicamente percebidas"* (JOHNSON, 2020).

Antes mesmo de trabalhar com o CAP e procurar realizar a seleção corretamente, cabe primeiro entender qual tipo de banco de dados usar, porque no caso dos bancos de dados não relacionais, tais escolhas podem ser classificadas com base no suporte a alta disponibilidade ou a alta consistência.

## Videoaula: Tipos de bases de dados



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

O armazenamento de dados é um grande negócio. As empresas de dados estão constantemente nos noticiários ultimamente, especialmente porque as empresas tentam maximizar o valor do potencial do Big Data. Para o leigo, o armazenamento de dados geralmente é tratado em um banco de dados tradicional. Mas para o Big Data, as empresas usam data warehouses e data lakes. Veremos neste bloco um pouco do armazenamento de dados explicando: Data Lake versus Warehouse versus Banco de Dados.

## Saiba mais



Para entender melhor os bancos de dados não relacionais, segue uma indicação de leitura extra.  
Vale a pena conferir:

["Teorema CAP"](#)

---

## Referências

# Bancos de Dados Não Relacionais



AJAY, M. S. Introduction to Databases and SQL. **Data Driven Investor**, 2020. Disponível em: <https://medium.datadriveninvestor.com/introduction-to-databases-and-sql-1c1ebcec596d>. Acesso em: 02 jul. 2020.

DRAKE, M. Understanding Relational Databases. **Digital Clean**, 2020. Disponível em: <https://www.digitalocean.com/community/tutorials/understanding-relational-databases>. Acesso em: 06 jul. 2022.

GESSSERT, F. NoSQL Databases: a Survey and Decision Guidance. **Medium**, 2016. Disponível em: <https://medium.baqend.com/nosql-databases-a-survey-and-decision-guidance-ea7823a822d>. Acesso em: 02 jul. 2022.

HEUSER, C. A. **Projeto de banco de dados**. 6 ed. Porto Alegre: Bookman, 2009. IBM EDUCATION CLOUD. **IBM**. O que é o Teorema CAP?. [S.I.]. IBM, 2019. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/cap-theorem>. Acesso em: 21 jul. 2022.

JOHNSON, J. CAP Theorem for Databases: Consistency, Availability & Partition Tolerance. **BMC**, 2020. Disponível em: <https://www.bmc.com/blogs/cap-theorem/>. Acesso em: 02 jul. 2022.

PIVERT, OI. **NoSQL Data Models**. New Jersey: Wiley, 2018. Disponível em: [https://www.packtpub.com/product/nosql-data-models/9781786303646?\\_ga=2.222124944.961170607.1656854447-906940344.1655237593](https://www.packtpub.com/product/nosql-data-models/9781786303646?_ga=2.222124944.961170607.1656854447-906940344.1655237593). Acesso em: 02 jul. 2022.

RAFAEL, B. SQL, NoSQL, NewSQL: Qual banco de dados usar?. In: Geek Hunter. **Geek Hunter**. [S.I.]. 24 out. 2019. Disponível em: <https://blog.geekhunter.com.br/sql-nosql-newsql-qual-banco-de-dados-usar/#NewSQL>. Acesso em: 20 jul. 2022.

RASANAYAGAM, J. NoSQL. Medium, 2020. Disponível em: <https://medium.com/@R.Sumangala/nosql-ab44a979a831>. Acesso em: 02 jul. 2022.

SHAFI, A. From Basic to Intermediate SQL in 10 Minutes. **Analytics Vidhya**, 2020. Disponível em: <https://medium.com/analytics-vidhya/from-basic-to-intermediate-sql-in-10-minutes->

[42b960ed6f9e](#). Acesso em: 02 jul. 2022.

## Aula 3

Introdução a big data

### Introdução



Começaremos conceituando Big Data, que pode assumir diversos significados, tanto em termos de magnitude quanto de aplicações para diferentes situações. Em seguida, abordaremos algumas das tecnologias de Big Data que são projetadas para analisar, processar e extrair informações de grandes quantidades de dados que não podem ser manipulados com o software de processamento de dados tradicional. As empresas precisavam de tecnologias de processamento de Big Data para analisar a enorme quantidade de dados em tempo real, por exemplo: para criar previsões e reduzir o risco de falhas. Finalmente, veremos algumas aplicações de Big Data e como colocar os dados para funcionar.

### Conceito de big data



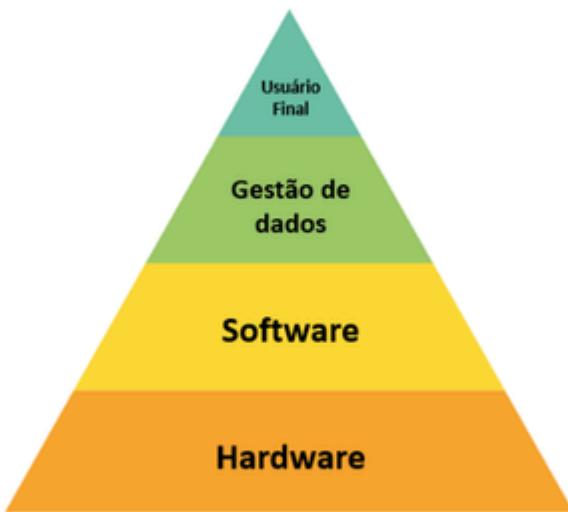
Uma definição simplificada de *Big Data* é aquela que o entende como uma grande coleção de informações, sejam dados armazenados em seu computador pessoal ou em um grande servidor de uma organização, tornando nada fácil a análise desses dados apenas com ferramentas tradicionais (DASGUPTA, 2018).

A internet das coisas, redes sociais e afins, tem contribuído para que o volume de dados crescesse muito, o qual passou a receber o nome de *Big Data*. Por sua vez, análises que incluem uma ampla gama de faculdades, desde mineração de dados básica até aprendizado de máquina avançado, passaram a ser nomeadas como análises de *Big Data* (DASGUPTA, 2018).

### **Blocos de construção da análise de big data**

Basicamente podemos considerar que os sistemas de *Big Data* apresentam quatro camadas principais, algumas dessas camadas são descritas de modo ambíguo na literatura, no entanto, vamos considerar as camadas definidas aqui de modo mais intuitivo e simplificado possível, como demonstrado na Figura 1.

Figura 1 | Níveis/camadas de análise de Big Data



Fonte: Adaptada de Dasgupta (2018).

Os níveis podem ser divididos das seguintes formas:

- **Hardware:** viabilizam tanto os recursos computacionais como os recursos de armazenamento, e os sistemas que suportam a interoperabilidade desses dispositivos é que compõem a camada principal dos blocos de construção (DASGUPTA, 2018).
- **Software:** este recurso permite facilitar a análise dos conjuntos de dados hospedados na camada de hardware, como vemos em sistemas do tipo Hadoop e NoSQL. O software de análise pode apresentar várias subdivisões, e em termos de alto nível apresentam duas classificações principais, que são:
  - a. A mineração de dados:** apresenta recursos que realiza agregações, junções em conjuntos de dados e tabelas dinâmicas em grandes conjuntos de dados. Plataformas de banco de dado não relacionais como o MongoDB, Cassandra e Redis são tidas como ferramentas de mineração de dados de alto nível para análise de *Big Data*.
  - b. A análise estatística:** algumas plataformas que oferecem também recursos de análise estatística, que podem ser desde a regressões simples até a redes neurais avançadas, como por exemplo o Google TensorFlow.
- **Gestão de dados:** governança, acesso e criptografia de dados, dentre outros, são pontos relevantes a serem considerados nas organizações, principalmente no que tange a ao gerenciamento dos dados. Embora sejam menos tangíveis que os hardwares ou softwares, as ferramentas voltadas para o gerenciamento de dados fornecem uma estrutura definida, com a qual as organizações podem de fato obter segurança e conformidade (DASGUPTA, 2018).

- **Usuário final:** este é quem molda o aspecto final do que chamamos de engajamento de análise de *Big Data*. Se pensarmos bem, uma plataforma de dados pode ser considerada boa quando esta poderá ser aproveitada com a maior eficiência e eficácia possível. É neste ponto que entra a importância do papel do profissional que utiliza a plataforma de análise para obter valor (DASGUPTA, 2018).

## Tecnologias big data



As tecnologias de *Big Data* são os utilitários de software projetados para analisar, processar e extrair informações de grandes quantidades de dados não estruturados, que não podem ser manipulados com o software de processamento de dados tradicional.

As empresas precisavam de tecnologias de processamento de *Big Data* para analisar a enorme quantidade de dados em tempo real. Eles usam tecnologias de *Big Data* para criar Previsões para reduzir o risco de falha.

As principais tecnologias de *Big Data* são:

- **Inteligência Artificial:** é conhecida como uma área da ciência da computação, que trabalha com desenvolvimento de máquinas ditas inteligentes que são capazes de realizar tarefas que requerem inteligência humana. Diversas abordagens são levadas em consideração na inteligência artificial, seus principais componentes são o aprendizado de máquina (*machine learning*), aprendizado profundo (*deep learning*) e o aprendizado por reforço

(*reinforcement learning*). A IA tem provocado mudanças consideráveis na área de tecnologia (TYAGI, 2020).

- **Banco de dados NoSQL:** foi desenvolvido para armazenar dados não estruturados com variedades de tipos, isso com um alto desempenho e flexibilidade para lidar com grande escala (TYAGI, 2020). Exemplos incluem: MongoDB, Redis e Cassandra.
- **Data Lakes:** é um “depósito fixo” que armazena todos tipos e modelos de dados (estruturados e não estruturados) e em qualquer escala. As organizações a partir do data lakes passam a ter mais conhecimento e melhor resposta de oportunidades para otimizarem seus negócios, permitindo maior engajamento com seus clientes, melhorias na produtividade, e principalmente tomadas de decisões mais eficazes (TYAGI, 2020).
- **Análise preditiva:** é uma divisão que compõem a análise de *big data*, cujo objetivo é prever comportamentos a partir de dados anteriores. Para isso utiliza métodos de aprendizado de máquina, modelagem estatística e matemática, mineração de dados, e outros (TYAGI, 2020).
- **Análise prescritiva:** permite que as organizações tenham uma melhor direção quanto ao que elas podem fazer para conseguirem atingir os resultados desejados. Por exemplo, pode auxiliar a empresa a entender melhor os fatores relacionados às mudanças do mercado como também prever quais resultados podem ser mais favoráveis (TYAGI, 2020).
- **Blockchain:** é a tecnologia de banco de dados que possibilita uma moeda digital com o Bitcoin, este possui particularidades que tornam os dados seguros – sendo que quando gravados, não permitem uma posterior exclusão ou alteração . É um ecossistema extremamente seguro e atualmente a melhor escolha para aplicações com *Big Data* em setores financeiros, seguros, saúde e outros (TYAGI, 2020).
- **Ecossistema Hadoop:** é uma plataforma que incorpora diversos componentes e serviços variados, como ingestão, armazenamento, análise e manutenção visando auxiliar no trabalho com *Big Data*. Temos no mercado uma ampla variedade de ferramentas e soluções comerciais de ecossistema Hadoop, por exemplo é o Apache Open Source, exemplos de código aberto bem conhecidos são o Spark, Hive, Sqoop e Oozie (TYAGI, 2020).

## Aplicações

# Bancos de Dados Não Relacionais



As aplicações de *Big Data* são a prática ou o processo de extrair dados úteis, que estão em grandes volumes, através do uso de tecnologias apropriadas e aplicá-las para diversos formatos ou finalidades. As aplicações apresentadas a seguir buscam servir de referências ou mesmo como pontos de partida para te ajudar nas aplicações de *Big Data* e sua importância (DASGUPTA, 2018).

- **Mineração de dados:** é o processo de extrair informações de conjuntos de dados através de consultas ou de métodos básicos de resumo, como as agregações. Um exemplo de aplicação seria identificar quais os cinco principais produtos vendidos para cachorro. A partir de um conjunto de dados que contém todos os registros de vendas de produtos para cachorro em um determinado site online, fazemos então o processo de extrair essas informações para que possa identificar os cinco principais produtos vendidos desse seguimento. Bancos como o Cassandra, Redis e MongoDB, são ferramentas que detêm fortes recursos de mineração de dados (DASGUPTA, 2018).
- **Business intelligence:** são ferramentas que permitem que os usuários consultem dados usando uma interface gráfica, como os painéis *front-end*. Um exemplo dessa ferramenta é

o Tableau. Esse tipo de ferramenta teve maior utilização quando passou a ter um crescimento dos dados à medida que os usuários buscavam também extrair informações. As interfaces normalmente são fáceis de usar e estabeleceram uma democratização do acesso analítico aos dados (DASGUPTA, 2018).

- **Visualização:** esta desempenhou um papel primordial na melhor compreensão dos dados, trazendo modelos mais sucintos e intuitivos, especialmente no aspecto das análises mais aprofundadas da natureza do conjunto de dados e sua distribuição. Desenvolvimentos em JavaScript, como D3.js e ECharts, são alguns dos principais pacotes usados para visualização e têm com vantagem o domínio de código aberto (DASGUPTA, 2018).
- **Análise estatística:** possuem ferramentas que possibilitam que sejam executadas operações estatísticas, como são de fácil acesso os usuários finais podem também utilizar. Essas ferramentas existem há muitos anos, porém ficaram em maior evidência após o advento do *Big Data* onde os desafios de analisar volumes de dados consideráveis consequentemente necessitou de aplicar estatísticas mais robustas. As linguagens como R e Python também ganharam força e são exemplos de linguagens muito utilizadas atualmente na área de estatística computacional (DASGUPTA, 2018).
- **Aprendizado de máquina:** é também conhecido como análise preditiva ou modelagem preditiva. Basicamente seu processo de aplicação utiliza algoritmos avançados que inevitavelmente acabam executando milhares de iterações; tão complexas quanto, como também computacionalmente intensivos (DASGUPTA, 2018).
- **Análise de rede social:** temos que o surgimento das redes sociais trouxe consigo uma explosão do big data. Diversas soluções foram desenvolvidas afim de analisar a atividade de mídia social, uma vez que elas passaram a fornecer informações extremamente valiosas sobre como o mercado, como produtos ou campanhas, e tudo em tempo real. Com o auxílio desses insights, as organizações passaram a ajustar seus preços, oferecer promoções e melhoraram até o posicionamento de suas campanhas. Assim, para conhecer o que se passa na mente do consumidor é importante aplicar decisões inteligentes advindas do big data.

## Videoaula: Introdução a big data



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

A tecnologia de hoje nos permite coletar dados a uma taxa surpreendente - tanto em termos de volume quanto de variedade. Existem várias fontes que geram dados, mas, no contexto de Big

Disciplina

# Bancos de Dados Não Relacionais

Data, as fontes primárias serão apresentadas neste bloco. Como também veremos que existem certos aspectos-chave que o tornam muito desafiador. Neste vídeo, discutiremos alguns deles.

## Saiba mais



Para entender melhor sobre o assunto, seguem alguns conteúdos extras.

Vale a pena conferir!

[\*\*"O que é Big Data?"\*\*](#)

[\*\*"Big Data: o que é, conceito e definição"\*\*](#)

Leia também a obra "**Big Data: Técnicas e tecnologias para extração de valor dos dados**"

---

## Referências



BIG DATA: O QUE É, CONCEITO E DEFINIÇÃO. **Cetax.** [S.I.]. Cetax, 2022. Disponível em: <https://cetax.com.br/big-data/>. Acesso em: 21 jul. 2022.

DASGUPTA, N. **Practical Big Data Analytics.** Packt, 2018. Disponível em: [https://www.packtpub.com/product/practical-big-data-analytics/9781783554393?\\_ga=2.33972513.1844863910.1657159495-906940344.1655237593](https://www.packtpub.com/product/practical-big-data-analytics/9781783554393?_ga=2.33972513.1844863910.1657159495-906940344.1655237593). Acesso em: 06 jul. 2022.

MARQUESONE, R. **Big Data:** Técnicas e tecnologias para extração de valor dos dados. Editora Casa do Código, 2016.

ORACLE. **Oracle.** O que é Big Data?. [S.I.]. Oracle, s.d. Disponível em: <https://www.oracle.com/br/big-data/what-is-big-data/#:~:text=A%20defini%C3%A7%C3%A3o%20de%20big%20data,de%20novas%20fontes%20de%20dados>. Acesso em: 21 jul. 2022.

TYAGI, N. Top 10 Big Data Technologies. **Analytics Steps**, 2020. Disponível em: <https://www.analyticssteps.com/blogs/top-10-big-data-technologies-2020>. Acesso em: 06 jul. 2022.

## Aula 4

Migração de bases de dados relacionais para NoSQL

### Introdução



Vamos começar estudando os requisitos para migração, que se trata de uma área que apresenta estudos bem atuais. À medida que tivemos um crescimento considerável de aplicativos e de modo rápido, tivemos como consequência um volume cada vez maior de coleta de dados, o que levou a migração da arquitetura do *Relational Database Management System* (RDBMS) para o banco de dados NoSQL (*Not Only SQL*), este último abriu caminhos e novos modelos para o gerenciamento de banco de dados.

Veremos a seguir os tipos e métodos de migração, sendo que a maioria das migrações de dados que ocorre hoje permite que soluções de banco de dados no local migrem para a nuvem ou para estabelecer sistemas de banco de dados híbridos. Os principais tipos de migração de dados são: de aplicativos, para nuvens, de armazenamento e de banco de dados. Por último, estudaremos algumas aplicações de migração de base de dados.

## Requisitos para migração



Os bancos de dados não relacionais, também conhecidos como NoSQL emergiram nos últimos anos apresentando uma estrutura menos restrita, com um design com esquema escalável e acesso mais rápido, quando comparado com os bancos de dados relacionais. O principal atributo que o banco de dados não relacional tem como diferencial é apresentar um esquema eficiente no tratamento de dados não estruturados, e a utilização de diversas técnicas de modelagem, como armazenamento de valores-chave, modelo de dados de documentos e bancos de dados de gráficos (HANINE *et al.*, 2015).

*Big Data* e análise de dados exigem a migração de bancos de dados relacionais (SQL) para estruturas de dados não relacionais (NoSQL) para representar os dados. Tal transformação é desafiadora devido à falta de processo de transformação automática e à exigência que é necessária para garantir o melhor desempenho e também uma representação precisa.

A migração de dados é o processo de mover dados de um sistema para outro, que pode envolver (RAVENDB, 2022):

- Seleção do conjunto de dados específico a ser movido do sistema anterior para o novo.
- Preparação para transferência e remoção do sistema anterior.
- Transformação para que todos os requisitos sejam preenchidos, como por exemplo as especificações de dados do novo armazenamento e para certificar que os metadados estejam refletindo os dados corretamente.
- Teste e validação dos dados são necessários para que seja removida qualquer duplicação, além de serem corrigidas falhas e removidos dados corrompidos.

- Carregamento de dados no sistema de destino e também é importante realizar testes para averiguar possíveis problemas pendentes.

A realização de fato o processo de migração de dados, e a escolha de qual ferramenta será adotada, caberá a organização escolher e a da expertise dos responsáveis pela migração. Porque o processo para efetuar a migração dos dados RDBMS para um banco de dados do tipo colunar, como temos no Cassandra, é bem diferente de uma migração para armazenamentos do tipo de valores-chave como é temos no MongoDB. Porém temos em comum as etapas do processo, em que na maioria envolvem as mesmas quatro etapas, como identificam Kristina Chodorow e Michael Dirolf citados por Asay (2012):

- Imprescindível conhecer o banco de dados NoSQL que irá trabalhar, baixe a ferramenta, leia seus tutoriais, e se possível aplique alguns projetos de sucessos previamente.
- Planeje como seria representar o modelo desejado em seu armazenamento de documentos, reveja as opções e estude qual seria a melhor, como: valor/chave; coluna, gráfico, conforme o que for mais apropriado.
- Transfira os dados do banco de dados relacional para seu banco de dados não relacional. Em seguida, carregue os dados no seu novo modelo de banco de dados para averiguar se está correto.
- Consulte o seu banco de dados agora não relacional, reescrevendo o código do seu aplicativo através de instruções como `insert()` ou `find()`.

Essas etapas apresentadas podem parecer diferentes a depender do modelo do banco de dados adotado, porém inicialmente é um guia básico que poderá te auxiliar em como iniciar uma migração e quais passos mínimos devem ser seguidos.

## Métodos de migração

# Bancos de Dados Não Relacionais



A maioria das migrações de dados ocorre hoje para permitir que soluções de banco de dados no local migrem para a nuvem ou para estabelecer sistemas de banco de dados híbridos. Os principais tipos de migração de dados são:

- **Migração de aplicativos:** Ocorre quando você substitui um aplicativo por outro. Pode incluir a mudança de um aplicativo local para um na nuvem.
- **Migração para a nuvem:** É a transferência de dados de um sistema local para uma plataforma de nuvem ou de uma plataforma de nuvem para outra.
- **Migração de armazenamento:** É a transferência dos dados para um armazenamento de dados com maior capacidade.
- **Migração de banco de dados:** Envolve mover seus dados de um fornecedor de banco de dados para outro.

## Principais abordagens para migração de dados de bancos de dados relacionais para NoSQL

- **Modelo intermediário com utilização de recursos de dados e consulta:** a principal característica desta abordagem é que ela se trata de um modelo uniforme e, de acordo com Gotiya et al. (2017), temos que:

1. Para termos um modelo de entrada é fornecido os dados do banco de dados relacional.

2. O modelo intermediário gera um objeto para cada entidade e os recursos de dados e de consulta são criados para os dados de origem.
  3. Ocorre a transferência para um modelo físico conforme a estratégia predefinida como saída.
  4. Depois do mapeamento do modelo, os dados são migrados de forma rápida e fácil.
- **NoSQLLayer:** as consultas são capturadas pelo NoSQLLayer e convertidas no formato específico do banco de dados NoSQL, após a conversão, os dados são capturados de novo por um *framework* e convertido agora para um formato específico do aplicativo (GOTIYA *et al.*, 2017).
  - **Adaptador de Dados:** este sistema busca integrar o SGBD e o bancos de dados NoSQL, que tratam da transformação do banco de dados. Suas principais características são a sua interface, o conversor e abordagem de consulta (GOTIYA *et al.*, 2017).
  - **Estrutura de Mapeamento Automático:** trata-se de um *framework* que oferece mapeamento automático de bancos de dados relacionais para um banco de dados NoSQL. O mapeamento envolve algumas etapas, como desenvolver o banco de dados e criar novas tabelas no banco de dados e seus relacionamentos com demais tabelas (GOTIYA *et al.*, 2017).

## Fatores importantes ao migrar de SQL para NoSQL:

- **Redesenhe o esquema:** Apenas a camada de dados e o esquema precisam ser alterados de acordo com um modelo otimizado para NoSQL.
- **Refatoração:** A lógica de dados e o esquema RDBMS precisam ser refatorados em um modelo otimizado para NoSQL.
- **Hospede primeiro e o faça o processo de otimização conforme a necessidade:** a hospedagem tem que ocorrer e, se necessário, deve-se otimizar o processo para um melhor desempenho.

## Aplicação de migração de bases dados relacionais para NoSQL



Um dos processos mais desafiadores é realizar a migração de banco de dados SQL para o MongoDB (banco de dados NoSQL). Temos diversas opções disponíveis que explicam essas migrações, e neste bloco veremos alguns deles que poderão te ajudar neste processo, quando for necessário fazer também a sua migração de banco de dados.

### Métodos para migrar SQL para MongoDB: ferramenta Mongify

Quando pensamos em uma ferramenta fácil para migrar banco de dados SQL para MongoDB, podemos dizer que o Mongify é uma delas. Esta possui suporte de qualquer ferramenta de banco de dados SQL porque possui o ActiveRecord integrado, e suas instalação é bem fácil, basta seguir os passos do Código 1 e 2 a seguir (RAJAGOPAL, 2019).

Código 1 | Execute o seguinte comando

```
1 | gem install mongify
```

Fonte: Rajagopal (2019).

Código 2 | Migração usando o Mongify

```
1 | mongify command database.config [database_translation.rb]
```

Para entendermos melhor o que foi escrito no Código 2 de migração de banco de dados, temos que o “mongify” é a ferramenta utilizada; o “command” que foi escrito está relacionado ao

processo; o “database.config” relaciona o arquivo que possui os detalhes de conexão do banco de dados SQL e NoSQL. Por último o “database\_translation.rb” refere-se ao arquivo que é escrito de modo manual ou é gerado usando o próprio processo de tradução (RAJAGOPAL, 2019).

Podemos também utilizar outros comandos que nos auxiliam no processo de migração, como os apresentados a seguir:

1. “check” or ‘ck’: garante que tanto a base de dados bem como as configurações do arquivo estejam corretas ou não, e se toda a conexão citada no arquivo está em pleno funcionamento.
2. “translation” or “tr”: este gera de forma automática o comando “database\_translation.rb”.
3. “process” or “pr”: neste os dados SQL são convertidos em documentos do banco de dados no MongoDB.

## Ferramenta de migração de dados MySQL para MongoDB

A ferramenta de migração de dados MySQL para MongoDB pode ser clonada a partir do projeto MySQL-Mongo-Migrate hospedada no GitHub. Temos dois métodos que podem ser utilizados para instalar e configurar esta ferramenta, que são (RAJAGOPAL, 2019):

### Método 1:

1. Clone o projeto do GitHub: <https://github.com/dannysoftie/mysql-mongo-migrate.git>.
2. Instale as dependências - npm install.
3. Execute o “Migrar” – npm run migrate.

### Método 2:

1. Instale o pacote do repositório global do NPM –npm i - g mysql-mongo-migrate.
2. Execute a ferramenta diretamente da linha de comando – \$> mysql-mongo-migrate.

Para obter mais instruções e detalhes, visite o projeto de Danny Softtie (SOFTTIE, 2019).

### Migração de SQL para MongoDB com Studio 3T

No próprio Studio 3T já aparece o recurso “SQL Import to MongoDB”, que vai possibilitar a importação dos registros contidos em uma única tabela para uma única coleção. Para abrir a migração SQL para MongoDB no Studio 3T basta seguir os passos sugeridos por Bastardas (2022):

1. Clique no botão “SQL Migration” contido na barra de ferramentas ou clique com o botão direito do mouse em um servidor, e vai até em coleção na Connection Tree e selecione a opção “SQL Migration”;
2. Por último selecione SQL → Migração do MongoDB. Irá abrir uma nova guia onde você poderá configurar e executar a importação.

O Studio 3T é interessante porque ele faz a conexão para a qual os registros serão exportados, como também faz a detecção do banco de dados de destino. Além dos métodos acima, existem muitas maneiras de migrar RDBMS para MongoDB. As formas acima mencionadas são formas populares sugeridas de migração.

## Videoaula: Migração de bases de dados relacionais para NoSQL



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Dada a escolha de um banco de dados relacional versus um banco de dados NoSQL, tornou-se mais importante selecionar o tipo certo de banco para armazenar dados. Torna-se imperativo escolher corretamente e, se necessário, migrar o banco de dados RDBMS existente para NoSQL para atender à nova dinâmica de requisitos de negócios. Neste vídeo, veremos dicas que ajudarão a decidir o banco de dados certo para sua necessidade.

### Saiba mais



Para entender melhor sobre migração de banco de dados, seguem alguns links, que abordam com mais detalhes esse assunto. Vale a pena conferir:

# Bancos de Dados Não Relacionais

["Comparação de Metodologias de Migração de Bancos de Dados Relacionais para Bancos Orientados a Documentos"](#)

["O que são os bancos de dados NoSQL?"](#)

["SQL Server: Importando/Exportando um banco de dados"](#)

## Referências



ASAY, M. Migrating from a relational to a NoSQL cloud database. **TechRepublic**, 2012. Disponível em: <https://www.techrepublic.com/article/migrating-from-a-relational-to-a-nosql-cloud-database/>. Acesso em: 09 jul. 2022.

BASTARDAS, Andreu. SQL to MongoDB Migration. **Studio3T**, 2022. Disponível em: <https://studio3t.com/knowledge-base/articles/sql-to-mongodb-migration/>. Acesso em: 09 jul. 2022.

GEEKS FOR GEEKS. Strategies For Migrating From SQL to NoSQL Database. **Geeks For Geeks**, 2022. Disponível em: <https://www.geeksforgeeks.org/strategies-for-migrating-from-sql-to-nosql-database/>. Acesso em: 09 jul. 2022.

GHOTIYA, S.; MANDAL, J.; KANDASAMY, S. Migration from relational to NoSQL database. In: **IOP Conference Series: Materials Science and Engineering**. IOP Publishing, 2017. p. 042055.

HANINE, M.; BENDARAG, A.; BOUTKHOUM, O. Data migration methodology from relational to NoSQL databases. **International Journal of Computer, Electrical, Automation, Control and Information Engineering**, v. 9, n. 12, p. 2566-2570, 2015.

MICROSOFT. **Azure.** O que são os bancos de dados NoSQL?. [S.I.]. Microsoft, s. d. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-nosql-database/>. Acesso em: 21 jul. 2022.

RAJAGOPAL, K. How to Migrate SQL to MongoDB. **Digital Varys**, 2019. Disponível em: <https://digitalvarys.com/how-to-migrate-sql-to-mongodb/>. Acesso em: 09 jul. 2022.

RAVEN DB. NOSQL Database data migration. RavenDB, 2022. Disponível em: <https://ravendb.net/news/articles/data-migration>. Acesso em: 09 jul. 2022.

SOFFTIE, Danny. Mysql-mongo-etl. **Github**, 2019. Disponível em: <https://github.com/dannyssofttie/mysql-mongo-etl>. Acesso em: 09 jul. 2022.

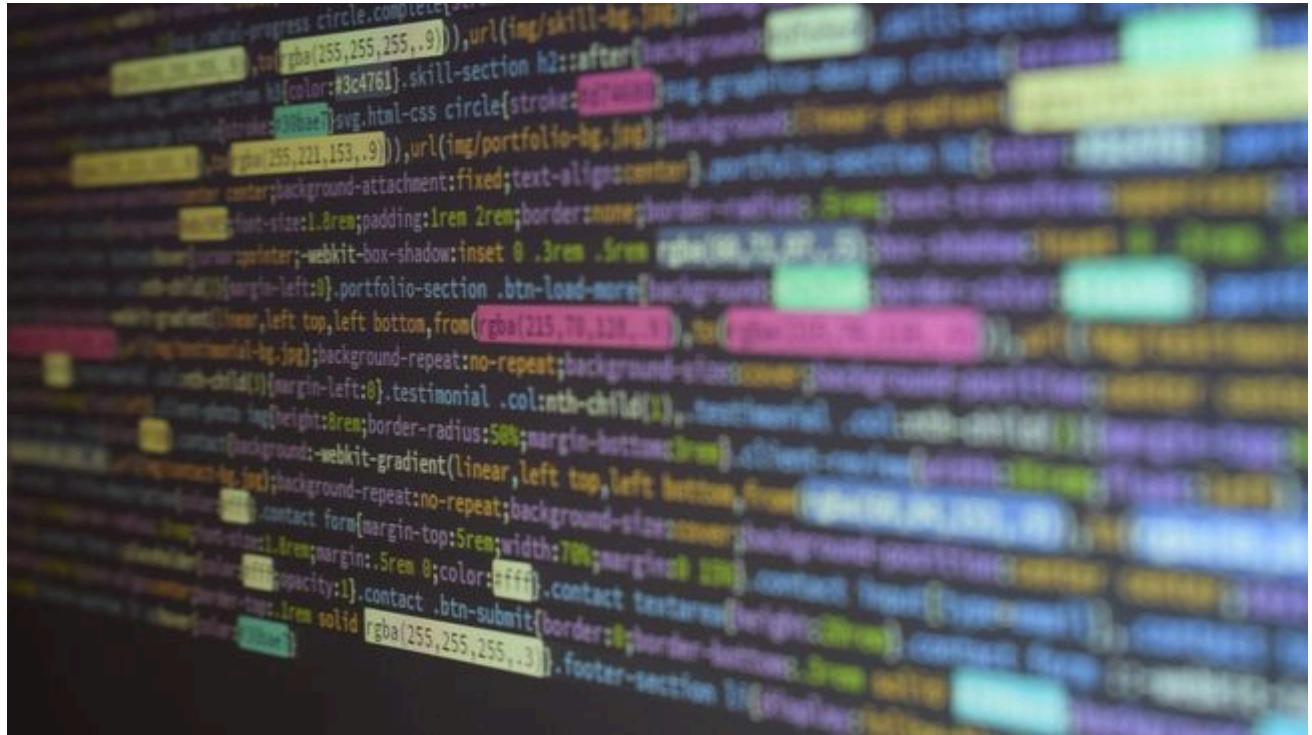
SOUZA, V. C. O.; PAULA, M. M. V.; BARROS, T. C. G. M. Comparação de Metodologias de Migração de Bancos de Dados Relacionais para Bancos Orientados a Documentos. **XI Computer on the beach**. Balneário de Camboriú, SC, set. 2020. Disponível em: <https://periodicos.univali.br/index.php/acotb/article/view/16777/9503>. Acesso em: 21 jul. 2022.

UMBLER. **Umbler.** SQL Server: Importando/Exportando um banco de dados. [S.I.]. Umbler Tutoriais, s. d. Disponível em: <https://help.umbler.com/hc/pt-br/articles/203989419-SQL-Server-Importando-Exportando-um-banco-de-dados>. Acesso em: 21 jul. 2022.

## Aula 5

Revisão da unidade

### Processo de migração de um banco de dados relacional para NoSQL



Os dados são a espinha dorsal de todos os projetos que envolvem tecnologia, não importa o tamanho ou mesmo a aplicação que a organização esteja desenvolvendo, com certeza precisará obtê-los ou analisá-los. Geralmente, os dados necessários são armazenados em um Sistema de Gerenciamento de Banco de Dados (SGBD). E para interagir e se comunicar com o SGBD, é preciso usar o Structured Query Language (SQL) (METWALLI, 2022).

As principais limitações do banco de dados relacionais são:

- **Escalabilidade:** “os bancos de dados relacionais são projetados para serem executados em um único servidor, a fim de manter a integridade dos mapeamentos de tabelas e evitar os problemas da computação distribuída” (ALLEN, 2022, [s. d.], tradução nossa).
- **Complexidade:** os dados têm que caber em tabelas de qualquer maneira. Se seus dados não couberem em tabelas, você precisará projetar sua estrutura de banco de dados de modo complexo e difícil de manusear.

Já no **NoSQL**, comumente referido como *Not Only SQL*, os dados são não estruturados e sem esquema e podem ser armazenados em várias coleções e nós. Seus principais benefícios são (LOGINRADIUS, 2022):

- **Escalabilidade:** tem como objetivo ser bancos de dados expansíveis horizontalmente, o que significa que podemos dimensioná-lo apenas adicionando mais máquinas em seu *pool* de recursos (LOGINRADIUS, 2022; SMALLCOMBE, 2021).
- **Baixo custo:** por necessitar de uma menor manutenção para o seu gerenciamento, além de possuir por exemplo reparo automático e maior facilidade na distribuição de dados,

# Bancos de Dados Não Relacionais



proporciona um menor custo para sua manutenção (LOGINRADIUS, 2022; SMALLCOMBE, 2021).

- **Menor custo de servidor e código aberto:** para implementar um banco de dados NoSQL normalmente usamos servidores muito mais baratos para gerenciar os dados e transações se comparado ao banco de dados SQL (LOGINRADIUS, 2022).
- **Não possui esquema ou modelo de dados fixos:** o formato ou modelo de dados pode ser alterado a qualquer momento, sem interrupção do aplicativo (LOGINRADIUS, 2022).
- **Suporte ao cache integrado:** este suporte ao cache na memória do sistema, possibilita a melhoria do desempenho da saída dos dados (LOGINRADIUS, 2022).

O banco de dados não relacional segue as propriedades do Teorema CAP, que é uma abordagem que foi defendida por Eric Brewer, que tem como prerrogativa “garantir no máximo duas das três propriedades a seguir ao mesmo tempo” (GESSERT, 2016):

- **Consistência (C):** todos conseguem ter a mesma visão dos dados de modo síncrono.
- **Disponibilidade (A):** cada nó que não apresenta falha no sistema poderá sempre admitir solicitações de leitura e escrita de clientes.
- **Tolerância à partição (P):** o sistema preserva tanto as garantias de consistência, como a disponibilidade dos dados quando na presença de perda da mensagem que está entre os nós ou mesmo de uma possível falha, tanto total como parcial do sistema.

O Big Data passou a ser uma referência do grande volume de dados que temos atualmente, e as suas análises tornou-se uma área interdisciplinar, porque abrange desde a mineração de dados até aprendizado de máquina avançado (DASGUPTA, 2018). As tecnologias que surgiram para lidar com o *Big Data* tornaram fundamentais para que a análise, o processamento e a extração de informações passem a ser manipulados, uma vez que estes possuem limitações quando trabalhado com software de processamento de dados tradicionais. Sua importância está por exemplo nas previsões e reduções de riscos de falhas para grandes empresas.

## Videoaula: Revisão da unidade



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Como os bancos de dados são tão críticos para o fluxo de informações em uma empresa, é essencial entender como eles funcionam e beneficiam seus negócios. Vamos analisar algumas das qualidades de um excelente banco de dados para aplicar em uma empresa. Para isso,

veremos primeiro como funcionam os bancos de dados relacionais e não-relacionais, seus desafios e as principais diferenças entre os bancos de dados SQL e NoSQL.

## Estudo de caso



As migrações de banco de dados, também conhecidas como migrações de esquema, migrações de esquema de banco de dados ou simplesmente migrações, são conjuntos controlados de alterações desenvolvidas para modificar a estrutura dos objetos em um banco de dados relacional.

Existem dois tipos principais de estratégias de transformação de esquema empregadas pelo software de migração: migrações baseadas em estado e migrações baseadas em alterações. Qualquer um pode ser usado efetivamente para gerenciar o processo e, às vezes, é útil empregar uma combinação de ambos. Os pontos fortes de cada abordagem, no entanto, geralmente determinam qual se encaixa melhor para um projeto com base em quão bem ele corresponde ao estilo de desenvolvimento da equipe envolvida.

### Migrações baseadas no estado

O software de migração baseado em estado cria artefatos que descrevem como recriar o estado do banco de dados desejado a partir do zero. Os arquivos que ele produz podem ser aplicados a um sistema de banco de dados relacional vazio para atualizá-lo totalmente.

### Migrações baseadas em alterações

A principal alternativa para migrações baseadas em estado é um sistema de migração baseado em mudanças. As migrações baseadas em alterações também produzem arquivos que alteram as estruturas existentes em um banco de dados para chegar ao estado desejado. Em vez de

# Bancos de Dados Não Relacionais

descobrir as diferenças entre o estado de banco de dados desejado e o atual, essa abordagem se baseia em um estado de banco de dados conhecido para definir as operações para trazê-lo para o novo estado. Arquivos de migração sucessivos são produzidos para modificar ainda mais o banco de dados, criando uma série de arquivos de alteração que podem reproduzir o estado final do banco de dados quando aplicados consecutivamente.

Após a criação dos artefatos que descrevem o estado desejado, a migração real envolve a comparação dos arquivos gerados com o estado atual do banco de dados. Esse processo permite que o software analise a diferença entre os dois estados e gere um novo arquivo ou arquivos para alinhar o esquema de banco de dados atual com o esquema descrito pelos arquivos.

Agora que apresentamos as duas principais classes de sistemas de migração, defina como você realmente usará essas ferramentas durante o desenvolvimento, implantação e colaboração de aplicativos. A ideia deste estudo de caso é que você possa explicar à sua equipe de trabalho, que necessitará de realizar a migração do banco de dados da empresa em que trabalha, como você usará as migrações de banco de dados e como escolherá as melhores ferramentas de migração para a organização. Prepare então uma apresentação para o grupo, lembrando também de abordar brevemente as vantagens e desvantagens das ferramentas de migração.

## Saiba mais



Tradicionalmente, em muitos projetos de banco de dados podemos utilizar banco relacionais ou não relacionais. Porém a migração entre eles tornou-se um desafio. Alguns pontos-chaves são

# Bancos de Dados Não Relacionais

importantes que fiquem fixos para que você possa realizar esse processo tranquilamente e são eles:

- 1º A migração de dados é o processo de mover dados de um sistema para outro e envolve:**  
SELECIONAR OS DADOS ® PREPARAR DADOS PARA TRANSFERÊNCIA ® TRANSFORMAR OS DADOS ® TESTAR E VALIDAR OS DADOS ® CARREGAR OS DADOS NO NOVO BANCO DE DADOS  
**2º A migração de dados pode acontecer por vários motivos. São eles:**

- Atualização de todo o sistema, especialmente quando é necessário modificar hardwares com maior capacidade de armazenamento e processamento.
- Atualização de bancos de dados.
- Uma fusão ou reestruturação da organização, isso acaba exigindo que os dados sejam mesclados.
- Recuperação de possível danos no sistema.

A maioria das migrações de dados que acontecem hoje são para banco de dados nas nuvens ou para estabelecer sistemas de banco de dados híbridos.

- 3º Os desafios mais comuns para a migração de dados são:**

- Governança de dados adequada: é importante se certificar de atribuir as responsabilidades para cada um dos envolvidos no processo de migração.
- Falta de perícia: como a migração de dados é algo complexo, é fundamental que a equipe responsável pela migração tenha um certo grau de experiência neste processo, além de possuir as ferramentas corretas para migração.
- Falta de planejamento: o não planejamento de uma migração de banco de dados pode acabar resultando em sérios problemas, como até perda da confiabilidade dos dados migrados. É de vital importância confirmar que o plano que está sendo implementado já foi testado antes.

## Videoaula: Resolução do estudo de caso



### Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Embora a prevenção da perda de dados seja geralmente um dos objetivos do software de migração, as alterações que eliminam ou modificam destrutivamente as estruturas que atualmente hospedam os dados podem resultar em exclusão. Para lidar com isso, a migração geralmente é um processo supervisionado que envolve a inspeção dos scripts de alteração

# Bancos de Dados Não Relacionais

resultantes e a realização de quaisquer modificações necessárias para preservar informações importantes.

## **Quais são as vantagens das ferramentas de migração?**

As migrações são úteis porque permitem que os esquemas de banco de dados evoluam à medida que os requisitos mudam. Eles ajudam os desenvolvedores a planejar, validar e aplicar com segurança as alterações de esquema em seus ambientes. Essas mudanças compartimentadas são definidas em um nível granular e descrevem as transformações que devem ocorrer para mover entre várias "versões" do banco de dados. Em geral, os sistemas de migração criam artefatos ou arquivos que podem ser compartilhados, aplicados a vários sistemas de banco de dados e armazenados no controle de versão. Isso ajuda a construir um histórico de modificações no banco de dados que pode estar intimamente ligado às alterações de código de acompanhamento nos aplicativos clientes.

## **Quais são algumas desvantagens das ferramentas de migração?**

Os sistemas de migração não estão isentos de falhas, mas, felizmente, a maioria delas pode ser atenuada por meio de processos e supervisão. Como as migrações modificam as estruturas de banco de dados existentes, deve-se tomar cuidado para evitar a perda de dados. Isso pode ser causado por suposições incorretas feitas pelas ferramentas ou por transformações que exigem uma compreensão do significado por trás das estruturas de dados. Embora possa ser tentador presumir que os arquivos de migração gerados estão corretos, uma vez que os arquivos de migração se preocupam principalmente com as estruturas de dados, é sua responsabilidade garantir que os dados também sejam preservados ou transformados corretamente.

## **Como utilizar as migrações de banco de dados?**

Durante o desenvolvimento: As migrações de banco de dados tornam-se potencialmente relevantes a partir do momento em que fazemos a primeira modificação no esquema de dados inicial. Na verdade, a maioria das ferramentas de migração gera arquivos de migração para trazer um banco de dados relacional de um estado completamente vazio para o esquema inicial. À medida que você desenvolve seu aplicativo, a forma dos dados que deseja armazenar, os requisitos do sistema de banco de dados e os detalhes específicos que deseja preservar mudam frequentemente. Isso pode acontecer à medida que você entende o espaço do problema mais completamente ou quando recursos adicionais exigem dados adicionais ou um layout diferente das informações atuais. É importante definir essas alterações de esquema à medida que ocorrem durante o desenvolvimento para garantir que os artefatos sejam sincronizados e implementados com o código associado.

Armazenar e gerenciar arquivos de migração com a base de código associada permite que as alterações do esquema do banco de dados passem pelo mesmo processo de revisão que as alterações de código recebem. Conforme mencionado anteriormente, as migrações baseadas em estado podem facilitar esse processo, porque os conflitos entre o estado desejado são aparentes ao diferenciar os arquivos. Os sistemas baseados em alterações exigem mais cuidado e resolução prática quando ocorrem conflitos, pois a ordenação é importante, e toda a estrutura de dados não é visível em nenhum dos arquivos de migração iterativos.

## **Durante o teste e a implantação**

Depois que as migrações forem criadas, você geralmente as implantará em ambientes cada vez mais importantes para garantir que funcionem conforme o esperado e que todos os requisitos do seu código sejam atendidos. Embora você provavelmente tenha desenvolvido e aplicado as

migrações em um ambiente de desenvolvimento, provavelmente precisará testá-las em ambientes adicionais com dados, integrações e carga mais realistas, assim como faria com alterações de código.

Neste ponto, você precisa garantir que os arquivos de migração se apliquem corretamente ao banco de dados de destino, que o estado de destino inicial compartilhe a estrutura e as suposições que você tinha ao desenvolver os arquivos de migração e que os dados mantidos pelo banco de dados não sejam afetados negativamente pelas mudanças que você está fazendo. Uma mistura de verificações manuais e automatizadas pode ser realizada para confirmar que esses requisitos são atendidos.

Depois que as alterações forem validadas como seguras e funcionais nos ambientes de teste, elas poderão estar prontas para serem aplicadas aos sistemas de banco de dados de produção. Antes de aplicar as migrações aos sistemas de produção, é importante considerar a realização de um backup completo dos dados atuais se uma cópia recente não estiver disponível no momento. Os backups são vitais caso a migração tenha consequências imprevistas ou se as diferenças entre os ambientes de teste e a produção causarem perda de dados.

## Como escolher as melhores ferramentas de migração?

Dadas todas as informações que você cobriu até agora, como você realmente decide sobre as soluções de migração mais adequadas para os projetos da organização? Há uma série de considerações diferentes que você pode usar para restringir suas melhores escolhas.

Às vezes, a linguagem da base de código, a estrutura de desenvolvimento ou o *back-end* do banco de dados que está usando sugerem fortemente as ferramentas que você deve usar para gerenciar migrações. Por exemplo, embora muitas vezes não sejam requisitos rígidos, muitos *frameworks* da web incluem seus próprios sistemas de migração para gerenciar o esquema de seus bancos de dados de apoio. Eles geralmente têm boa integração com o restante das ferramentas da estrutura e podem ajudar a reduzir o atrito de gerenciar as alterações do banco de dados como um processo separado.

Outra consideração que pode afetar sua seleção é o nível de maturidade e a quantidade de suporte fornecido para uma determinada ferramenta. Os arquivos de migração produzidos por várias ferramentas tendem a não ser diretamente compatíveis entre si, embora possam ser aplicados diretamente como SQL puro. Escolher uma opção estável pode ajudá-lo a evitar cenários em que uma ferramenta cai em desuso e não é mais mantida, forçando você a alterar seu processo de migração no meio do desenvolvimento.

Um outro ponto que você pode querer considerar é o formato dos artefatos de migração reais. Se eles forem arquivos SQL bem formatados e bem organizados, você provavelmente poderá contar com a capacidade de entender as alterações que os arquivos produzem. Da mesma forma, se as migrações forem escritas na linguagem de programação que o restante do seu projeto está usando, o significado geralmente pode ser fácil de analisar e exigir menos troca de contexto de outras tarefas de desenvolvimento. Evite ferramentas que produzem arquivos de migração difíceis de entender ou modificar manualmente.

Outro fator em sua decisão pode envolver recursos adicionais fornecidos pelos sistemas de migração. Algumas ferramentas oferecem grande integração com outras ferramentas ou oferecem opções flexíveis, como a capacidade de compactar arquivos de migração baseados em alterações anteriores em um arquivo de estado de "ponto de verificação". As ferramentas têm suporte variado para inspecionar falhas de migração ou reverter alterações quando ocorrerem

# Bancos de Dados Não Relacionais

problemas. Seus requisitos e filosofia de desenvolvimento podem afetar quais desses recursos você precisa

## Resumo Visual

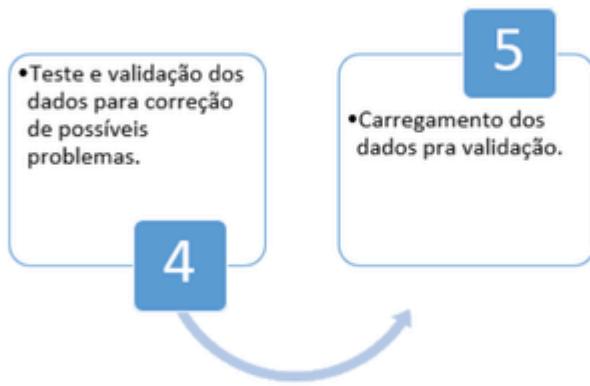


### MIGRAÇÃO DE BANCO DE DADOS

A migração de dados é o processo de mover dados de um sistema para outro, que pode envolver:

# Bancos de Dados Não Relacionais

A ampli



Alguns dos fatores importantes que precisam ser observados para migrar de SQL para NoSQL:

## Redesenhe o esquema

- Apenas a camada de dados e o esquema precisam ser alterados de acordo com um modelo otimizado para NoSQL. Na lógica de negócios, nenhuma alteração é necessária.

## Refatoração

- A lógica de dados e o esquema RDBMS precisam ser refatorados em um modelo otimizado para NoSQL.

**Hospede primeiro e faça o processo de otimização conforme a necessidade:**

- Na tecnologia proposta, a hospedagem tem que ocorrer e, se necessário, deve-se otimizar o processo para um melhor desempenho.

## Referências



ALLEN, M. Relational databases are not designed for scale. **Mark Logic**, 2022. Disponível em: <https://www.marklogic.com/blog/relational-databases-scale/>. Acesso 06 jul. 2022.

DASGUPTA, N. **Practical Big Data Analytics**. Packt, 2018. Disponível em: [https://www.packtpub.com/product/practical-big-data-analytics/9781783554393?\\_ga=2.33972513.1844863910.1657159495-906940344.1655237593](https://www.packtpub.com/product/practical-big-data-analytics/9781783554393?_ga=2.33972513.1844863910.1657159495-906940344.1655237593). Acesso em: 06 jul. 2022.

GESSERT, F. NoSQL Databases: a Survey and Decision Guidance. **Medium**, 2016. Disponível em: <https://medium.baqend.com/nosql-databases-a-survey-and-decision-guidance-ea7823a822d>. Acesso em: 02 jul. 2022.

LOGINRADIUS. RDBMS vs NoSQL. **LoginRadius**, 2022. Disponível em: <https://www.loginradius.com/blog/engineering/relational-database-management-system-rdbms-vs-nosql/#:~:text=NoSql%20database%20implementation%20is%20easy,than%20the%20cost%20of%20RDBMS>. Acesso em: 27 jun. 2022.

METWALLI, S. A. SQL vs. NoSQL: Which Should You Choose? **Builtin**, 2022. Disponível em: <https://builtin.com/data-science/sql-vs-nosql>. Acesso em: 27 jun. 2022.

SMALLCOMBE, Mark. SQL vs NoSQL: 5 Critical Differences. **Integrate.io**, 2021. Disponível em: <https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable,data%20like%20documents%20or%20JSON>. Acesso em: 27 jun. 2022.

## Aula 6

Roteiro de Aula Prática

### Roteiro de Aula Prática

Disciplina

# Bancos de Dados Não Relacionais



[Clique aqui](#) para acessar o manual de instalação!

[Clique aqui](#) para acessar o roteiro de aula prática!