Gestão de Tecnologia da Informação

Banco de dados avançado e não relacionais

Profa. Ms. Fabiana A. Rodrigues

EDUCAÇÃO METODISTA

Dia da aula: segunda-feira
Horário: 19h30 às 21h00

CRITÉRIOS DE AVALIAÇÃO

	Data	%
» Prova 1	23/09 a 27/09	30%
» Prova 2	25/11 a 29/11	30%
» Atividades Docentes	Durante a disciplina	40%
»Avaliação Substitutiva	De 02 a 06/12	
»Avaliação Suplementar	De 09 a 13/12	
»Apresentação PEU	Data será agendada	

EDUCAÇÃO METODISTA

AVALIAÇÃO SUBSTITUTIVA

Substituirá a P1 ou P2, e deverá contemplar o conteúdo de todo o semestre, de forma temática/modular.

AVALIAÇÃO SUPLEMENTAR

Avaliação Suplementar - 100%
Para quem: alunos com frequência mínima de 75% e nota menor que 6 e maior ou igual a 4.

EDUCAÇÃO METODISTA

EMENTA

- ✓ Bancos de Dados Não Relacionais.
- ✓ Introdução aos Bancos de Dados Distribuídos.
- ✓ Conceitos de BigData.
- ✓ Conceitos de Business Intelligence.
- ✓ Bancos de Dados Avançados: Normalização de Banco de Dados, Views Estrutura de Indexação de Tabelas, Conceito e Funcionalidades em Otimização de consultas, Introdução ao PL/SQL e seus Conceitos básicos: Expressões e estrutura de programação em PL/SQL, criação de Stored Procedure e Functions no banco de dados.
- ✓ Linguagens SQL - DQL e DTL

EDUCAÇÃO METODISTA

CONTEÚDO PROGRAMÁTICO

Teoria:

- Introdução aos Bancos de Dados não relacionais.

- Introdução aos Bancos de Dados Distribuídos

- Normalização de Banco de Dados Relacionais (Anomalias e Formas Normais - 3FN)

- Performance de Banco de Dados Relacionais (Índices e Backup)

- Evolução de Dado, Informação e Conhecimento e Introdução ao Business Intelligence - Data Warehouse/Data Marts e Modelagem Dimensional - Introdução ao Big Data (Hadoop e MapReduce/Spark) - Arquitetura dos Bancos de Dados NoSQL e Cloud Databases - Tipos de Banco de Dados NoSQL (Orientado a Documento, Chave-valor, Colunar, Grafo) - Introdução a Data Science e Mineração de Dados Gestão da Qualidade: atributos mensuráveis em projetos

- Inteligência de Dados e Computação Cognitiva - Atividades em Grupo sobre Normalização de Banco de Dados e Modelagem Dimensional

Prática em Laboratório: - Programação de Bancos de Dados Relacionais (Bloco Anônimos - MySQL) - Programação de Bancos de Dados Relacionais (Procedures - Oracle) - Programação de Bancos de Dados Relacionais (Functions - MySQL) - Python - Introdução e como trabalhar com esta linguagem, voltada a Data Science - MongoDB - Introdução, como usar e trabalhar com Collections.

Aula 1
Conceitos básicos dos bancos de dados não relacionais

EDUCAÇÃO METODISTA

Banco de Dados

- Banco de dados podem funcionar de diversas maneiras e possuir características bastante distintas entre si. Eles são projetados para serem extremamente escaláveis, focando em desempenho, confiabilidade dos dados e flexibilidade do modelo de dados. Muitos desses sistemas são otimizados para lidar com grandes volumes de dados distribuídos por múltiplos servidores, oferecendo alta performance em consultas e operações de escrita.
- Além disso, os bancos de dados suportam diferentes tipos de dados, como documentos, grafos e colunas, permitindo uma modelagem de dados mais flexível e adaptável às necessidades específicas das aplicações. Essa diversidade de modelos e capacidades torna os bancos de dados uma escolha poderosa para aplicações que exigem escalabilidade, flexibilidade e alto desempenho.

O que é Banco de dados relacional?

- Um banco de dados relacional é uma coleção de informações ou dados estruturados, tipicamente armazenados eletronicamente em um sistema de computador.

- Eles são conhecidos por serem sistemas de propósito geral, utilizando a linguagem SQL (Structured Query Language, ou Linguagem de Consulta Estruturada) para consulta e manipulação de dados e estruturas de dados. Além disso, os bancos de dados relacionais empregam as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) para garantir a confiabilidade das transações. Essas propriedades são fundamentais para qualquer aplicação crítica, assegurando que as operações sejam executadas de maneira confiável e segura.

EDUCAÇÃO METODISTA

Banco de dados Relacional ACID

- Uma das considerações mais relevantes ao decidir pela utilização de uma tecnologia de banco de dados é verificar se o banco possui propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade).
- Essas quatro características fornecem garantias essenciais que impactam diretamente o negócio, assegurando a confiabilidade e a integridade das transações realizadas no sistema.

EDUCAÇÃO METODISTA

Banco de dados Relacional - Propriedades ACID

ATOMICIDADE: Controle sobre início e fim da transação, é a garantia que todo o bloco de transações foi executado integralmente.

Um **exemplo prático** de como as propriedades ACID se aplicam a uma transação de transferência de dinheiro entre contas bancárias. Suponha que temos um sistema bancário com duas contas: Conta A e Conta B.

Vamos analisar como as propriedades ACID se aplicam a uma transação de transferência de dinheiro de uma conta para outra.

Imagine que um cliente deseja transferir R$100 da Conta A para a Conta B. A atomicidade garante que a transferência é tratada como uma única operação. Se a transferência for concluída com sucesso, R$100 é subtraído da Conta A e adicionado à Conta B. Se ocorrer algum erro no meio da operação (por exemplo, falta de fundos), nenhuma parte da transação é realizada e ambos os saldos permanecem inalterados.

Banco de dados Relacional ACID

CONSISTÊNCIA: A garantia de que um dado está íntegro durante e após a transação.

Exemplo prático: Antes da transferência, ambas as contas possuem um saldo válido. Após a transferência, a consistência garante que as contas ainda mantenham saldos válidos e que a soma dos saldos de todas as contas não seja afetada. A transferência deve manter as regras e restrições do sistema bancário.

ISOLAÇÃO: Controle sobre os dados de uma transação onde uma transação no banco de dados não pode impactar nos dados das transações em paralelo.

Exemplo prático: Suponha que dois clientes estejam realizando transferências simultaneamente, um da Conta A para a Conta B e outro da Conta B para a Conta A. O isolamento garante que as operações ocorram independentemente. Cada transação é isolada da outra, garantindo que não haja interferência nos saldos ou operações das contas.

EDUCAÇÃO METODISTA

Banco de dados Relacional ACID

Durabilidade: Controle da persistência do dado garantindo que após o "commit" é necessário que os dados estejam 100% íntegro e disponível mesmo em caso de falha.

Exemplo prático: uma vez que a transferência foi concluída com sucesso, os efeitos são permanentes. Isso significa que, mesmo que o sistema experimente uma falha após a transferência, os saldos das contas permanecerão conforme a operação realizada. Os dados da transferência são armazenados de forma durável no banco de dados, mesmo em caso de falhas.

EDUCAÇÃO METODISTA

Banco de Dados Não Relacional

Definição: Refere-se a qualquer banco de dados que não utiliza o modelo relacional. Pode incluir uma variedade de modelos de dados e tecnologias.

Características:

Modelo Flexível

- Suporta diferentes formas de armazenamento e organização de dados, como documentos, grafos, chave-valor e colunas.

Não Segue o Modelo Relacional:

- Não usa tabelas e relacionamentos como em bancos de dados SQL tradicionais.

EDUCAÇÃO METODISTA

Banco de Dados NoSQL

Definição:

- Os bancos de dados NoSQL são um subconjunto dos bancos de dados não relacionais que se destacam pela flexibilidade e escalabilidade horizontal. O termo "NoSQL" originalmente significava "Not Only SQL" para indicar que esses sistemas podem utilizar métodos de consulta além do SQL tradicional.

Origem:

- O termo "NoSQL" foi inicialmente introduzido por Carlo Strozzi em 1998 para descrever bancos de dados que não utilizavam SQL. Em 2009, o conceito foi popularizado em conferências dedicadas a tecnologias de bancos de dados NoSQL, refletindo um crescente interesse e adoção na indústria.

Objetivo:

- Os bancos de dados NoSQL visam atender às necessidades de empresas que lidam com grandes volumes de dados e tráfego intenso, proporcionando alto desempenho e eficiência no processamento e na gestão dessas informações.

EDUCAÇÃO METODISTA

Flexibilidade: Refere-se à capacidade de um sistema ou tecnologia de se adaptar a diferentes tipos de dados e mudanças nos requisitos sem a necessidade de reestruturações significativas. Em bancos de dados, a flexibilidade se traduz na habilidade de acomodar dados não estruturados ou semiestruturados, como documentos JSON, e de permitir ajustes rápidos na estrutura do banco de dados sem afetar a operação geral. Isso permite que as aplicações se adaptem facilmente a novas necessidades e mudanças nos padrões de uso.

Escalabilidade Horizontal: É a capacidade de um sistema para lidar com aumentos na carga de trabalho ou no volume de dados através da adição de mais servidores ou nós ao sistema, em vez de aumentar a capacidade de um único servidor (escalabilidade vertical). Em bancos de dados, a escalabilidade horizontal permite distribuir dados e processos entre múltiplos servidores ou máquinas, garantindo que o sistema possa crescer e manter seu desempenho à medida que o volume de dados e a quantidade de usuários aumentam.

Banco de Dados NoSQL - Características

1. Ausência de SQL: Bancos de dados NoSQL geralmente não utilizam SQL como a linguagem principal de consulta. Alguns podem adotar linguagens de consulta semelhantes, mas não seguem o modelo relacional tradicional.

Alguns bancos de dados NoSQL utilizam uma forma de SQL ou uma linguagem de consulta similar para facilitar a integração com usuários que estão familiarizados com SQL. Aqui estão alguns exemplos:

- **Apache Cassandra:** Utiliza o CQL (Cassandra Query Language), que é semelhante ao SQL, mas adaptado para o modelo de dados baseado em colunas do Cassandra.
- **MongoDB:** Usa o MongoDB Query Language (MQL), que é uma linguagem de consulta própria, mas que pode ser configurada para se assemelhar a SQL em alguns aspectos.
- **Couchbase:** Oferece o N1QL, uma linguagem de consulta que combina elementos de SQL com a estrutura de dados JSON do Couchbase.

Banco de Dados NoSQL - Características

2. Open Source: Muitos bancos de dados NoSQL são de código aberto (*open source*) e foram projetados para operar em *clusters* de computadores, o que facilita a distribuição e o gerenciamento de dados.

3. Flexibilidade de Esquema: Esses bancos de dados não exigem um esquema fixo, permitindo a adição e modificação de dados com facilidade. Isso melhora a escalabilidade e a disponibilidade dos dados, pois novas colunas e tipos de dados podem ser incorporados sem reestruturar o banco.

4. Escalabilidade: Projetados para lidar com grandes volumes de dados, esses sistemas oferecem escalabilidade horizontal. Isso significa que você pode aumentar a capacidade do sistema adicionando mais máquinas, distribuindo a carga e melhorando o desempenho.

Banco de Dados NoSQL - Características

5. Distribuição e Replicação: Bancos de dados distribuídos podem ser armazenados em vários servidores ou computadores. Isso melhora o desempenho e permite a replicação dos dados para maior escalabilidade e recuperação rápida em caso de falhas.

6. Modelos de Dados Heterogêneos: Esses bancos de dados oferecem esquemas flexíveis e não padronizados, suportando diversos modelos de dados, como documentos, grafos e colunas. Dados heterogêneos referem-se a dados que têm formatos e estruturas variadas, sem um padrão único. Essa flexibilidade permite adaptar a estrutura dos dados às necessidades específicas das aplicações, acomodando diferentes tipos de informações em um único sistema.

7. Simples Acesso e APIs: O acesso aos dados é facilitado por APIs (Interfaces de Programação de Aplicações), que são conjuntos de regras e definições que permitem que diferentes softwares se comuniquem entre si. As APIs são geralmente baseadas em chaves, tornando o desenvolvimento de aplicações web mais direto e eficiente, com integração fácil ao banco de dados.

Banco de Dados NoSQL - Características

8. Limitações: Não suportam junções ou relacionamentos entre dados e integridade referêncial, e nem sempre conseguem se manter consistentes.

Exemplos de uso:

Google: processa até 20 *petabytes* de dados por dia utilizando o *BigTable*, um exemplo de banco de dados NoSQL.

Spotify: Emprega o Cassandra e o *Elasticsearch* para armazenar e consultar grandes volumes de dados de músicas, playlists e preferências dos usuários. Isso permite uma busca eficiente e a personalização das recomendações musicais.

EDUCAÇÃO METODISTA

Banco de Dados NoSQL - Características

- Os bancos de dados NoSQL podem ser instalados localmente em servidores físicos ou virtuais, ou podem ser implantados na nuvem, aproveitando a escalabilidade e a flexibilidade dos ambientes de nuvem.
- Por exemplo, **MongoDB Atlas** é uma solução de banco de dados NoSQL que oferece uma plataforma gerenciada na nuvem, permitindo que as empresas escalem automaticamente e se beneficiem de recursos avançados sem a necessidade de configurar e manter servidores físicos.

Banco de Dados NoSQL - Vantagens

Flexibilidade de Implantação: Os bancos de dados NoSQL oferecem flexibilidade para serem instalados localmente em servidores físicos ou virtuais, ou para serem hospedados em plataformas de nuvem como AWS, Azure e Google Cloud Platform (GCP). Isso permite que você escolha a melhor opção de acordo com suas necessidades de infraestrutura e escalabilidade.

Desempenho: São projetados para oferecer alta performance em leitura e escrita, lidando rapidamente com grandes volumes de dados. Isso assegura um desempenho eficiente mesmo sob alta carga de trabalho.

Escalabilidade Horizontal: Facilitam o crescimento do sistema ao permitir a adição de novos servidores. Isso distribui a carga de trabalho de forma equilibrada, mantendo a eficiência e a performance à medida que a demanda aumenta.

Modelagem de Dados Flexível: Não exigem um esquema fixo para armazenar dados, permitindo a integração e adição fácil de novos tipos de dados. Isso proporciona maior adaptabilidade e agilidade para atender às mudanças nas necessidades das aplicações.

Alta Disponibilidade: Garantem alta disponibilidade e redundância por meio de replicação e distribuição geográfica dos dados. Isso assegura que o sistema permaneça operacional e acessível, mesmo em caso de falhas ou interrupções.

Banco de Dados NoSQL - Vantagens

Gerenciamento de Dados Não Estruturados: Bancos de dados NoSQL são ideais para armazenar e gerenciar dados não estruturados, como documentos, imagens, vídeos e informações de redes sociais, oferecendo flexibilidade para lidar com diferentes formatos de dados.

Variedade de Modelos de Dados: Eles suportam diversos modelos de dados, incluindo chave-valor, documento, coluna e grafo, permitindo a escolha do modelo que melhor se adapta às necessidades específicas da aplicação.

Redução de Custos: Muitos bancos de dados NoSQL são de código aberto (*open source*) e projetados para escalar horizontalmente em *hardware* comum, o que pode reduzir significativamente os custos de infraestrutura e operação.

Facilidade de Desenvolvimento: Acesso facilitado por meio de APIs simples e mecanismos baseados em chave tornam o desenvolvimento de aplicações mais rápido e eficiente, permitindo integração ágil com o banco de dados.

Manutenção e Operação Simplificadas: A ausência de um esquema fixo e a capacidade de adaptar-se a mudanças dinâmicas reduzem a complexidade da manutenção, simplificando a operação do sistema e tornando-o mais ágil para ajustes e atualizações.

Por que NoSQL?

Atualmente, as empresas estão cada vez mais adotando bancos de dados NoSQL para uma ampla gama de casos de uso. Essa tendência é impulsionada por quatro forças inter-relacionadas:

- **Big Data:** A crescente quantidade e complexidade de dados gerados exige soluções que possam armazenar e processar grandes volumes de informações de forma eficiente.
- **Big User:** O aumento no número de usuários e interações digitais demanda sistemas que possam escalar e manter o desempenho mesmo com altas cargas de trabalho.
- **Internet das Coisas (IoT):** A proliferação de dispositivos conectados gera uma grande quantidade de dados não estruturados que precisam ser gerenciados e analisados em tempo real.
- **Computação em Nuvem (Cloud Computing):** A flexibilidade e escalabilidade oferecidas pela nuvem permitem que os bancos de dados NoSQL sejam facilmente escalados e ajustados conforme a demanda, reduzindo custos e aumentando a eficiência.

Entendendo a Diferença entre Bancos de Dados Relacionais e NoSQL

Estrutura e Linguagem de Consulta:

- **Bancos de Dados Relacionais (SQL):** Utilizam a linguagem SQL (*Structured Query Language*) para consultas e manipulação de dados. Os dados são organizados em tabelas com linhas e colunas, e as relações entre dados são estabelecidas por meio de junções. Isso possibilita consultas complexas e manipulação estruturada de dados.
- **Bancos de Dados NoSQL:** Não seguem o modelo relacional e adotam formatos variados para armazenar dados, como:
  - **Documento:** Armazena dados em documentos, geralmente no formato JSON (JavaScript Object Notation) ou BSON (Binary JSON).
  - **Chave-Valor:** Utiliza pares chave-valor para armazenar e recuperar dados.
  - **Colunas:** Organiza dados em colunas em vez de linhas, facilitando consultas em grandes volumes de dados.
  - **Grafo:** Focado em armazenar e consultar dados que representam redes e relacionamentos complexos.

Aula 2

Conceitos básicos dos bancos de dados não relacionais
e
Tipos de Banco de dados NoSQL

EDUCAÇÃO METODISTA

Banco de dados NoSQL BASE

À medida que os bancos de dados NoSQL se tornaram mais populares, surgiu a necessidade de modelos que refletissem a flexibilidade e a escalabilidade que esses sistemas oferecem. Um desses modelos é o **BASE**, que representa uma abordagem alternativa à consistência estrita dos bancos de dados relacionais. BASE é um acrônimo que descreve três propriedades fundamentais:

- **Basically Available (Disponibilidade Básica):** Em vez de garantir a consistência imediata, os bancos de dados BASE priorizam a disponibilidade dos dados, assegurando que eles estejam acessíveis ao espalhar e replicar informações por vários nós do cluster. Isso significa que, mesmo em caso de falhas, o sistema permanece operacional e os dados continuam disponíveis.

EDUCAÇÃO METODISTA

Banco de dados NoSQL BASE

- **Soft State (Estado Suave):** Com a ausência de uma consistência imediata, o estado dos dados pode mudar ao longo do tempo. O modelo BASE permite que os dados sejam modificados e atualizados de forma assíncrona, delegando a responsabilidade de gerenciar a consistência final para os desenvolvedores e aplicações.
- **Eventually Consistent (Consistência Eventual):** Apesar de não impor a consistência imediata, o modelo BASE garante que, eventualmente, todos os dados se tornem consistentes. Isso significa que, após um período de tempo, todas as atualizações e modificações serão propagadas por todo o sistema, embora as leituras possam refletir informações desatualizadas até que a consistência seja alcançada.

Com esses princípios, o modelo BASE oferece uma abordagem prática para a manipulação de grandes volumes de dados em sistemas distribuídos, equilibrando a necessidade de alta disponibilidade e escalabilidade com a eventual consistência dos dados.

Exemplo de Aplicação das Propriedades BASE:

Durante eventos de alta demanda, como a "Black Friday", sistemas de comércio eletrônico enfrentam um aumento significativo no tráfego, o que pode sobrecarregar os servidores e causar latências elevadas. Neste cenário, a aplicação das propriedades BASE pode ser extremamente benéfica:

- **Basically Available (Disponibilidade Básica):** Para garantir que o sistema continue operando mesmo sob alta carga, os dados são replicados e distribuídos entre vários servidores. Isso assegura que os usuários possam acessar e realizar compras sem interrupções, mesmo se alguns servidores estiverem sobrecarregados ou temporariamente fora de serviço.
- **Soft State (Estado Suave):** Dado que o sistema pode lidar com uma grande quantidade de atualizações e transações simultâneas, o estado dos dados pode não ser imediatamente consistente. Por exemplo, o estoque de um produto pode não refletir as mudanças instantaneamente. No entanto, a flexibilidade do modelo permite que as atualizações se propaguem gradualmente.
- **Eventually Consistent (Consistência Eventual):** Embora as informações sobre o estoque possam não estar totalmente sincronizadas em tempo real, o sistema assegura que, eventualmente, todos os dados se tornarão consistentes. Isso significa que, após a carga de trabalho diminuir, todas as transações e atualizações serão consolidadas e refletidas de forma precisa em todos os pontos de acesso.

Diferença entre ACID e BASE

A principal diferença entre os modelos ACID e BASE está na forma como abordam a consistência e a disponibilidade dos dados:

- **ACID (Atomicidade, Consistência, Isolamento e Durabilidade):** Este modelo prioriza uma consistência rígida e a integridade dos dados. Em sistemas que seguem o ACID, cada transação deve ser totalmente concluída para garantir que os dados permaneçam consistentes e corretos, mesmo em caso de falhas. Esse enfoque assegura a segurança dos dados e é fundamental para aplicações onde a precisão e a confiabilidade são cruciais.

- **BASE (Basically Available, Soft State e Eventually Consistent):** Ao contrário do ACID, o modelo BASE opta por sacrificar a consistência imediata em favor de alta disponibilidade e escalabilidade. Em sistemas BASE, a consistência dos dados é alcançada de forma eventual, permitindo que o sistema se adapte e se expanda para lidar com grandes volumes de dados e alta carga de tráfego. A flexibilidade do BASE é especialmente útil em ambientes distribuídos e de alta demanda.

Diferença entre ACID e BASE

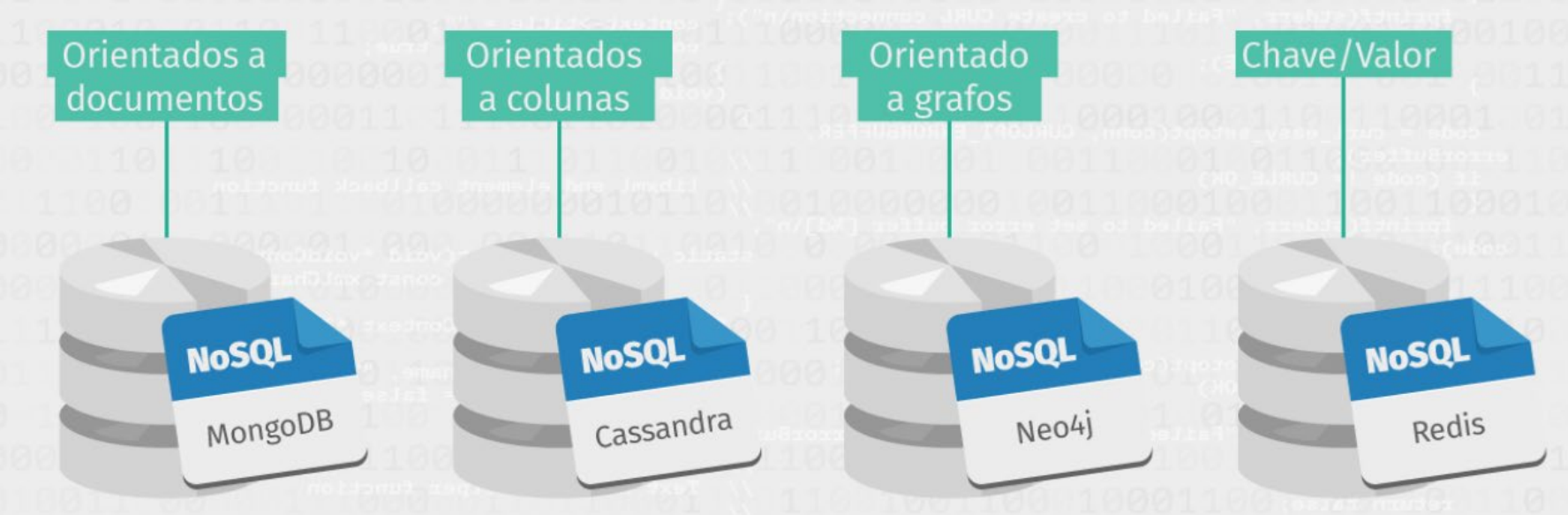
CASOS DE USO:

- **ACID:** É comumente aplicado em bancos de dados relacionais, onde a integridade dos dados e a realização de transações complexas são essenciais, como em sistemas financeiros e de gestão de inventário.

- **BASE:** É frequentemente utilizado por bancos de dados NoSQL e sistemas distribuídos, onde a escalabilidade e a disponibilidade são mais importantes do que a consistência imediata, como em plataformas de redes sociais e e-commerce de alta escala.

Ambos os modelos têm seus próprios casos de uso apropriados, dependendo das necessidades específicas e das prioridades do sistema em questão.

Tipos de bancos de dados NoSQL



Banco de dados NoSQL, apesar de serem classificados como tal, podem ser distintos uns dos outros. É por esse motivo que eles são classificados de acordo com a forma de os dados serem armazenados e/ou consultados.

EDUCAÇÃO METODISTA

Banco de Dados Orientados a Colunas

- Banco de dados colunares, ao contrário de banco de dados relacionais, armazenam os registros em disco agrupados por colunas.
- É um tipo de que armazena os dados em linha particulares de tabelas no disco.
- O banco de dados colunar é otimizado para recuperação rápida de colunas de dados.
- O armazenamento orientado a colunas pra tabelas com banco de dados é um fator muito importante para performance em consulta analítica, pois ele reduz expressivamente a entrada e saída em disco, diminuindo a quantidade de dados que você precisa carregar no disco.

EDUCAÇÃO METODISTA

Banco de Dados Orientados a Colunas

- Conceitos: keyspace (≡ BD), column family (≡ tabela) e um conjunto de colunas (≡ registro)

– Uma coluna possui um nome e um valor

– Um conjunto de colunas é acessado por uma chave

– Itens de dados (“registros”) podem ter colunas diferentes

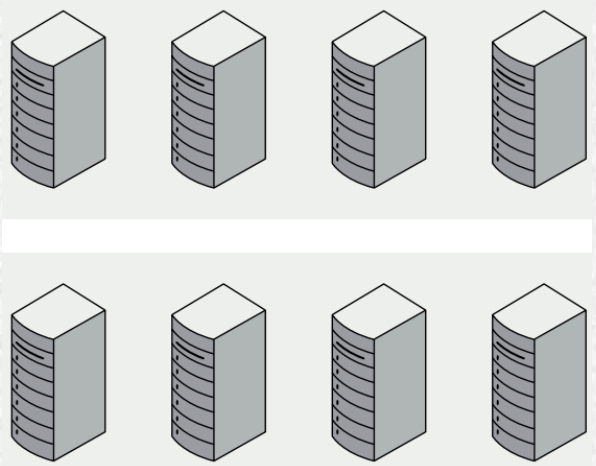
– Suporte a colunas multivaloradas e super-colunas

Usuário				
ID 1Ø1	email	nome	tel	
	joao@test.com	João	4556-2233	
ID 1Ø2	email	nome	tel	tel2
	maria@test.com	Maria	444-555	4544-4444
ID 1Ø3	nome			
	Pedro			

EDUCAÇÃO METODISTA

Banco de Dados Orientados a Colunas
Características e funcionalidades

- A escalabilidade horizontal consiste em aumentar o processamento pelo aumento do número de máquinas utilizadas para o processamento (LÓSCIO; OLIVEIRA; PONTES, 2011). Na Figura 2, há um exemplo de escalabilidade horizontal, no qual inicialmente existe uma máquina, depois duas até chegar a oito máquinas, aumentando, assim, o armazenamento e o processamento.



EDUCAÇÃO METODISTA

Banco de Dados Orientados a Colunas
Características e funcionalidades

- Ausência de esquema (schema-free) ou esquema flexível, que, em bancos de dados NoSQL, pode ser total ou parcial. Isso facilita a alta escalabilidade e a disponibilidade, embora não garanta a integridade (KOKAY, 2012).

- Já o suporte nativo à replicação é um recurso que fornece sincronização e integração dos dados para possibilitar um gerenciamento eficiente no crescimento dos dados. Entre as vantagens da replicação de dados, temos a integração de dados simplificada e em tempo real para aplicações de big data, a integração ágil e de baixo impacto dos dados e a alta disponibilidade e proteção contínua dos dados (REPLICAÇÃO..., 2020).

Banco de Dados Orientados a Colunas
Características e funcionalidades

- Outro ponto importante é a API (interface de programação de aplicações) simples para acessar o banco de dados, um conjunto de definições, protocolos e ferramentas utilizados para integrar serviços e aplicações (INTRODUÇÃO..., 2020). Alguns bancos NoSQL de família de colunas apresentam API para facilitar a manipulação e a busca de informações.

- Outra característica dos bancos NoSQL é que nem sempre a consistência é garantida, ou seja, sua consistência é eventual: somente se pode manter duas das três propriedades do teorema CAP (consistency, availability e partition tolerance) — consistência, disponibilidade e tolerância à partição —, por exemplo, o Cassandra garante as duas últimas propriedades (WESTOBY, 2019).

Exemplo usando um banco de dados orientado a colunas, como o Apache Cassandra.

Exemplo: Armazenamento de Dados de Usuários

- Suponha que estamos criando um sistema para armazenar dados de usuários, como nome, idade, e-mail e cidade. Vamos modelar esses dados em um banco de dados orientado a colunas.

- Keyspace (Banco de Dados): Criamos um keyspace chamado "meu_app" para armazenar todos os dados relacionados ao nosso aplicativo.

- Column Family (Tabela): Dentro do keyspace "meu_app", criamos uma column family chamada "usuarios" para armazenar os dados dos usuários. Essa column family terá uma estrutura de colunas comum para todos os registros.

- Conjunto de Colunas (Registro): Cada usuário é representado por um conjunto de colunas (registro) na column family "usuarios". A chave de linha é um identificador único para cada usuário.

Exemplo usando um banco de dados orientado a colunas, como o Apache Cassandra.

```
Keyspace: meu_app

Column Family: usuarios

Chave de Linha: 1
  nome: João
  idade: 30
  email: joao@example.com
  cidade: São Paulo

Chave de Linha: 2
  nome: Maria
  idade: 25
  email: maria@example.com
  cidade: Rio de Janeiro
```

Banco de Dados
Chave-Valor (Key-value)

- O banco de dados de chave-valor utiliza dados não relacionais (NoSQL) e o conceito de chave-valor para armazenar dados. Segundo Fernandes (2017), ao armazenar as informações, estas são obtidas por meio do acesso da chave.

- Por exemplo, quando se deseja ler um determinado conteúdo em um livro, o que se faz é ir até o índice e procurar o assunto desejado. O mesmo ocorre com o conceito de chave-valor. Para Oliveira (2014), essa forma de armazenamento é livre de estrutura, ou seja, não necessita atender a uma determinada estrutura, ao contrário de um banco de dados relacional, que tem as suas informações armazenadas em tabelas especificadas.

Banco de Dados
Chave-Valor (Key-value)

- O banco de dados de chave-valor utiliza dados não relacionais (NoSQL) e o conceito de chave-valor para armazenar dados. Segundo Fernandes (2017), ao armazenar as informações, estas são obtidas por meio do acesso da chave.

- Por exemplo, quando se deseja ler um determinado conteúdo em um livro, o que se faz é ir até o índice e procurar o assunto desejado. O mesmo ocorre com o conceito de chave-valor. Para Oliveira (2014), essa forma de armazenamento é livre de estrutura, ou seja, não necessita atender a uma determinada estrutura, ao contrário de um banco de dados relacional, que tem as suas informações armazenadas em tabelas especificadas.

Banco de Dados
Chave-Valor (Key-value)

- Uma das funcionalidades dos bancos de dados de chave-valor é a existência de APIs (Application Programming Interface; ou Interface de Programação de Aplicações, em português), desenvolvidas para facilitar a busca de informações.

- O objetivo de uma API é que os usuários não necessariamente precisem saber como é o funcionamento dos seus serviços. Em geral, as empresas desenvolvem APIs para que outras aplicações consumam seus serviços, a exemplo do Google em relação ao Google Maps. O mesmo ocorre com os bancos de dados de chave-valor, a exemplo do Redis e do DynamoDB.

Banco de Dados
Chave-Valor (Key-value)

- Modelo simples, similar a uma estrutura de indexação.

- Não suporta
  - Definição de esquemas
  - Relacionamento entre dados
  - Linguagem de consulta

EDUCAÇÃO METODISTA

Exemplo de dados armazenados com chave-valor

Exemplo: Armazenamento de Dados de Produtos em um E-Commerce

Suponha que estamos criando um sistema para um e-commerce e precisamos armazenar informações sobre os produtos disponíveis, como nome, preço, descrição e quantidade em estoque. Vamos modelar esses dados em um Banco de Dados Chave-Valor.

Nesse exemplo, cada produto será representado como um valor e será associado a uma chave única, que é o identificador do produto.

```
Chave: produto:1
Valor: {
  "nome": "Camiseta Branca",
  "preco": 39.99,
  "descricao": "Camiseta branca de algodão",
  "quantidade_estoque": 100
}

Chave: produto:2
Valor: {
  "nome": "Calça Jeans",
  "preco": 79.99,
  "descricao": "Calça jeans azul",
  "quantidade_estoque": 50
}
```

Banco de Dados Orientados Documento

- Trabalha com dados não relacionais e os armazena como documentos estruturados, geralmente nos formatos XML (eXtensible Markup Language; ou Linguagem de Marcação Extensível, em português) ou JSON (JavaScript Object Notation; ou Notação de Objetos JavaScript, em português).

- Podem implementar transações ACID (atomicidade, consistência, isolamento e durabilidade) ou outras características de um RDBMS (Relational Database Management System; ou Sistema de Gerenciamento de Bancos de Dados Relacionais, em português) tradicional, embora os bancos de dados orientados a documentos dominantes forneçam um suporte transacional relativamente modesto.

Banco de Dados Orientados Documento

Modelo adequado à representação de objetos complexos

– Um objeto (“documento”) possui uma chave e um conjunto de atributos;

– Atributos podem ter domínios atômicos ou complexos (listas, tuplas, conjuntos).

- APIs proprietárias e/ou linguagens de consulta simples
- Não suporta relacionamentos entre dados
- Falta de padronização

Banco de Dados Orientados Documento

- Algumas das vantagens do uso de bancos de dados orientados a documentos são: ganho de flexibilidade, disponibilidade e linguagem de consulta simples e performática. Como desvantagem, em alguns casos, há a perda da consistência.

- O MongoDB é um exemplo de banco de dados orientado a documentos, o qual possui código aberto, alto desempenho e é escrito em linguagem C++, além de não possuir esquemas, como em um contexto relacional.

EDUCAÇÃO METODISTA

Banco de Dados Orientados Documento

Exemplo: Armazenamento de Dados de Livros em uma Biblioteca

Suponha que estamos criando um sistema para uma biblioteca e precisamos armazenar informações sobre os livros disponíveis, como título, autor, ano de publicação e gênero. Vamos modelar esses dados em um Banco de Dados Orientado a Documento, como o MongoDB.

Nesse exemplo, cada livro será representado como um documento JSON, que conterá os campos com as informações do livro.

json

```
{
  "_id": 1,
  "titulo": "Dom Casmurro",
  "autor": "Machado de Assis",
  "ano_publicacao": 1899,
  "genero": "Romance"
}
```

json

```
{
  "_id": 2,
  "titulo": "O Príncipe",
  "autor": "Maquiavel",
  "ano_publicacao": 1532,
  "genero": "Política"
}
```

json

```
{
  "_id": 3,
  "titulo": "1984",
  "autor": "George Orwell",
  "ano_publicacao": 1949,
  "genero": "Ficção Distópica"
}
```

Comparação entre um banco de dados orientado a documentos e um banco de dados relacional

Baseado em documentos autocontidos, os quais contêm todas as informações necessárias.	Tabelas com metadados internos, de acordo com cada gerenciador de banco de dados.
Não há necessidade de um esquema fixo.	Baseado em ligações de chaves primária e estrangeira.
Criação de um novo campo dentro do banco de dados sem que isso afete outros registros.	A inserção de novos campos precisará respeitar as chaves primárias e estrangeiras.
Os dados podem ser repetidos em diversos documentos.	Entrada única para cada registro, já que possuiu uma chave primária rígida.
Possuiu um UUID.	Possuiu uma chave primária simples ou composta.
Não suporta as junções, porém permite realizar visualizações de união de documentos.	Suporta junções através das chaves primárias e estrangeiras.
Os documentos não precisam armazenar campos vazios.	Ainda que um campo não receba nenhum valor, ele será criado.

EDUCAÇÃO METODISTA

Tipo de documento

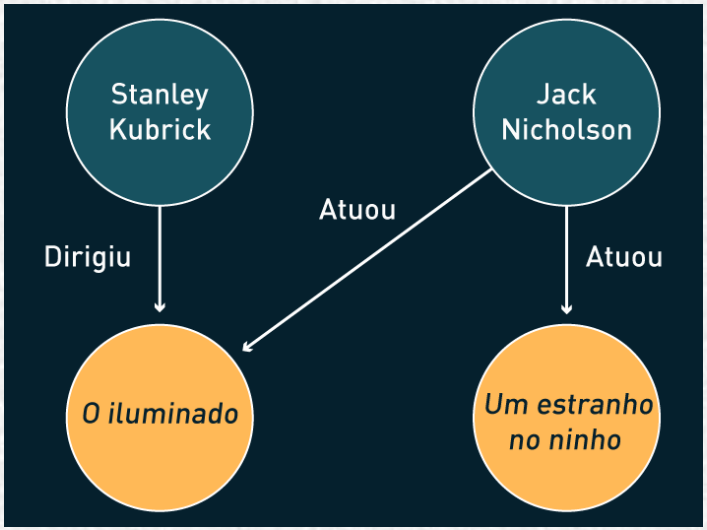
Modelo de dados do Cloud Firestore

- O Cloud *Firestore* é um banco de dados NoSQL orientado a documentos. Ao contrário de um banco de dados SQL, não há tabelas nem linhas. Em vez disso, os dados são armazenados em documentos, que são organizados em coleções.

Saiba mais por meio do *link:*
https://firebase.google.com/docs/firestore/data-model?hl=pt-br

Banco de dados orientado a Grafos

- Os **bancos de dados orientados a grafos** implementam um sistema que armazena os dados em estruturas de grafos e provê consultas mais semânticas, por meio de dados interconectados por nós e arestas. Uma das vantagens desse tipo de banco é a flexibilidade do modelo dos dados, oriundo dos bancos de dados NoSQL (não relacional). Ao contrário dos bancos relacionais, em que é necessária a criação ou a alteração de tabelas existentes para adicionar novos tipos de dados, em um banco orientado a grafos, é permitida a adição de novos tipos de vértices e arestas a um grafo existente sem precisar fazer modificações no esquema de dados.



EDUCAÇÃO METODISTA

Banco de dados orientado a Grafos

- São bancos mais complexos.
- Armazena Objetos e não registros.
- As buscas são realizadas pela navegação nos objetos.
- Possui três componentes básicos:

Nós (vértices)	Contém os pares chave–valor com as informações de uma entidade.
Subgrafo	É um grafo dentro de um grafo maior.
Relacionamentos (arestas)	Realiza a conexão de dois nós, de modo que sempre haverá um nó de início e um de fim, os quais têm uma única direção e tipo.
Propriedades	Propriedades dos nós e relacionamentos.

EDUCAÇÃO METODISTA

Banco de dados orientado a Grafos

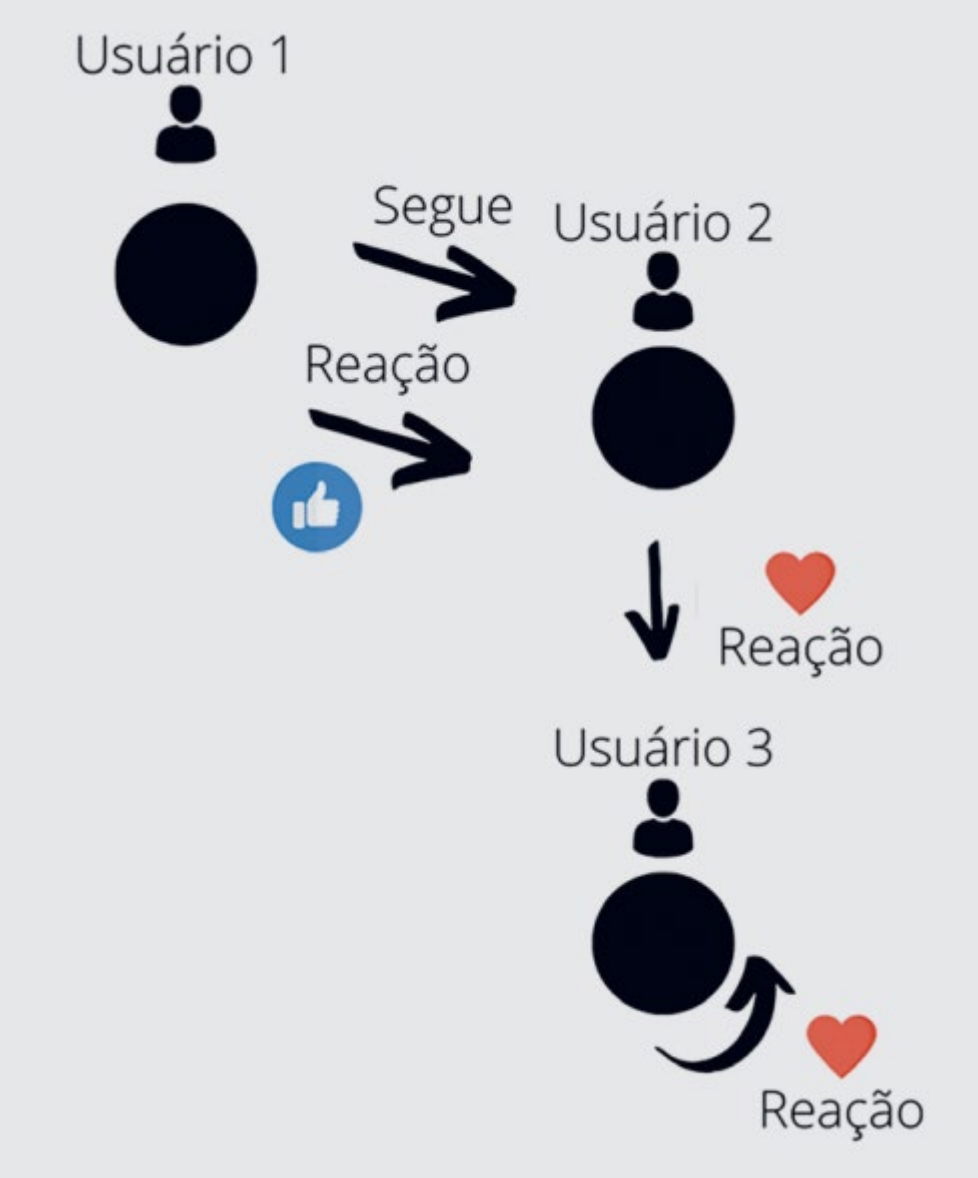
Exemplo: Rede Social

Suponha que estamos criando uma aplicação de rede social e precisamos armazenar informações sobre usuários e suas conexões. Vamos modelar esses dados em um Banco de Dados Orientado a Grafos, como o Neo4j.

plaintext

```
Nó (Usuário):
- ID: 1
- Nome: João
- Idade: 30

Nó (Usuário):
- ID: 2
- Nome: Maria
- Idade: 25

Nó (Usuário):
- ID: 3
- Nome: Pedro
- Idade: 28

Aresta (Conexão - Amizade):
- De: 1 (João)
- Para: 2 (Maria)
- Tipo: Amizade

Aresta (Conexão - Segue):
- De: 1 (João)
- Para: 3 (Pedro)
- Tipo: Segue
```

Banco de dados orientado a Grafos



Banco de dados orientado a Pesquisas

O tipo Pesquisa é um modelo construído especificamente para indexação, agregação e pesquisa de registros em dados semi estruturados. Esse modelo é pensado ainda mais para alta performance, baixa latência e análise de dados praticamente em tempo real.

Banco de dados orientado a Pesquisas

Exemplo: Catálogo de Produtos em uma Loja Online

Suponha que estamos criando uma loja *online* e precisamos armazenar informações sobre os produtos disponíveis para que os clientes possam pesquisar e encontrar os itens desejados. Vamos modelar esses dados em um Banco de Dados Orientado a Pesquisas, como o Elasticsearch.

Nesse exemplo, cada produto será representado como um documento JSON, contendo os campos relevantes para a pesquisa e os detalhes do produto.

Banco de dados orientado a Pesquisas

Exemplo: Catálogo de Produtos em uma Loja Online

json

```json
{
  "id": 1,
  "nome": "Camiseta Branca",
  "descricao": "Camiseta branca de algodão",
  "preco": 39.99,
  "marca": "Marca A",
  "categoria": "Vestuário",
  "tamanho": ["P", "M", "G"],
  "cor": "Branca"
}
```

json

```json
{
  "id": 2,
  "nome": "Calça Jeans",
  "descricao": "Calça jeans azul",
  "preco": 79.99,
  "marca": "Marca B",
  "categoria": "Vestuário",
  "tamanho": ["36", "38", "40", "42"],
  "cor": "Azul"
}
```

DB-ENGINES

DB-Engines Ranking

https://db-engines.com/en/ranking

EDUCAÇÃO METODISTA

Exemplos por tipo

EDUCAÇÃO METODISTA

Exemplos por tipo

Exemplos por tipo

EDUCAÇÃO METODISTA

Principais utilizadores do NoSQL

• Google	BigTable
• Amazon	Dynamo
• Yahoo	Hadoop
• Facebook	Cassandra
• Digg	Cassandra
• Twitter	Cassandra
• IBM	Cassandra
• Netflix	Cassandra
• LinkedIn	Voldemort
• Engine Yard	MongoDB

EDUCAÇÃO METODISTA

Atividade 2 de Produção Textual: Banco de Dados NoSQL e Sistemas de Armazenamento Big Data

Objetivo: Compreender as características diferenciadas dos bancos de dados NoSQL e suas aplicações e vantagens.

Instruções:

1. Leitura Obrigatória: Leia o Capítulo 24: "Banco de Dados NoSQL e Sistemas de Armazenamento Big Data" do livro *Sistemas de Banco de Dados*, da página 795 até a página 818. Link para a leitura: Clique aqui

2. Tarefa:

Após a leitura, produza um texto explicativo abordando os seguintes pontos:

a. **Introdução aos Bancos de Dados NoSQL:**
- Definição e características principais.
- Comparação com bancos de dados relacionais tradicionais.

c. **Aplicações e Vantagens dos Bancos de Dados NoSQL:**
- Casos de uso comuns.
- Vantagens em relação aos bancos de dados relacionais.

1.Formatação do Texto:

1. Utilize as normas da ABNT para a formatação do texto.
2. Inclua introdução, desenvolvimento e conclusão.
3. Utilize citações e referências conforme necessário.

2.Critérios de Avaliação:

1. Clareza e organização do texto.
2. Coerência e coesão na apresentação das ideias.
3. Adequação às normas da ABNT.
4. Originalidade e profundidade da análise.
5. Uso correto das referências bibliográficas.

Aula 03

EDUCAÇÃO METODISTA

INTRODUÇÃO ao TEOREMA de CAP

O **Teorema de CAP** afirma que em um sistema distribuído é impossível garantir simultaneamente as três propriedades:

- **Consistência** - todos os nós veem os mesmos dados ao mesmo tempo
- **Disponibilidade** - o sistema sempre responde às solicitações)
- **Tolerância a Particionamento** - o sistema continua funcionando mesmo se houver falhas na comunicação entre os nós.

Portanto, em qualquer sistema distribuído, é preciso escolher duas dessas propriedades, mas nunca as três ao mesmo tempo.

EDUCAÇÃO METODISTA

TEOREMA DE CAP

Consistência

- **Consistência**, no contexto do Teorema de CAP, significa que todas as réplicas de dados em um sistema distribuído terão os mesmos valores após uma operação de gravação.
- Em outras palavras, se você escreve um dado em um nó, esse dado será imediatamente refletido em todos os outros nós do sistema.
- Essa é uma propriedade importante para garantir que os usuários vejam as mesmas informações, independentemente de qual nó eles estejam acessando.
- No entanto, garantir essa consistência pode impactar outras propriedades, como a disponibilidade do sistema.

TEOREMA DE CAP

Disponibilidade

- A **Disponibilidade**, por outro lado, refere-se à capacidade do sistema de responder a todas as solicitações, mesmo que nem todos os nós estejam operando corretamente.
- Isso significa que, em um sistema altamente disponível, sempre que um usuário fizer uma solicitação, ele receberá uma resposta, seja ela bem-sucedida ou com falha.
- A grande vantagem aqui é que o sistema continua funcionando mesmo diante de problemas.
- No entanto, para manter essa disponibilidade, podemos ter que comprometer a consistência dos dados.

Tolerância a Particionamento

TEOREMA DE CAP

- **Tolerância a Particionamento** é a capacidade do sistema de continuar operando mesmo que haja uma falha de comunicação entre diferentes partes do sistema.
- Imagine que uma rede se divida em duas, mas o sistema deve continuar funcionando e processando solicitações em ambos os lados da partição.
- Essa propriedade é essencial para sistemas distribuídos, pois falhas de rede são inevitáveis.
- No entanto, ao garantir a tolerância a particionamento, pode ser necessário sacrificar a consistência ou a disponibilidade em determinadas situações.

TEOREMA DE CAP

Tolerância a Partição

Para isto, a arquitetura do sistema precisa ser concebida de maneira a lidar com possíveis interrupções ou falhas na rede, de forma a manter a operação em condições adversas. Isso pode incluir a implementação de mecanismos de replicação de dados, balanceamento de carga, detecção de falhas e outras estratégias que permitam que o sistema continue a funcionar de maneira confiável, mesmo em situações onde a conectividade entre os componentes do sistema é interrompida.

Teorema CAP

C – Consistência
A – Disponibilidade
P - Tolerância a Partição

CAp (Consistência + Disponibilidade) – são sistemas que garantem a consistência dos dados e possuem alta disponibilidade, mas não suportam falhas de partição. Em caso de falha da rede ou de grande latência, elas não consegue responder a todos os pedidos. (Existem formas de resolver este problema, mas nenhum é perfeito).

cAP (Disponibilidade + Tolerância a Particionamento) – estes são sistemas altamente tolerantes a falhas (assumindo que existem um grande número de servidores a suportar o sistema) e elas fornecem disponibilidade também, sendo que normalmente seguem o modelo de consistência eventual. Podem possuir possíveis inconsistências temporárias. Ex. Cassandra, DynamoDB.

CaP (Consistência + Tolerância a Particionamento) – esta não é uma opção atrativa. O sistema não vai dar garantias de estar sempre disponível. Um exemplo é um sistema em que um nodo falha e os outros não podem responder aos pedidos. Sacrifica a Disponibilidade durante uma partição. Ex. Hbase, MongoDB, dentre outros.

Sistemas CA

C – Consistência
A – Disponibilidade
P - Tolerância a Partição

- Os sistemas com consistência forte e alta disponibilidade não sabem lidar com a possível falha de uma partição.
- Caso ocorra, o sistema inteiro pode ficar indisponível até o membro do cluster voltar.

Imagine um sistema de banco de dados distribuído usado por um banco para processar transações financeiras. Nesse cenário, a consistência é crucial, já que a integridade dos dados financeiros é fundamental. Se um cliente fizer uma transação que envolva a retirada de dinheiro de sua conta, é importante que essa transação seja refletida imediatamente em todas as partes do sistema para garantir que o saldo correto esteja disponível.

Ao mesmo tempo, a disponibilidade também é crucial. Os clientes precisam acessar suas contas e realizar transações a qualquer momento. Se o sistema estiver inacessível, os clientes não poderão acessar seus fundos ou efetuar pagamentos, o que pode causar grandes problemas.

Sistemas CP

C – Consistência
A – Disponibilidade
P - Tolerância a Partição

- Para sistemas que precisam da consistência forte e tolerância a particionamento é necessário abrir a mão da disponibilidade (um pouco).
- Exemplos são BigTable, Hbase ou MongoDB entre vários outros.

Imagine um sistema de gerenciamento de estoque utilizado por uma cadeia de supermercados. Nesse sistema, é fundamental que a quantidade de produtos em estoque seja consistente e precisa em todas as lojas. Se um cliente comprar um item em uma loja, é importante que essa informação seja refletida imediatamente no estoque de todas as outras lojas, para evitar vendas duplicadas ou falta de produtos.

Ao mesmo tempo, a tolerância a partições é importante, uma vez que a rede pode enfrentar falhas temporárias ou perda de conectividade entre diferentes locais das lojas. Isso pode ocorrer devido a problemas de rede, manutenção ou outros eventos imprevistos.

Sistemas AP

C – Consistência
A – Disponibilidade
P - Tolerância a Partição

- Há sistemas que jamais podem ficar offline, portanto não desejam sacrificar a disponibilidade. Para ter alta disponibilidade mesmo com tolerância a particionamento é preciso prejudicar a consistência.
- Exemplos de Bancos são: Cassandra, MongoDB e Voldemort.

Considere um sistema de streaming de vídeo *online*, como uma plataforma de transmissão ao vivo de jogos ou eventos. Nesse tipo de sistema, é crucial que os espectadores possam assistir aos conteúdos sem interrupções, especialmente em eventos ao vivo que estão ocorrendo em tempo real.

A disponibilidade é uma prioridade nesse cenário, uma vez que os espectadores precisam acessar e assistir ao conteúdo a qualquer momento. Se o sistema ficar indisponível durante um evento ao vivo, os espectadores podem ficar insatisfeitos e perderem a oportunidade de acompanhar o conteúdo que desejam.

Ao mesmo tempo, a tolerância a partições é importante para lidar com possíveis falhas de rede ou interrupções temporárias na conectividade. Eventos ao vivo podem atrair um grande número de espectadores, o que pode sobrecarregar temporariamente a infraestrutura de rede.

Impacto do Teorema de CAP na Escolha do Banco de Dados

- Entender o **Teorema de CAP** é crucial para tomar decisões arquiteturais em sistemas distribuídos.
- Dependendo das necessidades do seu sistema, você precisará escolher qual das propriedades vai priorizar.
- Se sua aplicação não pode tolerar inconsistências, você pode optar por um sistema que priorize a Consistência.
- Se a disponibilidade é mais crítica, você escolherá um sistema que garanta resposta mesmo diante de falhas.
- Essas decisões têm um impacto direto no tipo de banco de dados e na arquitetura que você vai adotar.

EDUCAÇÃO METODISTA

Impacto do Teorema de CAP na Escolha do Banco de Dados

- Para concluir, o **Teorema de CAP** nos ensina que, em sistemas distribuídos, precisamos fazer escolhas difíceis sobre quais propriedades priorizar.
- Não podemos ter tudo, e as decisões que tomamos têm impactos diretos na experiência do usuário e na confiabilidade do sistema.
- Compreender essas limitações é essencial para projetar sistemas eficazes e escolher a tecnologia certa para cada caso.

{ "name": "Mongo", "type": "DB", "authors": ["João Pedro Castro", "Cristina Ciferri"]
}

Introdução

- O **MongoDB** é um banco de dados NoSQL conhecido por sua flexibilidade e capacidade de escalabilidade.
- Ao contrário dos bancos de dados relacionais tradicionais, que armazenam dados em tabelas, o MongoDB utiliza **coleções de documentos no formato JSON**. Isso torna a modelagem de dados mais ágil e facilita a integração com outras tecnologias.
- Além disso, o MongoDB oferece **recursos avançados de indexação e busca**, permitindo trabalhar com consultas rápidas e eficientes, mesmo em grandes volumes de dados. O banco de dados também suporta **sharding** e **replicação**, o que ajuda a garantir **alta disponibilidade** e **tolerância a falhas**, mantendo o sistema operacional mesmo em caso de problemas.

Por que escolher o MongoDB?

- Uma das principais vantagens do **MongoDB** é sua capacidade de escalar horizontalmente. Isso significa que, à medida que a demanda aumenta, os usuários podem simplesmente adicionar mais servidores para lidar com o crescimento. Além disso, o MongoDB oferece uma grande flexibilidade, permitindo que os desenvolvedores trabalhem com diferentes tipos de dados e usem esquemas dinâmicos. Isso facilita a adição de novos campos ou a modificação da estrutura dos dados, sem a necessidade de alterar o esquema existente.
- Outra vantagem importante do **MongoDB** é seu excelente desempenho. Ele utiliza indexação automática para tornar as consultas mais rápidas e suporta operações de leitura e gravação em paralelo, o que aumenta ainda mais a velocidade.
- Além disso, o MongoDB tem recursos avançados de cache, que armazenam em memória as consultas mais acessadas, permitindo um acesso rápido e eficiente aos dados.

Modelagem de dados no MongoDB

O **MongoDB** é um banco de dados NoSQL que armazena informações de forma diferente dos bancos de dados relacionais tradicionais.

Em vez de usar tabelas e linhas, o MongoDB usa coleções e documentos:

- Coleção: Similar a uma tabela em um banco de dados relacional.
- Documento: Similar a uma linha, mas com muito mais flexibilidade. Cada documento é uma unidade independente que pode conter diversos campos e até subdocumentos (documentos dentro de documentos).

Introdução: O que é o MongoDB?

O **MongoDB** é um **Sistema de Gerenciamento de Banco de Dados (SGBD) NoSQL**, que é **open-source** e **orientado a documentos**. Aqui estão alguns dos seus principais diferenciais:

- **Alto Desempenho**: O MongoDB utiliza **documentos embutidos** e **índices** para otimizar as operações, o que resulta em um desempenho rápido e eficiente.
- **Linguagem de Consulta Rica**: Ele oferece uma **linguagem de consulta poderosa** que suporta:
  - **Operações CRUD**: Criar, Ler, Atualizar e Deletar dados.
  - **Agregações de Dados**: Permite transformar e combinar dados de diferentes formas.
  - **Busca por Texto**: Facilita a pesquisa por palavras-chave dentro dos documentos.
  - **Consultas Geoespaciais**: Suporta consultas relacionadas à localização e espaço geográfico.

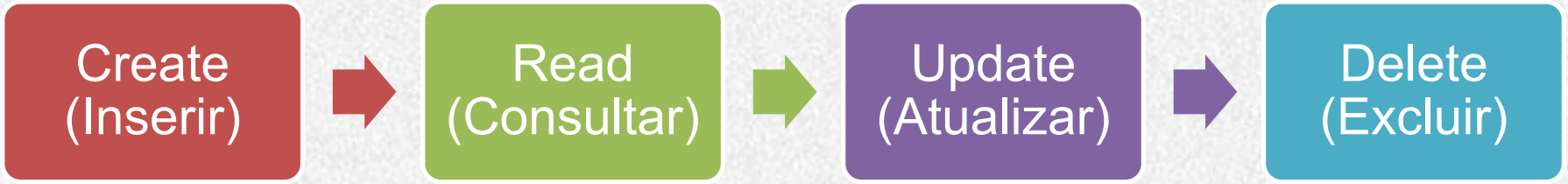
Introdução: O que é o MongoDB?

- **Alta Disponibilidade:** O MongoDB utiliza **Replica Sets**, que garantem que os dados estejam sempre disponíveis, mesmo se um dos servidores falhar.
- **Escalabilidade Horizontal:** Com **Sharding**, o MongoDB pode dividir dados entre vários servidores, permitindo que o sistema se expanda facilmente para lidar com grandes volumes de dados e tráfego.

Vantagens da modelagem de dados no MongoDB:

- **Flexibilidade:** Você pode adicionar novos campos aos documentos sem precisar modificar a estrutura da coleção inteira. Isso significa que o sistema pode evoluir e se adaptar com o tempo sem a necessidade de grandes alterações no banco de dados.
- **Esquemas dinâmicos:** O MongoDB permite que os desenvolvedores criem estruturas de dados que podem mudar e crescer conforme as necessidades do sistema, sem rigidez.
- **Índices eficientes:** O MongoDB suporta vários tipos de índices, que ajudam a acelerar as consultas e melhorar o desempenho geral do sistema.

CRUD no MongoDB



1. Create (Inserir)

- **Descrição**: Adiciona novos documentos a uma coleção.

2. Read (Consultar)

- **Descrição**: Recupera documentos de uma coleção com base em critérios de busca.

3. Update (Atualizar)

- **Descrição**: Modifica documentos existentes em uma coleção.

4. Delete (Excluir)

- **Descrição**: Remove documentos de uma coleção.

Documentos e Coleções no MongoDB

Estrutura de Dados do MongoDB:

- No MongoDB, os dados são armazenados em uma estrutura de **documentos**. Esses documentos são semelhantes a registros em um banco de dados tradicional, mas com mais flexibilidade.

Documentos BSON:

- Os documentos no MongoDB são armazenados no formato **BSON** (Binary JSON). BSON é uma forma binária do JSON (JavaScript Object Notation), que permite uma representação compacta e eficiente dos dados. Isso facilita a manipulação e a consulta dos dados dentro do banco.

Documentos e Coleções no MongoDB

Estrutura de Dados do MongoDB:

Coleções:

- **Coleções** no MongoDB são equivalentes às **tabelas** em bancos de dados relacionais (SQL).
- Uma coleção é um agrupamento de documentos que compartilham características semelhantes.
- Assim como uma tabela contém várias linhas em um banco de dados SQL, uma coleção contém vários documentos no MongoDB.

EDUCAÇÃO METODISTA

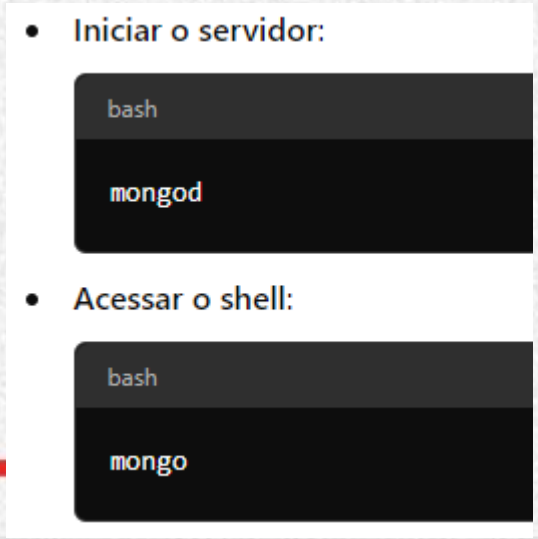
Executando o MongoDB

O **MongoDB** está disponível para as plataformas MacOS, Linux e Windows. Para instalar o MongoDB em seu sistema, siga as instruções detalhadas disponíveis na documentação oficial.

Após a instalação, você precisará iniciar o servidor do MongoDB. Para isso, execute o arquivo **mongod**. Este arquivo é o responsável por rodar o servidor de banco de dados.

Depois que o servidor estiver em execução, você pode acessar o **shell do MongoDB** usando o arquivo **mongo**.

No MacOS e no Linux, basta abrir o terminal e digitar o nome dos executáveis para iniciar o servidor e o shell.

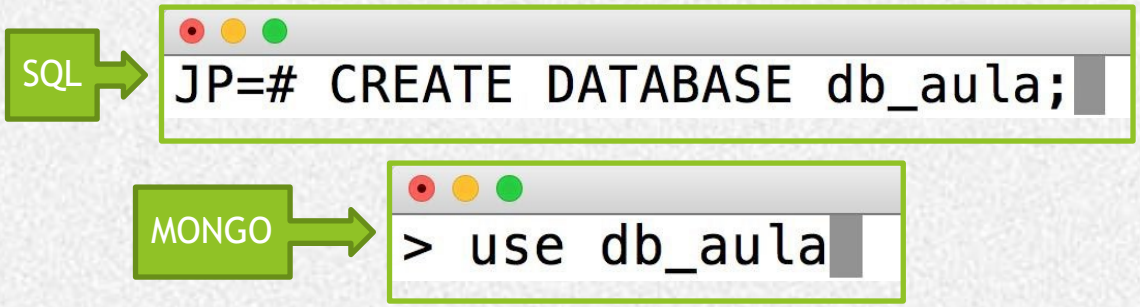


Criando um Banco de Dados

No **MongoDB**, muitos dos comandos de definição de dados (DDL) são simplificados e abstraídos. Em vez de definir a estrutura do banco de dados e das coleções antecipadamente, o MongoDB cria essas estruturas conforme elas são necessárias.

Para criar um banco de dados no MongoDB, você simplesmente usa o comando para acessar um banco de dados que ainda não foi criado. Assim que você insere o primeiro registro nesse banco de dados, ele é automaticamente criado e salvo.

Portanto, no MongoDB: **Não é necessário criar o banco de dados ou coleções antes de usá-las. As estruturas são criadas dinamicamente quando você adiciona dados.**



EDUCAÇÃO METODISTA

Coleções e Documentos no MongoDB

O **MongoDB** é um banco de dados orientado a documentos que organiza os dados de maneira flexível e intuitiva. Aqui está como isso funciona:

- **Coleções:** Os dados no MongoDB são agrupados em coleções.
  - Pense em uma coleção como uma pasta que contém documentos relacionados, assim como uma pasta no seu computador pode conter vários arquivos.

EDUCAÇÃO METODISTA

Coleções e Documentos no MongoDB

Documentos: Dentro das coleções, os dados são armazenados em **documentos**. Cada documento é uma unidade individual de dados, semelhante a um registro ou um objeto.

- **Identificador Único:** Cada documento possui um atributo especial chamado _id, que serve como um identificador único. Isso garante que cada documento possa ser facilmente localizado e diferenciado dos outros.
- **Atributos:** Além do _id, um documento pode ter uma variedade de outros atributos.

EDUCAÇÃO METODISTA

Coleções e Documentos no MongoDB

Flexibilidade:

- **ID Opcional:** Embora cada documento tenha um **_id**, você não precisa definir manualmente esse identificador. Se não o fizer, o MongoDB gera um automaticamente.
- **Tipos de Atributos:** Não é necessário especificar o tipo de dados dos atributos ao criar um documento. Você pode adicionar atributos conforme necessário, sem precisar definir antecipadamente seu tipo.
- **Estrutura Variável:** Documentos dentro da mesma coleção podem ter diferentes atributos. Por exemplo, um documento pode ter um atributo "endereço" enquanto outro pode não ter. Isso permite uma grande flexibilidade na organização dos dados.

EDUCAÇÃO METODISTA

Coleções e Documentos (Insert)

Criando uma Coleção: No MongoDB, uma coleção é criada automaticamente ao inserir o primeiro documento nela.

Operações de Inserção:

- Inserção de um único documento: Utilize o método **insertOne**. Este método recebe um único documento como parâmetro.
- Inserção de múltiplos documentos: Utilize o método **insertMany**. Este método recebe um vetor (array) de documentos como parâmetro, permitindo a inserção de vários documentos de uma vez.

EDUCAÇÃO METODISTA

Coleções e Documentos (Insert)

```javascript
// Inserção de um único documento
db.timesfutebol.insertOne({
  "_id": 1,
  "nome": "Cruzeiro",
  "pais": "Brasil"
})
// Resultado esperado
{
  "acknowledged": true,
  "insertedId": 1
}

// Inserção de múltiplos documentos
db.timesfutebol.insertMany([
  { "nome": "Barcelona", "pais": "Espanha" },
  { "nome": "Palmeiras", "pais": "Brasil", "mundial": 0 }
])
// Resultado esperado
{
  "acknowledged": true,
  "insertedIds": [
    ObjectId("5911b28009702042baa255e2"),
    ObjectId("5911b28009702042baa255e3")
  ]
}
```

db.timesfutebol.**insertOne()**

Função: Insere um único documento em uma coleção.

db.timesfutebol. **insertMany()**

Função: Insere múltiplos documentos de uma vez em uma coleção.

Resultado: Retorna um objeto com "acknowledged": true, indicando que a inserção foi bem-sucedida, e "insertedIds", que é um array contendo os IDs dos documentos inseridos.

EDUCAÇÃO METODISTA

Coleções e Documentos (FIND)

```javascript
db.clientes.find({ cidade: "São Paulo" })
```

db = nome do banco de dados.

clientes = nome da coleção dentro desse banco de dados.

```
db.clientes.findOne({ cidade: "São Paulo" })
```

O MongoDB oferece dois métodos principais para buscar informações em documentos:

find(): Este método retorna todos os documentos que correspondem aos critérios de busca que você especificar. Pense nele como uma lista de resultados que atende à sua pesquisa.

findOne(): Este método retorna apenas um único documento que atende aos critérios de busca. Se houver mais de um documento que corresponda, ele trará apenas o primeiro que encontrar.

Coleções e Documentos (Select)

Tanto o método find() quanto o findOne() permitem que você defina critérios de seleção e projeção para os resultados.

Isso significa que você pode escolher quais documentos deseja buscar e quais campos desses documentos serão retornados na consulta.

```javascript
db.usuarios.findOne(
  { "idade": 35 },  // Critério de seleção
  { "nome": 1, "idade": 1, "_id": 0 }  // Projeção: exibe apenas nome e idade, oculta _id
)
```

EDUCAÇÃO METODISTA

Coleções e Documentos (Update)

O MongoDB oferece três métodos principais para atualizar dados em documentos. Veja como cada um deles funciona:

updateOne() - atualiza um único documento que corresponde aos critérios especificados. **Uso:** Ideal quando você deseja alterar apenas um documento que atenda às condições.

updateMany() - atualiza todos os documentos que correspondem aos critérios especificados. **Uso:** Utilize este método quando você precisa modificar vários documentos ao mesmo tempo.

replaceOne() - substitui um único documento que corresponde aos critérios especificados por um novo documento completo. **Nota:** O atributo _id do documento original permanece o mesmo após a substituição.

EDUCAÇÃO METODISTA

Coleções e Documentos (Update)

updateOne()

Objetivo: Atualizar um único documento que corresponde aos critérios especificados.

Exemplo: Suponha que você tenha uma coleção clientes e deseja atualizar o endereço do cliente com o nome "João" para "Rua Nova, 123".

```javascript
db.clientes.updateOne(
  { nome: "João" },                    // Critério de busca
  { $set: { endereco: "Rua Nova, 123" } }  // Atualização
);
```

Explicação: O método updateOne() encontra o primeiro documento onde o campo nome é "João" e atualiza seu campo endereco para "Rua Nova, 123".

EDUCAÇÃO METODISTA

Coleções e Documentos (Update)

updateMany()

Objetivo: Atualizar todos os documentos que correspondem aos critérios especificados.

Exemplo: Agora, imagine que você deseja atualizar o status de todos os clientes cujo status atual é "inativo" para "ativo".

```javascript
db.clientes.updateMany(
  { status: "inativo" },           // Critério de busca
  { $set: { status: "ativo" } }    // Atualização
);
```

Explicação: O método updateMany() encontra todos os documentos onde o campo status é "inativo" e altera o campo status para "ativo".

EDUCAÇÃO METODISTA

Coleções e Documentos (Update)

replaceOne()

Objetivo: Substituir um único documento que corresponde aos critérios especificados por um novo documento completo.

Exemplo: Suponha que você deseja substituir o documento do cliente com o nome "Maria" por um novo documento com todas as informações atualizadas.

```javascript
db.clientes.replaceOne(
  { nome: "Maria" },                // Critério de busca
  { nome: "Maria", endereco: "Avenida Central, 456", idade: 30 } // Novo documento
);
```

Explicação: O método replaceOne() encontra o documento onde o campo nome é "Maria" e substitui todo o documento existente por um novo documento com os campos nome, endereco, e idade.

Coleções e Documentos (DELETE)

O MongoDB oferece dois métodos principais para remover documentos de uma coleção:

deleteOne() - remove um único documento que corresponde aos critérios especificados. **Uso:** Utilize este método quando você deseja excluir apenas um documento que atende às condições fornecidas.

Exemplo: Para remover o documento do cliente com o nome "João".

```javascript
db.clientes.deleteOne(
  { nome: "João" }  // Critério de busca
);
```

Explicação: O método deleteOne() encontra o primeiro documento onde o campo nome é "João" e o remove da coleção.

Coleções e Documentos (DELETE)

O MongoDB oferece dois métodos principais para remover documentos de uma coleção:

deleteMany() - Remove todos os documentos que correspondem aos critérios especificados. **Uso:** Use este método quando você precisa excluir vários documentos ao mesmo tempo que atendem às condições fornecidas.

Exemplo: Para remover todos os documentos de clientes cujo status é "inativo".

```javascript
db.clientes.deleteMany(
  { status: "inativo" }  // Critério de busca
);
```

Explicação: O método deleteMany() encontra todos os documentos onde o campo status é "inativo" e os remove da coleção.

Referências

https://www.mongodb.com/try/download/community
https://www.mongodb.com/try/download/compass

https://www.youtube.com/watch?v=Rb25Ka905Bs
https://www.mongodb.com/docs/manual/reference/method/db.collection.find/
https://paca.ime.usp.br/pluginfile.php/76002/mod_resource/content/4/mac439_aula8.pdf
http://www.w3big.com/pt/mongodb/default.html

EDUCAÇÃO METODISTA