
alpha
<ed/tech>
BANCOS DE DADOS
</>

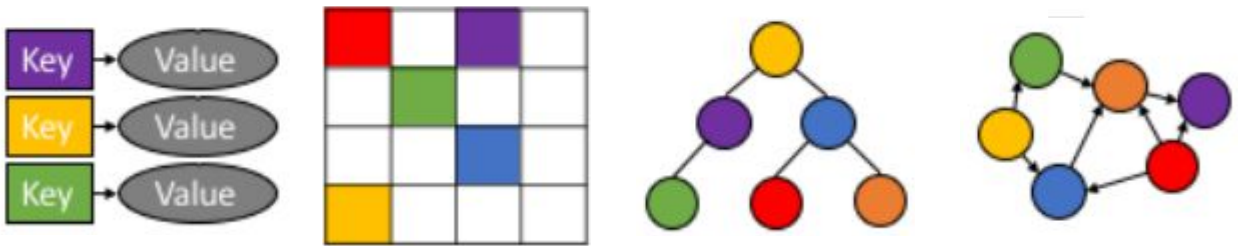
Banco de Dados Não-Relacional

NO-SQL

Olá pessoal! Agora que já compreendemos os banco de dados relacionais falta conhecermos o não-relacionais!

Recapitulando...

Como já vimos anteriormente, na nossa primeira aula, um banco de dados não-relacional, também conhecido como NoSQL (Not Only SQL), é uma categoria de sistemas de gerenciamento de banco de dados que não segue o modelo tradicional de banco de dados relacionais. Isso significa que eles não usam a estrutura tabular de linhas e colunas para armazenar dados, como fazem os bancos de dados relacionais. Em vez disso, eles são projetados para lidar com uma variedade de tipos de dados e estruturas, incluindo documentos, grafos, pares de chave-valor e colunas.



Exemplos tipos de bancos de dados não-relacionais.

Por exemplo, imagine que você está organizando sua coleção de livros. Em um banco de dados relacional, você poderia criar uma tabela com as colunas "Título", "Autor", "Gênero" e assim por diante. No entanto, em um banco de dados não-relacional, você poderia optar por armazenar cada livro como um documento, onde todas as informações sobre o livro, como título, autor e gênero, são agrupadas em um único documento.

Os bancos de dados não-relacionais surgiram como uma resposta às limitações dos sistemas de banco de dados relacionais tradicionais em lidar com a escala e a variedade de dados gerados por aplicativos modernos. Com o advento da internet e das redes sociais, a quantidade de dados não estruturados, como textos, imagens e vídeos, aumentou significativamente. Os bancos de dados relacionais, que foram projetados para lidar principalmente com dados estruturados em tabelas, começaram a mostrar sinais de sobrecarga quando confrontados com esse tipo de cenário.

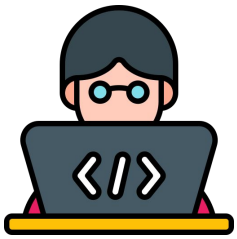
Por exemplo, se você estiver construindo um aplicativo de rede social, onde cada usuário pode ter uma variedade de informações associadas a ele, como postagens, fotos, amigos e comentários, um banco de dados relacional tradicional pode se tornar complicado de gerenciar. Os bancos de dados não-relacionais oferecem uma solução mais flexível para lidar com esses tipos de dados variados.

Eles também oferecem um série de vantagens em relação aos sistemas relacionais, especialmente em cenários onde a escalabilidade, a flexibilidade e o desempenho são prioridades.

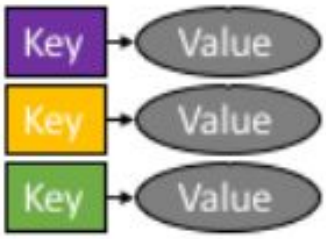
Sabemos que eles suportam uma variedade de modelos de dados, cada um projetado para atender a diferentes necessidades e requisitos de aplicação. Então vamos rever cada um deles:

Banco de Dados Não-Relacional

Modelos



Bancos de Dados de Chave-Valor

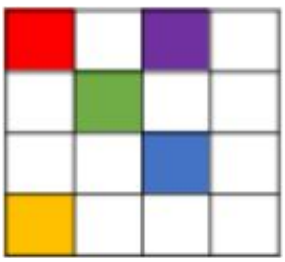


Características Principais: Bancos de dados colunares armazenam dados em colunas em vez de linhas, o que os torna eficientes para consultas analíticas que envolvem a análise de um subconjunto de colunas. São ideais para cenários em que é necessário analisar grandes volumes de dados.

Exemplo de Uso:

- Armazenamento de Sessões de Usuários: Usado em sistemas web para armazenar sessões de usuário. Exemplo de software: Redis.
- Cache de Dados: Usado para armazenar em cache dados frequentemente acessados. Exemplo de software: Memcached.
- Gerenciamento de Configurações: Usado para armazenar configurações de aplicativos. Exemplo de software: etcd.

Bancos de Dados Colunar



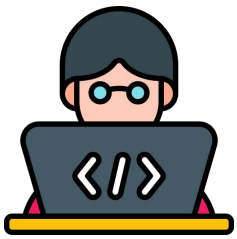
Características Principais: Nesse modelo, os dados são armazenados em colunas em vez de linhas. Isso permite uma compressão eficiente e a recuperação rápida de conjuntos de dados específicos, tornando-os ideais para análises e consultas complexas.

Exemplo de Uso:

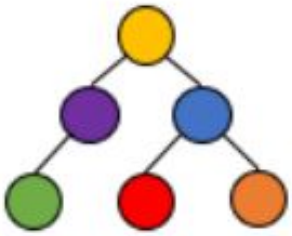
- Análise de Dados: Usado para análise de dados em cenários de business intelligence e data warehousing. Exemplo de software: Amazon Redshift.
- Armazenamento de Dados de Log: Usado para armazenar grandes volumes de dados de log para análise. Exemplo de software: Apache Kudu.
- Processamento Analítico Online (OLAP): Usado para consultas analíticas interativas em tempo real. Exemplo de software: ClickHouse.

Banco de Dados Não-Relacional

Modelos



Bancos de Dados de Documentos

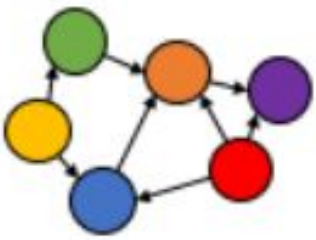


Características Principais: Bancos de dados de documentos armazenam dados em formato de documento, geralmente em JSON ou BSON. Cada documento é autocontido e pode conter campos de diferentes tipos. São flexíveis e escaláveis, adequados para uma ampla variedade de casos de uso.

Exemplo de Uso:

- Aplicações Web: Usado como banco de dados principal em muitas aplicações web modernas. Exemplo de software: MongoDB.
- Gestão de Conteúdo: Usado para gerenciar conteúdo de sites e blogs. Exemplo de software: Couchbase.
- Armazenamento de Perfil de Usuário: Usado para armazenar informações de perfil de usuário em sistemas web e aplicativos móveis. Exemplo de software: Firebase Firestore.

Bancos de Dados de Grafos



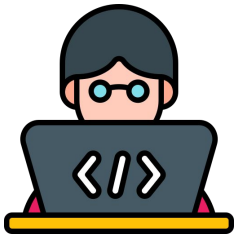
Características Principais: Bancos de dados de grafos são projetados para armazenar e consultar dados que estão estruturados como grafos, compostos por nós, arestas e propriedades. Eles são eficazes para modelar relacionamentos complexos e realizar consultas de grafos, como busca de caminhos e análises de redes.

Exemplo de Uso:

- Redes Sociais: Usado para modelar conexões entre usuários em redes sociais. Exemplo de software: Neo4j.
- Análise de Redes: Usado para analisar relacionamentos em redes complexas, como redes de transporte ou redes de infraestrutura. Exemplo de software: Amazon Neptune.
- Recomendações Personalizadas: Usado para sistemas de recomendação que dependem da análise de relações entre diferentes itens. Exemplo de software: ArangoDB.

Banco de Dados Não-Relacional

Comparativo

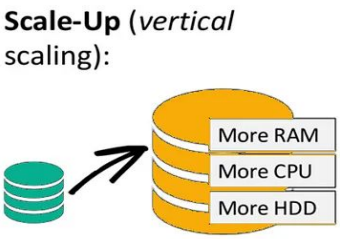


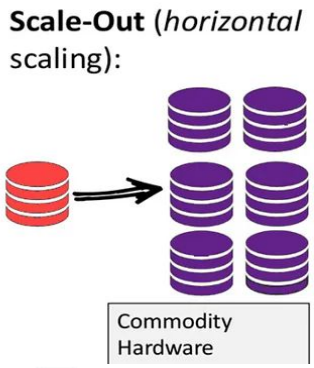
Vantagens

Flexibilidade de Esquema
Os bancos de dados não-relacionais permitem que você armazene dados com estruturas flexíveis, sem a necessidade de definir um esquema rígido de antemão. Isso significa que você pode adicionar novos tipos de dados sem ter que modificar todo o esquema do banco de dados.

Escalabilidade Horizontal
Eles são projetados para escalonamento horizontal, o que significa que podem lidar com grandes volumes de dados distribuídos em vários servidores. Isso os torna ideais para aplicativos que precisam escalar rapidamente para atender a demandas crescentes.

Desempenho
Para determinados casos de uso, como consultas em dados altamente conectados (grafos) ou análises em grande escala, os bancos de dados não-relacionais podem oferecer melhor desempenho do que os sistemas relacionais tradicionais.



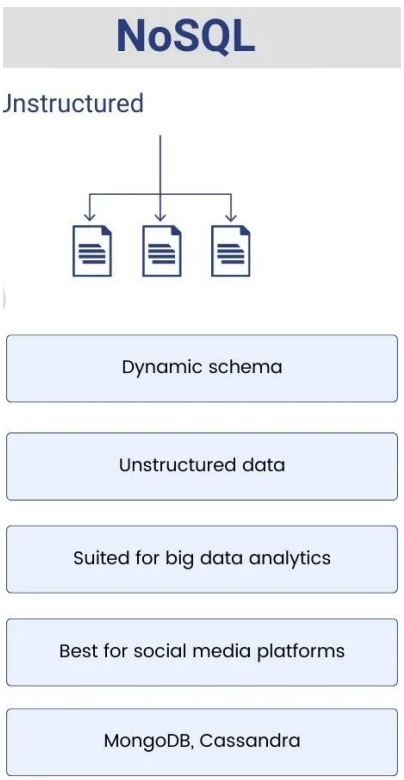


Desvantagens

Consistência Eventual
Alguns sistemas de banco de dados não-relacionais priorizam a disponibilidade e a particionabilidade em detrimento da consistência imediata dos dados, o que pode levar a uma eventual consistência nos dados. Isso significa que pode haver um atraso entre a atualização dos dados em diferentes partes do sistema.

Menor Maturidade
Comparado aos sistemas de banco de dados relacionais, o ecossistema de bancos de dados não-relacionais pode ser menos maduro, com menos ferramentas e recursos disponíveis. Isso pode tornar mais difícil encontrar soluções para problemas específicos.

Menor Suporte a Transações
Alguns bancos de dados não-relacionais podem oferecer suporte limitado a transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade), o que pode ser um problema para aplicativos que exigem garantias de integridade dos dados.



Banco de Dados Não-Relacional

MongoDB

O MongoDB é um sistema de gerenciamento de banco de dados NoSQL (Not Only SQL) de código aberto e orientado a documentos. Isso significa que ele armazena dados em documentos JSON-like com esquemas dinâmicos, em vez de usar tabelas e linhas como em um banco de dados relacional tradicional. O MongoDB foi projetado para oferecer escalabilidade, desempenho e flexibilidade para lidar com uma variedade de casos de uso, desde aplicativos da web até análises em tempo real.

Por exemplo, imagine que você está construindo um aplicativo de comércio eletrônico. Com o MongoDB, você poderia armazenar informações sobre produtos, clientes e pedidos em documentos flexíveis, onde cada documento pode ter uma estrutura diferente. Isso oferece mais flexibilidade do que um banco de dados relacional tradicional, onde você precisaria definir um esquema rígido antes de armazenar os dados.

História

O MongoDB foi desenvolvido pela empresa MongoDB Inc. e foi lançado pela primeira vez em 2009. Foi inspirado pelo modelo de banco de dados orientado a documentos usado internamente pela empresa de jogos online 10gen (agora MongoDB Inc.). Desde então, o MongoDB cresceu rapidamente em popularidade devido à sua facilidade de uso, escalabilidade e flexibilidade.

O MongoDB é escrito em C++ e possui suporte para várias plataformas, incluindo Linux, Windows e macOS. Ele é distribuído sob a licença GNU Affero General Public License, que permite seu uso gratuito e redistribuição, mesmo em aplicações comerciais. Atualmente, é uma das principais escolhas para bancos de dados NoSQL em uma variedade de aplicações, desde pequenos aplicativos de inicialização até grandes empresas de tecnologia.

Características

Modelo de Dados Orientado a Documentos: Ele armazena dados em documentos JSON-like, o que permite uma representação flexível e hierárquica dos dados.

Esquema Dinâmico: Os documentos no MongoDB não precisam ter uma estrutura fixa, permitindo a adição de novos campos ou a modificação da estrutura dos documentos conforme necessário.

Escalabilidade Horizontal: Ele suporta escalabilidade horizontal, o que significa que pode distribuir dados em vários servidores para lidar com grandes volumes de dados e alta carga de trabalho.

Alta Disponibilidade: O MongoDB oferece recursos de replicação e failover automático para garantir alta disponibilidade e resiliência contra falhas.
Índices: Ele suporta índices secundários e indexação geoespacial para melhorar o desempenho de consultas.

Consulta Poderosa: Ele fornece uma linguagem de consulta flexível e poderosa que permite realizar consultas complexas e agregações de dados.
Suporte a Transações: A partir da versão 4.0, o MongoDB oferece suporte a transações ACID em um único documento ou em vários documentos em uma coleção.

Banco de Dados Não-Relacional

MongoDB: Instalação

Instalação no Linux

1. **Abra o terminal.**

2. **Adicione a chave GPG para o repositório oficial do MongoDB**

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc
| sudo apt-key add -
```

3. **Adicione o repositório do MongoDB**

```
echo "deb [ arch=amd64,arm64 ]
https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4
multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.4.list
```

4. **Atualize os pacotes do sistema**

```
sudo apt-get update
```

5. **Instale o MongoDB**

```
sudo apt-get install -y mongodb-org
```

6. **Inicie o serviço do MongoDB**

```
sudo systemctl start mongod
```

7. **Verifique o status do MongoDB**

```
sudo systemctl status mongod
```

Banco de Dados Não-Relacional

MongoDB: Instalação

Instalação no Mac

1. **Você pode instalar o MongoDB usando o Homebrew. Se você não tiver o Homebrew instalado, instale-o primeiro**

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. **Instale o MongoDB usando o Homebrew**

```
brew tap mongodb/brew
brew install mongodb-community
```

3. **Inicie o MongoDB**

```
brew services start mongodb/brew/mongodb-community
```

4. **Verifique se o MongoDB está em execução**

```
brew services list
```

Banco de Dados Não-Relacional

MongoDB: Instalação

Instalação no Windows

1. **Baixe o instalador do MongoDB para Windows no site oficial (https://www.mongodb.com/try/download/community).**

2. **Execute o instalador e siga as instruções na tela.**

3. **Durante a instalação, selecione a opção "Complete" para instalar todos os componentes do MongoDB.**

4. **Após a instalação, abra o prompt de comando como administrador.**

5. **Navegue até o diretório onde o MongoDB foi instalado (por padrão, é C:\Program Files\MongoDB\Server{versão}) e então navegue para a pasta "bin".**

6. **Inicie o MongoDB executando o seguinte comando**

```
mongod
```

7. **O MongoDB deve estar agora em execução. Para interagir com ele, abra outra janela do prompt de comando e navegue até a pasta "bin" novamente. Execute o comando abaixo para iniciar o shell do MongoDB:**

```
mongo
```

Banco de Dados Não-Relacional

MongoDB: Arquitetura

Arquitetura do MongoDB

O MongoDB é um banco de dados NoSQL que segue uma arquitetura distribuída e orientada a documentos. Sua arquitetura é composta por três componentes principais:

Servidor MongoDB (mongod)
Este é o processo responsável por armazenar e manipular os dados. Ele gerencia a leitura e gravação dos dados em disco, bem como a indexação e a execução de consultas.

Cliente MongoDB (mongo)
Este é o cliente de linha de comando usado para interagir com o servidor MongoDB. Ele permitirá aos desenvolvedores executarem comandos de banco de dados, consultas e operações administrativas.

Conjunto de Réplicas (Replica Set)
O conjunto de réplicas é uma configuração onde múltiplos servidores MongoDB são configurados para mantermos cópias sincronizadas dos mesmos dados. Isso oferece alta disponibilidade e tolerância a falhas, garantindo que os dados permaneçam acessíveis mesmo em caso de falha de um servidor.

Por exemplo, imagine que você está construindo um aplicativo de comércio eletrônico. O servidor MongoDB (mongod) armazenaria todos os dados relacionados aos produtos, pedidos e clientes, enquanto o cliente MongoDB (mongo) permitiria que você executasse consultas e operações administrativas no banco de dados. E se você estiver usando um conjunto de réplicas, várias cópias dos dados serão mantidas em servidores diferentes para garantir alta disponibilidade e resiliência contra falhas.

Para começar a trabalhar com o MongoDB em Python, primeiro você precisa instalar o pacote `*pymongo*`, que é o driver oficial do MongoDB para Python. Você pode instalar o `*pymongo*` usando o pip:

```
pip install pymongo
```

Depois de instalar o pymongo, você pode se conectar ao servidor MongoDB usando a classe *MongoClient*. Aqui está um exemplo simples de como fazer isso:

```
from pymongo import MongoClient

# Conectando ao servidor MongoDB
client = MongoClient('mongodb://localhost:27017/')
```

Banco de Dados Não-Relacional

MongoDB: Arquitetura

Coleções e Documentos

No MongoDB, os dados são organizados em bancos de dados, coleções e documentos:

Banco de Dados (Database)
Um banco de dados no MongoDB é uma coleção de coleções. Cada banco de dados é representado por um nome único e pode conter várias coleções.

Coleção (Collection)
Uma coleção no MongoDB é semelhante a uma tabela em um banco de dados relacional. Ela é um grupo de documentos relacionados, onde cada documento pode ter uma estrutura diferente. As coleções são criadas dinamicamente quando você insere dados pela primeira vez.

Documento (Document)
Um documento no MongoDB é uma unidade básica de dados e é representado em formato JSON-like (BSON). Ele pode conter pares de chave-valor e aninhamento de documentos, permitindo uma representação flexível dos dados.

Por exemplo, imagine que você está criando um banco de dados para um blog. Você pode ter um banco de dados chamado "blog" que contém várias coleções, como "posts" e "comentários". Cada documento na coleção "posts" pode conter informações sobre um post específico, como título, autor e conteúdo.

Para criar um banco de dados, coleção e inserir documentos usando o pymongo, você pode fazer o seguinte:

```
# Acessando um banco de dados
db = client['mydatabase']

# Acessando uma coleção dentro do banco de dados
collection = db['mycollection']

# Inserindo um documento na coleção
document = {'name': 'John', 'age': 30, 'city': 'New York'}
collection.insert_one(document)
```

Banco de Dados Não-Relacional

MongoDB: Operações

CRUD (Criar, Ler, Atualizar, Deletar)

No MongoDB, as operações CRUD são fundamentais para interagir com os dados:

Criar (Create)
Para inserir dados no MongoDB, você pode usar o método `insertOne()` para inserir um único documento ou `insertMany()` para inserir vários documentos de uma vez.

Ler (Read)
Para ler dados do MongoDB, você pode usar o método `find()` para recuperar documentos que correspondam a um critério de consulta específico. Você também pode usar métodos como `findOne()` para recuperar um único documento ou `count()` para contar o número de documentos em uma coleção.

Atualizar (Update)
Para atualizar dados no MongoDB, você pode usar o método `updateOne()` para atualizar um único documento ou `updateMany()` para atualizar vários documentos de uma vez. Você pode especificar um critério de consulta para selecionar os documentos a serem atualizados e uma operação de modificação para atualizar os campos desejados.

Deletar (Delete)
Para excluir dados do MongoDB, você pode usar o método `deleteOne()` para excluir um único documento ou `deleteMany()` para excluir vários documentos de uma vez. Você pode especificar um critério de consulta para selecionar os documentos a serem excluídos.

Aqui está como você pode executar operações CRUD usando o pymongo:

```
# Criar: Inserir um documento
collection.insert_one({'name': 'Alice', 'age': 25, 'city':
'London'})

# Ler: Encontrar documentos
result = collection.find_one({'name': 'Alice'})
print(result)

# Atualizar: Atualizar um documento
collection.update_one({'name': 'Alice'}, {'$set': {'age':
26}})

# Deletar: Deletar um documento
collection.delete_one({'name': 'Alice'})
```

Este código cria um novo documento, encontra, atualiza e deleta um documento já existente na coleção.

Banco de Dados Não-Relacional

MongoDB: Operações

Indexação

A indexação é importante para melhorar o desempenho das consultas no MongoDB. Você pode criar índices em campos específicos em uma coleção para acelerar a recuperação de dados:

Índice Simples: Um índice simples é criado em um único campo em uma coleção. Ele acelera consultas que filtram ou ordenam os dados com base nesse campo.

Índice Composto: Um índice composto é criado em múltiplos campos em uma coleção. Ele acelera consultas que filtram ou ordenam os dados com base em uma combinação desses campos.

Índice Texto: Um índice de texto é criado em um campo de texto em uma coleção. Ele suporta consultas de texto completo, permitindo que você pesquise por palavras-chave em documentos de texto.

Por exemplo, se você estiver criando uma aplicação de blog e quiser acelerar consultas por título de post, você pode criar um índice simples no campo "title" da coleção "posts":

Você pode criar um índice em um campo específico da seguinte forma:

```
# Criar um índice em um campo
collection.create_index([('name', pymongo.ASCENDING)])
```

Isso criará um índice ascendente no campo 'name'.

Operações Agregadas em MongoDB

As operações agregadas no MongoDB permitem realizar operações de agregação, como agrupamento, contagem, média, entre outras, em documentos em uma coleção.

Você pode usar operações de agregação para encontrar a contagem total de documentos em uma coleção:

```
# Contagem total de documentos na coleção
total_count = collection.count_documents({})
print(total_count)
```

Isso imprimirá o número total de documentos na coleção.

Trabalhar com o MongoDB em Python é bastante simples usando o pacote `*pymongo*`. Com os conceitos de arquitetura do MongoDB, bancos de dados, coleções, documentos, operações CRUD, indexação e operações agregadas, você pode começar a criar aplicativos poderosos que utilizam o MongoDB como seu banco de dados.

Banco de Dados Não-Relacional

MongoDB: Modelagem de Dados

Modelagem de dados é o processo de projetar a estrutura de um banco de dados para atender aos requisitos de um aplicativo específico. Envolve identificar as entidades relevantes, seus atributos e os relacionamentos entre elas. Os conceitos básicos de modelagem de dados incluem:

Entidades
São objetos ou conceitos sobre os quais você deseja armazenar informações. Por exemplo, em um sistema de gerenciamento escolar, as entidades podem incluir Aluno, Professor e Disciplina.

Atributos
São características ou propriedades das entidades. Por exemplo, um aluno pode ter atributos como nome, idade e número de identificação.

Relacionamentos
São associações entre diferentes entidades. Por exemplo, um aluno está matriculado em várias disciplinas, o que representa um relacionamento entre as entidades Aluno e Disciplina.

Modelos Relacionais vs Modelos Não-Relacionais

Nos modelos de dados relacionais, os dados são organizados em tabelas, onde cada linha representa uma instância de uma entidade e cada coluna representa um atributo. Os relacionamentos são estabelecidos por meio de chaves estrangeiras que conectam as tabelas. Por exemplo, em um banco de dados relacional, você pode ter uma tabela de Alunos e uma tabela de Disciplinas, conectadas por uma chave estrangeira que representa a relação de matrícula.

Nos modelos de dados não-relacionais, como o MongoDB, os dados são armazenados em documentos flexíveis no formato JSON (JavaScript Object Notation). Isso permite que você modele os dados de forma mais natural, sem a necessidade de seguir um esquema rígido. Por exemplo, em um banco de dados MongoDB, você pode ter um documento representando um aluno com todos os seus atributos, incluindo as disciplinas em que está matriculado, em um único documento.

Modelagem de Dados

Na modelagem de dados no MongoDB, você pode usar os seguintes conceitos:

Coleções
São equivalentes às tabelas em bancos de dados relacionais. Cada coleção contém um conjunto de documentos relacionados.

Documentos
São os objetos de dados individuais armazenados no MongoDB. Cada documento é representado em formato JSON e pode ter uma estrutura diferente, permitindo uma modelagem flexível de dados.

Chaves Primárias
No MongoDB, cada documento em uma coleção tem um campo de identificação exclusivo chamado de "_id". Esse campo funciona como a chave primária e é automaticamente atribuído pelo MongoDB, a menos que especificado de outra forma.

Banco de Dados Não-Relacional

MongoDB: Modelagem de Dados

Exemplos Práticos de Modelagem de Dados no MongoDB

Vamos considerar um exemplo de modelagem de dados para um aplicativo de e-commerce usando o MongoDB:

Coleção "Produtos"
Nesta coleção, podemos armazenar informações sobre os produtos vendidos na loja. Cada documento na coleção representa um produto e pode conter atributos como nome, descrição, preço e categoria.

```
{
    "_id": ObjectId("617b1d4b68f57515e7c585b5"),
    "nome": "Camiseta",
    "descricao": "Camiseta de algodão",
    "preco": 29.99,
    "categoria": "Roupas"
}
```

Coleção "Clientes"
Aqui, podemos armazenar informações sobre os clientes que compram na loja. Cada documento representa um cliente e pode incluir atributos como nome, endereço e histórico de pedidos.

```
{
    "_id": ObjectId("617b1d4b68f57515e7c585b6"),
    "nome": "João",
    "endereco": "Rua A, 123",
    "historico_pedidos": [
        {
            "data": ISODate("2023-10-30T08:00:00Z"),
            "produtos": [
                ObjectId("617b1d4b68f57515e7c585b5")
            ]
        }
    ]
}
```

Banco de Dados Não-Relacional

MongoDB: Modelagem de Dados

Exemplos Práticos de Modelagem de Dados no MongoDB

Neste nosso aplicativo de e-commerce usando o MongoDB também teremos:

Coleção "Pedidos"
Aqui, podemos armazenar informações sobre os pedidos feitos pelos clientes. Cada documento representa um pedido e pode conter atributos como data, cliente e lista de produtos.

```
{
    " id": ObjectId("617b1d4b68f57515e7c585b7"),
    "data": ISODate("2023-10-30T08:00:00Z"),
    "cliente": ObjectId("617b1d4b68f57515e7c585b6"),
    "produtos": [
        {
            "produto id": ObjectId("617b1d4b68f57515e7c585b5"),
            "quantidade": 2
        }
    ]
}
```

Esses são exemplos simples de como você pode modelar dados em um banco de dados MongoDB. Você pode expandir esses modelos adicionando mais atributos e relacionamentos conforme necessário para atender aos requisitos do seu aplicativo.

A flexibilidade do MongoDB permite que você ajuste o esquema de dados conforme seu aplicativo evolui ao longo do tempo.

Ao longo desta aula, exploramos o fascinante mundo do MongoDB. Começamos entendendo os conceitos básicos de modelagem de dados. Em seguida, comparamos os modelos de dados relacionais e não-relacionais, destacando as diferenças fundamentais. Aprendemos sobre as coleções, documentos e chaves primárias que formam a base do armazenamento de dados neste poderoso banco de dados NoSQL.

Espero que esta aula tenha sido uma jornada informativa e inspiradora para todos vocês. Lembrem-se sempre de que a modelagem de dados é uma arte e uma ciência, e dominá-la é essencial para o sucesso de qualquer projeto de desenvolvimento de software. Continuem explorando, praticando e aprimorando suas habilidades, pois o conhecimento adquirido aqui será um alicerce sólido para seus futuros empreendimentos na área de tecnologia.

Com isso, encerramos esta aula de MongoDB. Desejo a todos um excelente caminho em sua jornada de aprendizado e desenvolvimento.

Obrigado e até a próxima aula!