



# Database Application

Manipulação de Dados

## Resumo

Um Relatório Detalhado de Extração, Transformação e Carregaregamento (ETL), processo que as organizações orientadas a dados usam para coletar dados de várias fontes e reuni-los para dar suporte à descoberta, à geração de relatórios, à análise e à tomada de decisões.

Integrantes:

Ângelo Santos (01589358)

José Miguel (01665230)

Livya Carvalho (01645272)

Thais Melo (01068175)

Túlio Mendes (01633581)

Recife/PE

14 de maio, 2024

*Documento de software*

Professor Responsável	Disciplina
Filipe Nascimento	Database Application

*Objetivo deste Documento*

Este documento tem como objetivo analisar e manipular o dataset NYC Airbnb Open Data, disponível na plataforma Kaggle. Fazendo gerenciamento em contêiner pelo Docker, utilizando bancos SQL e NoSQL com a tecnologia MongoDB orientado a documentos.

*Histórico de Revisão*

Data	Versão	ABR	MAI
23/04/2024	Início do Projeto	X	
30/04/2024	Primeiro Checkpoint	X	
30/04/2024	Elaboração do Documento	X	
07/05/2024	Segundo Checkpoint		X
11/05/2024	Revisão do Documento		X
13/05/2024	Ajustes Finais		X
14/05/2024	Entrega		X

## Sumário

1 Introdução.....	7
1.1 Finalidade.....	8
1.2 Escopo.....	8
1.3 Visão Geral.....	8
2 Tratamento e Normalização de Dados.....	10
2.1 Definições.....	10
2.1.1 Limpeza de Dados.....	10
2.1.2 Transformação de Dados.....	10
2.1.3 Análise de Dados.....	10
2.1.4 Visualização de Dados.....	10
2.1.5 Interpretação de Dados.....	10
2.2 Aplicações.....	11
2.2.1 Tomada de Decisões.....	11
2.2.2 Análise de Negócios.....	11
2.2.3 Pesquisa Científica.....	11
2.2.4 Personalização de Serviços.....	11
2.2.5 Detecção de Fraudes.....	11
2.2.6 Previsão e Planejamento.....	11
2.3 Métodos de Tratamento de Dados.....	11
2.3.1 Limpeza e Pré-processamento.....	11
2.3.2 Transformação e Enriquecimento.....	12
2.3.3 Análise Estatística.....	12
2.3.4 Mineração de Dados.....	12
2.3.5 Aprendizado de Máquina.....	12
2.4 Tipos de Tratamento de Dados.....	12
2.4.1 Tratamento de Dados Estruturados.....	12
2.4.2 Tratamento de Dados Não Estruturados.....	12
2.4.3 Tratamento de Dados Semi-Estruturados.....	12
2.4.4 Tratamento de Dados em Tempo Real.....	13
2.4.5 Tratamento de Big Data.....	13
3 Dicionário de Dados.....	14
3.1 Estrutura.....	14
3.2 Identificação da Entidade.....	14
3.3 Descrição dos Atributos e Tipos de Dados.....	14
3.4 Chaves Primárias e Estrangeiras.....	14
3.5 Restrições e Regras de Negócio.....	14
4 Modelagem de Dados.....	16
4.1 Modelo Conceitual.....	16
4.2 Modelo Lógico.....	17
4.3 Modelo Físico.....	18
5 DDLs.....	19
5.1 Principais exemplos de DDLs.....	19
5.1.1 CREATE TABLE.....	19
5.1.2 ALTER TABLE.....	19
5.1.3 DROP TABLE.....	19

---

6 DMLs.....	20
6.1 Principais exemplos de DMLs.....	20
6.1.1 INSERT INTO.....	20
6.1.2 UPDATE.....	20
6.1.3 DELETE.....	20
7 DQLs.....	21
7.1 Exemplo de DQL.....	21
7.1.1 SELECT.....	21
8 Docker.....	22
8.1 Definições.....	22
8.1.1 Contêineres.....	22
8.1.2 Imagens.....	22
8.1.3 Dockerfile.....	22
8.1.4 Docker Engine.....	22
8.1.5 Docker Compose.....	23
8.2 Usos.....	23
8.2.1 Padronização de Ambientes de Desenvolvimento.....	23
8.2.2 Implantação de Aplicativos em Micro Serviços.....	23
8.2.3 Implantação em Nuvem e Orquestração de Contêineres.....	23
8.2.4 Integração Contínua e Implantação Contínua (CI/CD).....	24
8.2.5 Desenvolvimento de Aplicativos Multiplataforma.....	24
8.3 Importância.....	24
8.3.1 Consistência e Reprodutibilidade.....	24
8.3.2 Eficiência e Economia de Recursos.....	24
8.3.3 Flexibilidade e Portabilidade.....	24
8.3.4 Escalabilidade e Resiliência.....	25
8.3.5 Facilidade de Gerenciamento e Manutenção.....	25
8.4 Como pode ser feito.....	25
8.4.1 Instalação do Docker.....	25
8.4.2 Executar o Docker.....	25
8.4.3 Criar uma Imagem Docker.....	26
8.4.4 Construir a imagem Docker.....	26
8.4.5 Executar um Contêiner Docker.....	26
8.4.6 Gerenciamento de Contêineres e Imagens.....	26
8.5 Tipos de Docker.....	27
8.5.1 Docker Engine.....	27
8.5.2 Docker Image.....	27
8.5.3 Docker Container.....	27
8.5.4 Dockerfile.....	28
8.5.5 Docker Hub.....	28
8.5.6 Docker Compose.....	28
8.5.7 Docker Swarm.....	28
9 NoSQL.....	29
9.1 Definições e Características.....	29
9.1.1 Escalabilidade Horizontal.....	29
9.1.2 Alta Disponibilidade e Tolerância a Falhas.....	29

9.1.3 Desempenho Otimizado para Carga de Trabalho Específica.....	29
9.1.4 Suporte a Modelos de Dados Diversificados.....	29
9.1.5 Facilidade de Escalabilidade e Desenvolvimento Ágil.....	30
9.2 Usos.....	30
9.2.1 Aplicações Web Escaláveis.....	30
9.2.2 Análise de Big Data.....	30
9.2.3 Aplicações em Tempo Real.....	30
9.2.4 Gestão de Conteúdo.....	30
9.2.5 IoT.....	31
9.3 Importância.....	31
9.3.1 Flexibilidade de Modelagem de Dados.....	31
9.3.2 Escalabilidade Horizontal.....	31
9.3.3 Alta Disponibilidade e Tolerância a Falhas.....	31
9.3.4 Desempenho Otimizado.....	31
9.3.5 Suporte a Diversos Casos de Uso.....	32
9.4 Como pode ser feito.....	32
9.4.1 Instalação Local.....	32
9.4.2 Implantação em Nuvem Pública.....	32
9.4.3 Contêineres Docker.....	33
9.4.4 Orquestras de Contêineres.....	33
9.4.5 Frameworks de Desenvolvimento.....	33
9.5 Tipos.....	33
9.5.1 Documentos.....	34
9.5.2 Chave-Valor.....	34
9.5.3 Coluna Larga.....	34
9.5.4 Grafos.....	34
9.5.5 Orientados a Objetivos.....	34
10 Banco de Dados de Documentos.....	36
10.1 Componentes principais.....	36
10.1.1 Documento.....	36
10.1.2 Coleção.....	37
10.1.3 Banco de Dados.....	37
10.1.4 Chave Primária.....	37
10.1.5 Operações CRUD.....	37
10.1.6 Consultas e Índices.....	37
10.1.7 Replicação e Escalabilidade.....	38
10.2 Tipos de Banco de Dados em Documento.....	38
10.2.1 MongoDB.....	38
10.2.2 Couchbase.....	38
10.2.3 CouchDB.....	38
10.2.4 Firebase Firestore.....	39
10.2.5 Amazon DynamoDB.....	39
10.2.6 RethinkDB.....	39
10.3 Usos.....	39
10.3.1 Desenvolvimento Web e Aplicativos Móveis.....	39
10.3.2 Gestão de Conteúdo.....	40

10.3.3	Análise de Big Data.....	40
10.3.4	Gestão de Sistemas de IoT.....	40
10.3.5	Aplicações de E-Commerce.....	40
10.3.6	Aplicações de Jogos.....	40
10.4	Vantagens.....	41
10.4.1	Flexibilidade de Esquema.....	41
10.4.2	Modelagem de Dados Intuitiva.....	41
10.4.3	Escalabilidade Horizontal.....	41
10.4.4	Desempenho.....	41
10.4.5	Agilidade no Desenvolvimento.....	42
10.4.6	Suporte a Consultas Complexas.....	42
10.4.7	Integração com Linguagens de Programação.....	42
10.5	Desvantagens.....	42
10.5.1	Menor Suporte a Transações Complexas.....	42
10.5.2	Desempenho de Consulta em Estruturação Profunda.....	43
10.5.3	Consumo de Espaços de Armazenamento.....	43
10.5.4	Maior Complexidade em Casos de Uso Relacionais.....	43
10.5.5	Falta de Padronização.....	43
10.5.6	Curva de Aprendizado para Desenvolvedores.....	43
10.6	MongoDB.....	44
10.6.1	Arquitetura.....	44
10.6.1.1	Modelo de Dados Orientado a Documentos.....	44
10.6.1.2	Sharding.....	44
10.6.1.3	Replicação.....	44
10.6.1.4	Balanceamento de Carga Automático.....	45
10.6.1.5	Arquitetura Cliente-Servidor.....	45
10.6.2	Características.....	45
10.6.2.1	Flexibilidade do Esquema .....	45
10.6.2.2	Consultas Poderosas .....	45
10.6.2.3	Indexação.....	45
10.6.2.4	Suporte a Transações.....	46
10.6.2.5	Escalabilidade Horizontal.....	46
10.6.2.6	Documentação Rica.....	46
10.6.3	Linguagem Python.....	46
	Referências bibliográficas.....	48

## 1 Introdução

Na era da tecnologia da informação, a eficiente modelagem e gestão de dados desempenham um papel crucial no desenvolvimento de aplicativos e sistemas robustos e escaláveis. Em meio à vasta gama de opções de bancos de dados disponíveis, os bancos de dados de documentos surgem como uma solução poderosa para lidar com a representação e consulta de dados que possuem estruturas flexíveis e complexas.

Neste contexto, o MongoDB destaca-se como uma das principais escolhas quando se trata de bancos de dados de documentos. Sua arquitetura flexível oferece uma solução dinâmica para o armazenamento e consulta de dados, além de uma série de recursos avançados que impulsionam a eficiência e a escalabilidade.

Este artigo mergulha no mundo do MongoDB, explorando conceitos, definições e aplicações de tecnologias fundamentais, como Dicionário de Dados e tratamento de dados. Além disso, examina os diferentes tipos de modelagens de dados e suas estruturas, proporcionando uma visão clara e objetiva sobre a tecnologia Docker e os princípios essenciais por trás dos bancos de dados de documentos.

Por meio de uma análise detalhada do MongoDB, abordaremos sua arquitetura, características distintivas e discutiremos as vantagens e desvantagens dessa tecnologia inovadora. Ao final deste artigo, você terá uma compreensão mais profunda do papel vital que o MongoDB desempenha no panorama da tecnologia de dados, e como ele pode ser uma solução valiosa para suas necessidades de modelagem e gerenciamento de dados.

## **1.1 Finalidade**

Este artigo tem como objetivo oferecer uma visão abrangente dos bancos de dados de documentos, com destaque especial para o MongoDB, reconhecido como um dos principais líderes nesse segmento. Ao mesmo tempo, exploraremos conceitos fundamentais relacionados ao Dicionário de Dados, tratamento de dados, diversos tipos de modelagens de dados e suas estruturas, além da tecnologia Docker, entre outros.

Nosso propósito é capacitar os leitores a compreenderem os conceitos básicos do MongoDB e demonstrar como podem ser aplicados em uma variedade de cenários do mundo real. Além disso, buscamos contribuir para o desenvolvimento do conhecimento dos alunos da Uninassau 3NA, da turma de Database Application, e promover o aprendizado contínuo dos autores.

Ao final desta jornada, os leitores estarão equipados com uma compreensão sólida dos princípios e práticas essenciais do MongoDB, preparando-os para enfrentar desafios e explorar oportunidades no vasto campo da modelagem e gestão de dados.

## **1.2 Escopo**

Este artigo tem como objetivo explicar e conceituar aspectos relacionados à importância da boa gestão de dados na tecnologia, com foco na utilização dos bancos de dados de documentos, especialmente o MongoDB. Além disso, será exemplificado o que é um Dicionário de Dados e sua utilidade, discutido o tratamento de dados e os diferentes tipos de modelagem de dados: conceitual, lógica e física.

Também serão abordadas noções básicas sobre Docker, oferecendo uma explicação simples sobre a estrutura de documentos, os campos e suas relações. Por fim, será explorada a tecnologia MongoDB, incluindo o funcionamento, a linguagem de consulta e manipulação de dados chamada MongoDB Query Language (MQL), bem como sua escalabilidade e flexibilidade na gestão de grandes volumes de dados.

## **1.3 Visão Geral**



Este artigo aborda a importância da eficiente gestão de dados no mundo da tecnologia, com destaque para os bancos de dados de documentos, especialmente o MongoDB. Exploraremos também conceitos fundamentais, como Dicionário de Dados e tratamento de dados, além dos diferentes tipos de modelagem de dados.

A tecnologia Docker será introduzida de forma simples, juntamente com uma explicação básica sobre a estrutura de documentos, campos e suas relações. O MongoDB será analisado em detalhes, incluindo sua arquitetura, funcionamento e as principais características que o tornam uma opção líder no mercado de bancos de dados de documentos. Também discutiremos a linguagem de consulta e manipulação de dados utilizada no MongoDB, assim como sua escalabilidade e flexibilidade.

O objetivo do trabalho é capacitar os leitores a compreenderem esses conceitos e como aplicá-los em cenários do mundo real, fornecendo uma visão abrangente sobre a tecnologia de banco de dados de documentos e seus benefícios no desenvolvimento de aplicativos e sistemas escaláveis e robustos.

## **2 Tratamento e Normalização de dados**

Processamento de dados, também conhecido como tratamento de dados, refere-se ao conjunto de procedimentos aplicados aos dados com o objetivo de torná-los mais úteis, relevantes e acessíveis para análise, tomada de decisão e outras finalidades específicas. Este processo engloba atividades como limpeza, transformação, análise, visualização e interpretação dos dados. No contexto do MongoDB, essas operações podem incluir a manipulação dos documentos para assegurar a consistência dos dados, formatos adequados e preparação para consultas eficientes e análises significativas.

### **2.1 Definições**

#### **2.1.1 Limpeza de Dados**

Processo de identificar e corrigir erros, inconsistências e valores ausentes nos documentos.

#### **2.1.2 Transformação de Dados**

Modificação da estrutura ou formato dos documentos para atender aos requisitos específicos de uma aplicação ou análise no MongoDB.

#### **2.1.3 Análise de Dados**

Exploração dos documentos para identificar padrões, tendências, correlações ou insights relevantes.

#### **2.1.4 Visualização de Dados**

Representação gráfica dos documentos para facilitar a compreensão e interpretação dos padrões e tendências no MongoDB.

#### **2.1.5 Interpretação de Dados**

Processo de extrair significado e insights dos documentos para suportar a tomada de decisões informadas no MongoDB."

## **2.2 Aplicações**

### **2.2.1 Tomada de Decisões**

Os dados tratados são fundamentais para embasar decisões estratégicas, operacionais e táticas em organizações que utilizam o MongoDB.

### **2.2.2 Análise de Negócios**

Possibilitam às empresas compreender melhor o desempenho, os padrões de consumo, as preferências dos clientes e as tendências de mercado por meio da análise dos documentos.

### **2.2.3 Pesquisa Científica**

Facilitam a análise de dados experimentais, observacionais e quantitativos em diversas áreas da ciência, utilizando os recursos oferecidos pelo MongoDB.

### **2.2.4 Personalização de Serviços**

Permitem às empresas oferecer serviços e produtos personalizados com base nas preferências e comportamentos dos clientes, utilizando os dados armazenados no MongoDB.

### **2.2.5 Detecção de Fraudes**

Os dados são analisados para identificar padrões suspeitos que possam indicar atividades fraudulentas, com o suporte da tecnologia de documentos MongoDB.

### **2.2.6 Previsão e Planejamento**

Utilização de técnicas de análise de dados no MongoDB para prever tendências futuras e planejar estratégias de negócios de forma mais precisa e eficaz.

## **2.3 Métodos de Tratamento de Dados**

### **2.3.1 Limpeza e Pré-processamento**

Inclui a remoção de valores duplicados, tratamento de valores ausentes, normalização de dados e outras operações para preparar os documentos para análise e consulta no MongoDB.

### **2.3.2 Transformação e Enriquecimento**

Envolve a conversão de formatos de dados, agregação de informações, padronização e enriquecimento com dados adicionais para aprimorar a utilidade dos documentos no MongoDB.

### **2.3.3 Análise Estatística**

Utilização de técnicas estatísticas para análise exploratória dos documentos, modelagem preditiva e outras análises estatísticas no MongoDB.

### **2.3.4 Mineração de Dados**

Identificação de padrões, tendências e correlações nos documentos armazenados no MongoDB através de técnicas de mineração de dados.

### **2.3.5 Aprendizado de Máquina**

Aplicação de algoritmos de aprendizado de máquina para automatizar análises e tomadas de decisões com base nos documentos do MongoDB."

## **2.4 Tipos de Tratamento de Dados**

### **2.4.1 Tratamento de Dados Estruturados**

Refere-se ao processamento de dados organizados em documentos, seguindo uma estrutura definida semelhante às tabelas em bancos de dados relacionais.

### **2.4.2 Tratamento de Dados Não Estruturados**

Envolve o processamento de dados que não possuem uma estrutura pré-determinada, como texto, áudio, vídeo, entre outros, armazenados em documentos no MongoDB.

### **2.4.3 Tratamento de Dados Semi-Estruturados**

Diz respeito ao processamento de dados que possuem alguma estrutura, mas não se encaixam perfeitamente em um formato estruturado, como documentos XML, JSON, etc, armazenados no MongoDB.

#### **2.4.4 Tratamento de Dados em Tempo Real**

Refere-se ao processamento e análise de dados à medida que são gerados, com baixa latência, em documentos armazenados no MongoDB.

#### **2.4.5 Tratamento de Big Data**

Consiste no processamento de grandes volumes de dados que ultrapassam a capacidade de processamento dos sistemas tradicionais de gerenciamento de banco de dados, utilizando a escalabilidade e capacidade de armazenamento do MongoDB.

### **3 Dicionário de Dados**

Um dicionário de dados desempenha um papel importante na gestão eficaz de bancos de dados, oferecendo uma referência detalhada e estruturada sobre todas as informações contidas no banco de dados. Essa ferramenta essencial garante consistência, integridade e compreensão dos dados, facilitando seu uso, manutenção e apoiando as operações de negócios, além de promover uma tomada de decisões informadas.

#### **3.1 Estrutura**

O dicionário de dados vai listar como a organização do banco de dados está realizada, detalhando todas as tabelas, os campos e seus relacionamentos. Neste projeto, encontramos definições para tabelas, como: 'host', 'rental\_ad', 'address', 'reviews', 'description' e 'geolocation'.

#### **3.2 Identificação da Entidade**

Cada entrada no dicionário de dados deve identificar claramente a entidade ou objeto de dados a que se refere. Isso pode incluir o nome da tabela, coleção ou conjunto de dados em questão.

#### **3.3 Descrição dos Atributos e Tipos de Dados**

Para cada entidade, é importante descrever os atributos ou campos que compõem os dados. Isso pode incluir o nome do atributo, o tipo de dados, a descrição do que o atributo representa e quaisquer restrições ou regras aplicáveis. Como exemplo, temos, 'host', encontramos os atributos 'host\_id' (do tipo INT) e 'host\_name' (do tipo VARCHAR).

#### **3.4 Chaves Primárias e Estrangeiras**

O dicionário de dados deve identificar as chaves primárias e estrangeiras que são usadas para estabelecer relacionamentos entre diferentes entidades ou tabelas.

#### **3.5 Restrições e Regras de Negócio**

É útil incluir informações sobre quaisquer restrições ou regras de negócio que se apliquem aos dados, como restrições de integridade referencial, restrições de valor permitido (por exemplo, NOT NULL) e quaisquer outras regras específicas do domínio.

Esses são alguns dos elementos comuns encontrados na estrutura de um dicionário de dados, mas a organização e o formato exatos podem variar dependendo das necessidades específicas de cada organização ou sistema.

## **4 Modelagem de Dados**

Modelagem de dados é o processo de criar uma representação estruturada e organizada dos dados e das relações entre eles. Essa representação pode ser usada para entender, especificar e documentar os requisitos de um sistema antes de sua implementação. A modelagem de dados envolve a identificação das entidades (objetos ou conceitos do mundo real), seus atributos (características) e os relacionamentos entre as entidades.

Existem várias técnicas e abordagens para a modelagem de dados, incluindo o modelo entidade-relacionamento (ER), o modelo relacional, o modelo dimensional e o modelo de bancos de dados de documentos, entre outros. Cada modelo tem suas próprias características e é escolhido com base nos requisitos específicos do sistema e nas necessidades de armazenamento e consulta de dados.

No caso deste projeto, o modelo escolhido e executado foi o relacional. O modelo relacional é utilizado em ambientes transacionais (OLTP), pois são ambientes na qual os principais comandos utilizados são de inserção e atualizações, desta forma o modelo é normalizado a fim de evitar redundâncias de dados; essas relações se dão através de chaves que criam restrições e também gerem o relacionamento entre os atributos (campos) nas entidades (tabelas).

### **4.1 Modelo Conceitual**

Representa o domínio do negócio em um nível de abstração mais alto com o objetivo de descrever as principais entidades, relacionamentos e regras de negócio que compõem o domínio do problema.



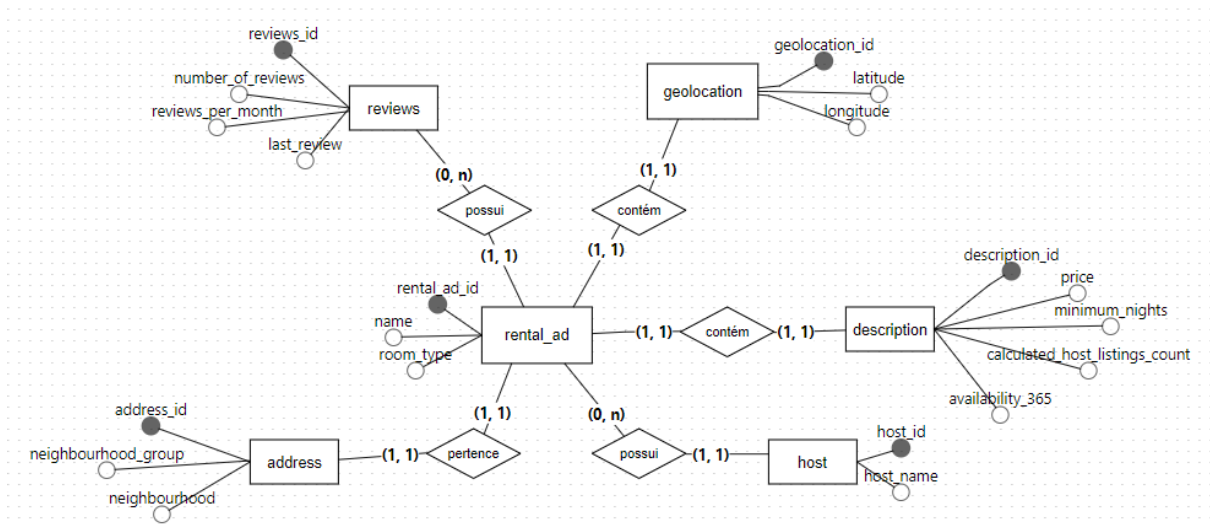


Figura 1 - Modelo Conceitual

## 4.2 Modelo Lógico

Representação de dados mais detalhadas do que na modelagem conceitual, com suas entidades, relacionamentos e atributos definidos e como estes dados serão armazenados em um banco de dados.

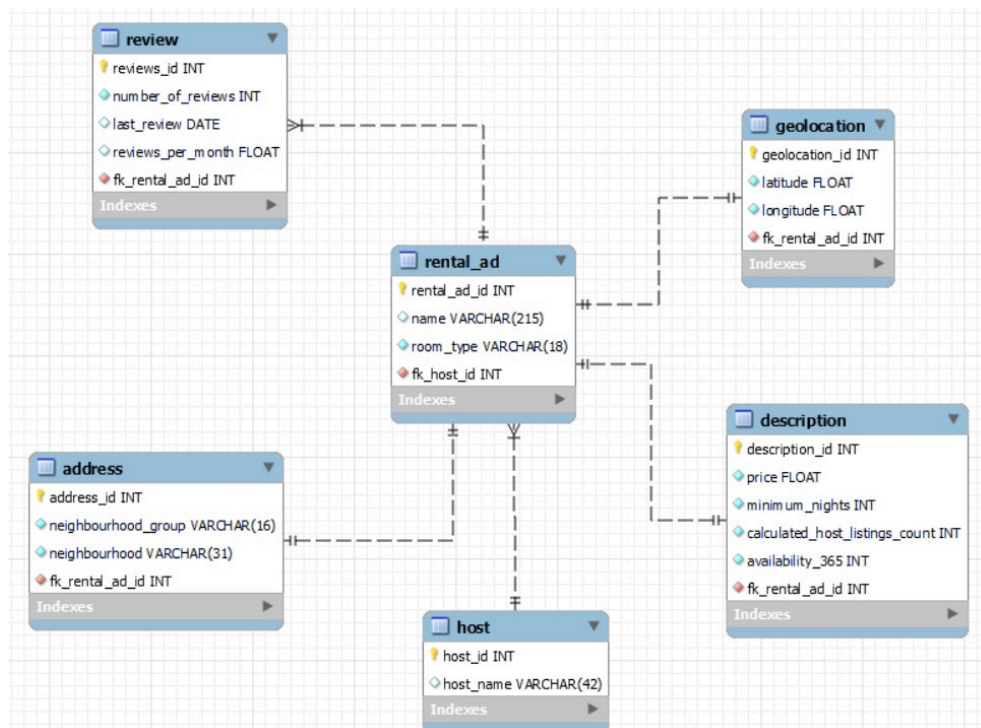
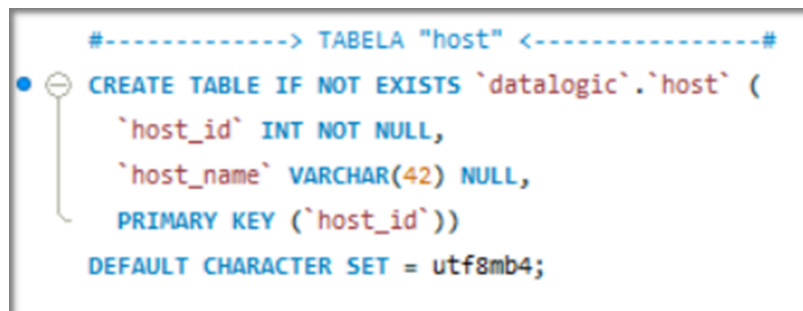


Figura 2 - Modelo Lógico

### 4.3 Modelo Físico

Representa um modelo definido para um banco de dados específico e na construção dos códigos na sua linguagem específica para criação das entidades, atributos, dependências e restrições.

The image shows a screenshot of a code editor with a light blue background. It contains SQL code for creating a table. At the top, there is a comment line: `#-----> TABELA "host" <-----#`. Below it, the main code is: `CREATE TABLE IF NOT EXISTS `datalogic`.`host` (  
 `host_id` INT NOT NULL,  
 `host_name` VARCHAR(42) NULL,  
 PRIMARY KEY (`host_id`))  
DEFAULT CHARACTER SET = utf8mb4;`. A blue cursor icon is positioned at the start of the first line of the code block.

*Figura 3 - Exemplo de Modelo Físico*

## 5 DDLs

A DDL (Data Definition Language) é uma linguagem de manipulação de dados que permite aos usuários definir e gerenciar a estrutura dos objetos de banco de dados, como tabelas, índices, visões e restrições. As principais operações realizadas por DDL incluem a criação, alteração e exclusão desses objetos.

### 5.1 Principais exemplos de DDLs

#### 5.1.1 CREATE TABLE

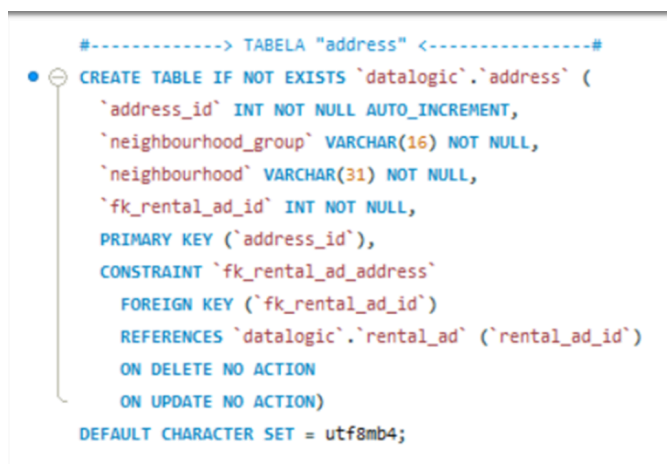
Usada para criar uma nova tabela.

#### 5.1.2 ALTER TABLE

Usada para modificar a estrutura de uma tabela existente.

#### 5.1.3 DROP TABLE

Usada para excluir uma tabela existente do banco de dados.



```
#-----> TABELA "address" <-----#
CREATE TABLE IF NOT EXISTS `datalogic`.`address` (
  `address_id` INT NOT NULL AUTO_INCREMENT,
  `neighbourhood_group` VARCHAR(16) NOT NULL,
  `neighbourhood` VARCHAR(31) NOT NULL,
  `fk_rental_ad_id` INT NOT NULL,
  PRIMARY KEY (`address_id`),
  CONSTRAINT `fk_rental_ad_address`
    FOREIGN KEY (`fk_rental_ad_id`)
      REFERENCES `datalogic`.`rental_ad` (`rental_ad_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
DEFAULT CHARACTER SET = utf8mb4;
```

*Figura 4 - Comando DDL*

## 6 DMLs

A DML (Data Manipulation Language) é usada para gerenciar os dados dentro do banco de dados. Ao contrário do DDL, que se concentra na estrutura dos objetos do banco de dados, o DML se concentra na manipulação dos próprios dados. As principais operações realizadas por DML incluem inserção, consulta, atualização e exclusão de dados.

### 6.1 Principais exemplos de DMLs

#### 6.1.1 INSERT INTO

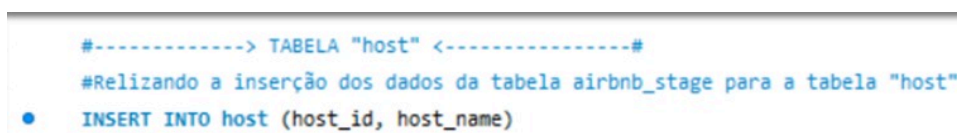
Usada para inserir novos registros na tabela.

#### 6.1.2 UPDATE

Usada para atualizar registros existentes na tabela com base em critérios especificados.

#### 6.1.3 DELETE

Usada para excluir registros da tabela com base em critérios especificados.



```
#-----> TABELA "host" <-----#  
#Realizando a inserção dos dados da tabela airbnb_stage para a tabela "host"  
• INSERT INTO host (host_id, host_name)
```

*Figura 5 - Comandos DMLs*

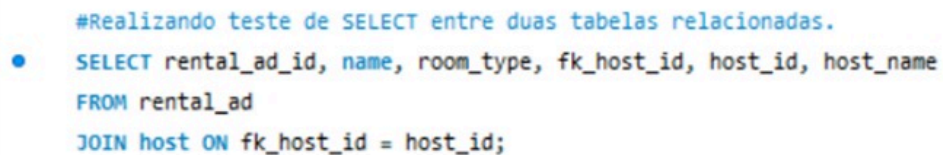
## 7 DQLs

O DQL (Data Query Language) é usado especificamente para fazer consultas em um banco de dados para recuperar informações. Em bancos de dados relacionais, consultas são comumente feitas usando a linguagem SQL (Structured Query Language). Portanto, em essência, consultas SQL são um exemplo de DQL.

### 7.1 Exemplo de DQL

#### 7.1.1 SELECT

Usada para recuperar dados de um banco de dados. É uma das declarações mais fundamentais e comuns na linguagem SQL.



```
#Realizando teste de SELECT entre duas tabelas relacionadas.  
● SELECT rental_ad_id, name, room_type, fk_host_id, host_id, host_name  
FROM rental_ad  
JOIN host ON fk_host_id = host_id;
```

*Figura 6 - Comandos DQL*

## 8 Docker

O Docker é uma plataforma de código aberto que permite a criação, o empacotamento e a execução de aplicativos em contêineres. Um contêiner é uma unidade de software leve e autossuficiente que contém tudo o que é necessário para executar um aplicativo, incluindo o código, as bibliotecas, as dependências e as configurações. O Docker fornece uma maneira consistente e confiável de criar, distribuir e executar aplicativos em diferentes ambientes, garantindo que o aplicativo funcione da mesma forma em qualquer lugar onde seja executado.

### 8.1 Definições

#### 8.1.1 Contêineres

São instâncias isoladas e autônomas de aplicativos que são executadas no sistema operacional host. Os contêineres são portáteis, leves e podem ser facilmente transferidos entre diferentes ambientes de desenvolvimento, teste e produção.

#### 8.1.2 Imagens

São pacotes autossuficientes que contêm o código-fonte, as bibliotecas, as dependências e as configurações necessárias para executar um aplicativo em um contêiner. As imagens são criadas a partir de arquivos Dockerfile e podem ser compartilhadas e reutilizadas em diferentes projetos e ambientes.

#### 8.1.3 Dockerfile

É um arquivo de texto simples que contém instruções para construir uma imagem Docker. Ele define os passos necessários para configurar e preparar o ambiente de execução do aplicativo dentro do contêiner.

#### 8.1.4 Docker Engine

É o componente do Docker responsável por criar, executar e gerenciar contêineres. O Docker Engine inclui um daemon de serviço

(dockerd) que executa em segundo plano e uma interface de linha de comando (docker) que permite interagir com contêineres e imagens.

#### **8.1.5 Docker Compose**

É uma ferramenta que permite definir e gerenciar aplicativos Docker multi contêiner em um único arquivo YAML. Ele simplifica o processo de configuração de ambientes complexos e a coordenação entre contêineres interdependentes.

Em resumo, o Docker é uma tecnologia de virtualização de contêineres que simplifica o desenvolvimento, a distribuição e a execução de aplicativos, fornecendo uma maneira eficiente e padronizada de empacotar e implantar software em diferentes ambientes de computação.

### **8.2 Usos**

Os usos e a importância do Docker são diversos e abrangentes, refletindo sua popularidade e ampla adoção na comunidade de desenvolvimento de software.

#### **8.2.1 Padronização de Ambientes de Desenvolvimento**

O Docker permite que os desenvolvedores criem ambientes de desenvolvimento consistentes e replicáveis, garantindo que todos os membros da equipe trabalhem com as mesmas configurações de software e dependências.

#### **8.2.2 Implantação de Aplicativos em Micro Serviços**

O Docker é frequentemente usado para implantar aplicativos em arquiteturas de micro serviços, onde cada serviço é empacotado e executado em seu próprio contêiner. Isso facilita a escalabilidade, a manutenção e a atualização de aplicativos complexos.

#### **8.2.3 Implantação em Nuvem e Orquestração de Contêineres**

O Docker é amplamente usado em ambientes de nuvem para implantar e gerenciar aplicativos em contêineres. Ferramentas como

Kubernetes, Docker Swarm e Amazon ECS permitem a orquestração eficiente de contêineres em escala, garantindo alta disponibilidade e escalabilidade.

#### **8.2.4 Integração Contínua e Implantação Contínua (CI/CD)**

O Docker é frequentemente usado em pipelines de CI/CD para automatizar a construção, teste e implantação de aplicativos. Os contêineres Docker fornecem ambientes isolados e previsíveis para executar testes de integração e implantações automatizadas.

#### **8.2.5 Desenvolvimento de Aplicativos Multiplataforma**

Com o Docker, os desenvolvedores podem criar aplicativos que são independentes do sistema operacional e das configurações de ambiente, permitindo que sejam executados em qualquer plataforma compatível com Docker, como Windows, Linux e macOS.

### **8.3 Importância**

#### **8.3.1 Consistência e Reprodutibilidade**

O Docker fornece ambientes de desenvolvimento e implantação consistentes e reprodutíveis, garantindo que os aplicativos funcionem da mesma maneira em diferentes ambientes.

#### **8.3.2 Eficiência e Economia de Recursos**

Os contêineres Docker são leves e compartilham recursos do sistema operacional host, o que os torna mais eficientes em termos de recursos e economiza custos de infraestrutura em comparação com máquinas virtuais tradicionais.

#### **8.3.3 Flexibilidade e Portabilidade**

Os contêineres Docker são portáteis e podem ser facilmente movidos entre diferentes ambientes de desenvolvimento, teste e produção. Isso



oferece flexibilidade aos desenvolvedores e permite a implantação consistente em vários ambientes de nuvem e locais.

#### **8.3.4 Escalabilidade e Resiliência**

O Docker facilita a escalabilidade horizontal de aplicativos por meio da implantação de contêineres replicáveis e orquestração eficiente em clusters de contêineres. Isso garante alta disponibilidade e resiliência de aplicativos em ambientes de produção.

#### **8.3.5 Facilidade de Gerenciamento e Manutenção**

A utilização de contêineres Docker simplifica o gerenciamento e a manutenção de aplicativos, reduzindo a complexidade operacional e permitindo implantações rápidas e rollbacks simples.

Dessa forma, o Docker desempenha um papel crucial na modernização do desenvolvimento de software e operações de TI, fornecendo uma maneira eficiente, consistente e escalável de empacotar, distribuir e executar aplicativos em ambientes de computação diversificados.

### **8.4 Como pode ser feito**

A implementação do Docker pode ser realizada seguindo algumas etapas básicas. Aqui está uma visão geral de como o Docker pode ser configurado e usado:

#### **8.4.1 Instalação do Docker**

O primeiro passo é instalar o Docker no seu sistema operacional. O Docker fornece versões para Windows, macOS e Linux. Você pode baixar e instalar o Docker Desktop para Windows e macOS a partir do site oficial do Docker, ou seguir as instruções específicas para sua distribuição Linux.

#### **8.4.2 Executar o Docker**

Após a instalação, inicie o Docker Desktop ou o serviço Docker no seu sistema operacional. Isso iniciará o daemon do Docker, que é responsável por gerenciar contêineres e imagens Docker.

#### **8.4.3 Criar uma Imagem Docker**

Crie uma imagem Docker usando um arquivo Dockerfile. O Dockerfile é um arquivo de texto simples que contém instruções para construir uma imagem Docker. Ele define o ambiente de execução do aplicativo dentro do contêiner, incluindo o sistema operacional base, as dependências e as configurações necessárias.

Você pode criar um Dockerfile no diretório do seu aplicativo e especificar as instruções necessárias, como a escolha da imagem base, a instalação de pacotes e a configuração do ambiente de execução.

#### **8.4.4 Construir a Imagem Docker**

Use o comando ``docker build`` para construir a imagem Docker a partir do Dockerfile. Este comando irá ler o Dockerfile, executar as instruções especificadas e criar uma imagem Docker com base nessas instruções.

Por exemplo: ``docker build -t nome_da_imagem .``, onde ``-t`` especifica o nome da imagem e ``.`` indica o diretório atual como o contexto de construção.

#### **8.4.5 Executar um Contêiner Docker**

Após construir a imagem Docker, você pode executar um contêiner Docker usando o comando ``docker run``. Este comando cria uma instância do contêiner a partir da imagem especificada e a executa.

Por exemplo: ``docker run -d -p 8080:80 nome_da_imagem``, onde ``-d`` indica que o contêiner será executado em segundo plano, ``-p`` mapeia a porta do contêiner para a porta do host e ``nome_da_imagem`` é o nome da imagem Docker a ser usada.

#### **8.4.6 Gerenciamento de Contêineres e Imagens**

Você pode usar uma variedade de comandos Docker para gerenciar contêineres e imagens. Alguns comandos comuns incluem ``docker ps`` para listar contêineres em execução, ``docker stop`` para parar um contêiner em execução, ``docker rm`` para remover um contêiner, ``docker images`` para listar imagens Docker no seu sistema, entre outros.

Essas são apenas algumas das etapas básicas para configurar e usar o Docker. À medida que você se familiariza com o Docker, pode explorar recursos mais avançados, como volumes, redes, composição com o Docker Compose e orquestração de contêineres com ferramentas como o Kubernetes.

## **8.5 Tipos de Docker**

Existem vários tipos de Docker que se referem a diferentes aspectos e recursos da plataforma Docker. Os tipos mais comuns de Docker são:

### **8.5.1 Docker Engine**

O Docker Engine é o componente principal do Docker que permite a criação, execução e gerenciamento de contêineres Docker em um sistema operacional host. Ele inclui o daemon de serviço ``dockerd`` e a interface de linha de comando ``docker``.

### **8.5.2 Docker Image**

Uma imagem Docker é um pacote auto suficiente que contém tudo o que é necessário para executar um aplicativo, incluindo o código, as bibliotecas, as dependências e as configurações. As imagens Docker são criadas a partir de um Dockerfile e podem ser compartilhadas e reutilizadas em diferentes ambientes.

### **8.5.3 Docker Container**

Um contêiner Docker é uma instância em execução de uma imagem Docker. Ele fornece um ambiente isolado e auto suficiente para executar um aplicativo, garantindo que todas as suas dependências sejam atendidas e que ele funcione de maneira consistente em diferentes ambientes.

#### **8.5.4 Dockerfile**

Um Dockerfile é um arquivo de texto simples que contém instruções para construir uma imagem Docker. Ele define o ambiente de execução do aplicativo dentro do contêiner, incluindo o sistema operacional base, as dependências e as configurações necessárias.

#### **8.5.5 Docker Hub**

O Docker Hub é um serviço de registro de imagens Docker hospedado pela Docker, Inc. Ele permite que os usuários armazenem, compartilhem e distribuam imagens Docker publicamente ou de forma privada. O Docker Hub também é uma fonte de imagens Docker pré-construídas disponíveis para uso.

#### **8.5.6 Docker Compose**

O Docker Compose é uma ferramenta que permite definir e gerenciar aplicativos Docker multi contêiner em um único arquivo YAML. Ele simplifica o processo de configuração de ambientes complexos e a coordenação entre contêineres interdependentes.

#### **8.5.7 Docker Swarm**

Docker Swarm é uma ferramenta de orquestração de contêineres integrada ao Docker Engine. Ele permite que os usuários criem e gerenciem clusters de contêineres Docker para escalabilidade e alta disponibilidade de aplicativos.

Esses são alguns dos tipos fundamentais de Docker que são amplamente utilizados na plataforma Docker. Cada um desempenha um papel importante na criação, execução e gerenciamento de aplicativos em contêineres Docker.

## 9 NoSQL

O NoSQL, que significa "Not Only SQL" (Não Apenas SQL), é uma abordagem alternativa ao modelagem, armazenamento e recuperação de dados que difere dos tradicionais sistemas de gerenciamento de banco de dados relacionais (RDBMS). O termo "NoSQL" foi apresentado para descrever os bancos de dados que não seguem o modelo relacional clássico, oferecendo uma variedade de modelos de dados e métodos de consulta.

### 9.1 Definições e Características

#### 9.1.1 Escalabilidade Horizontal

Muitos bancos de dados NoSQL são projetados para escalar horizontalmente, o que significa que eles podem lidar com grandes volumes de dados distribuindo o armazenamento e o processamento entre vários servidores. Isso os torna ideais para aplicativos que precisam lidar com grandes volumes de dados e / ou alta carga de usuários.

#### 9.1.2 Alta Disponibilidade e Tolerância a Falhas

NoSQL geralmente oferece recursos integrados de replicação e distribuição para garantir alta disponibilidade e tolerância a falhas. Isso permite que os sistemas NoSQL continuem funcionando mesmo em caso de falha de hardware ou rede.

#### 9.1.3 Desempenho Otimizado para Carga de Trabalho Específica

Bancos de dados NoSQL são frequentemente otimizados para cargas de trabalho específicas, como armazenamento e recuperação de dados em tempo real, análise de big data, processamento de transações de alta velocidade, entre outros.

#### 9.1.4 Suporte a Modelos de Dados Diversificados

NoSQL suporta uma variedade de modelos de dados, incluindo documentos, grafos, pares chave-valor e colunas amplas. Isso permite que

os desenvolvedores escolham o modelo de dados mais adequado para seus aplicativos específicos.

#### **9.1.5 Facilidade de Escalabilidade e Desenvolvimento Ágil**

Com sua flexibilidade de esquema e capacidade de escalar horizontalmente, os bancos de dados NoSQL permitem que os desenvolvedores reitem rapidamente em seus aplicativos e se adaptem facilmente às mudanças nos requisitos de dados.

### **9.2 Usos**

#### **9.2.1 Aplicações Web Escaláveis**

Bancos de dados NoSQL são frequentemente usados em aplicativos da web que precisam lidar com grandes volumes de dados e alta concorrência. Eles podem escalar horizontalmente para atender às demandas crescentes de usuários e dados.

#### **9.2.2 Análise de Big Data**

NoSQL é amplamente utilizado em sistemas de análise de big data devido à sua capacidade de armazenar e processar grandes volumes de dados de forma eficiente. Eles são ideais para lidar com dados não estruturados e semi-estruturados em ambientes de análise de dados em tempo real.

#### **9.2.3 Aplicações em Tempo Real**

Bancos de dados NoSQL são frequentemente usados em aplicativos que exigem baixa latência e alta disponibilidade, como sistemas de mensagens, sistemas de monitoramento e análise de streaming em tempo real.

#### **9.2.4 Gestão de Conteúdo**

Muitos sistemas de gerenciamento de conteúdo (CMS), plataformas de comércio eletrônico e aplicativos de mídia social usam bancos de dados

NoSQL para armazenar e recuperar grandes volumes de dados, como postagens de blog, produtos e mídia.

### **9.2.5 IoT (Internet das Coisas)**

Bancos de dados NoSQL são usados em sistemas de IoT para armazenar e analisar dados de dispositivos conectados, sensores e outros dispositivos de IoT em tempo real.

## **9.3 Importância**

### **9.3.1 Flexibilidade de Modelagem de Dados**

Bancos de dados NoSQL oferecem flexibilidade na modelagem de dados, permitindo que os desenvolvedores trabalhem com uma variedade de modelos de dados, como documentos, grafos, pares chave-valor e colunas largas.

### **9.3.2 Escalabilidade Horizontal**

NoSQL é altamente escalável horizontalmente, o que significa que pode lidar com grandes volumes de dados distribuindo o armazenamento e o processamento entre vários servidores. Isso permite que os sistemas NoSQL dimensionem para atender às demandas crescentes de dados e usuários.

### **9.3.3 Alta Disponibilidade e Tolerância a Falhas**

Muitos bancos de dados NoSQL oferecem recursos integrados de replicação e distribuição para garantir alta disponibilidade e tolerância a falhas. Isso garante que os sistemas NoSQL permaneçam operacionais mesmo em caso de falha de hardware ou rede.

### **9.3.4 Desempenho Otimizado**

NoSQL é frequentemente otimizado para cargas de trabalho específicas, como armazenamento e recuperação de dados em tempo real, análise de big data e processamento de transações de alta velocidade.

### **9.3.5 Suporte a Diversos Casos de Uso**

Bancos de dados NoSQL suportam uma variedade de casos de uso, desde aplicativos da web escaláveis até análise de big data e sistemas de IoT. Sua flexibilidade os torna adequados para uma ampla gama de aplicativos e cargas de trabalho.

Dessa forma, os bancos de dados NoSQL desempenham um papel importante na infraestrutura de dados moderna, oferecendo flexibilidade, escalabilidade e desempenho otimizado para uma variedade de aplicativos e casos de uso em um mundo de dados em constante mudança e crescimento.

## **9.4 Como pode ser feito**

A implementação de bancos de dados NoSQL pode ser feita de várias maneiras, dependendo das necessidades específicas do projeto e das preferências da equipe de desenvolvimento. Aqui estão algumas abordagens comuns para implementar bancos de dados NoSQL:

### **9.4.1 Instalação Local**

Você pode configurar e executar um banco de dados NoSQL em um servidor local ou em uma máquina virtual. Isso é útil para desenvolvimento e testes locais, e geralmente envolve baixar e instalar o software do banco de dados NoSQL correspondente.

### **9.4.2 Implantação em Nuvem Pública**

Muitos provedores de nuvem oferecem serviços gerenciados de bancos de dados NoSQL, onde você pode provisionar e implantar instâncias do banco de dados na nuvem. Exemplos incluem Amazon DynamoDB, Google Cloud Firestore e Azure Cosmos DB. Esses serviços geralmente lidam com tarefas de gerenciamento, como escalabilidade, disponibilidade e backup, permitindo que você se concentre no desenvolvimento de aplicativos.



### **9.4.3 Contêineres Docker**

Você pode usar contêineres Docker para implantar e gerenciar instâncias de bancos de dados NoSQL. Isso oferece flexibilidade e portabilidade, permitindo que você implante o banco de dados em qualquer ambiente compatível com Docker, como desenvolvimento local, nuvem ou data center.

### **9.4.4 Orquestradores de Contêineres**

Ferramentas de orquestração de contêineres, como Kubernetes, podem ser usadas para implantar e gerenciar clusters de contêineres que executam bancos de dados NoSQL em escala. Isso é útil para implantações de produção de alto desempenho e alta disponibilidade.

### **9.4.5 Frameworks de Desenvolvimento**

Alguns frameworks de desenvolvimento, como o Apache Cassandra para armazenamento de grande escala distribuído ou o MongoDB para armazenamento de documentos flexíveis, podem ser integrados diretamente em aplicativos para fornecer funcionalidades de banco de dados NoSQL sem a necessidade de implantação separada.

Independentemente do método escolhido, é importante considerar fatores como desempenho, escalabilidade, disponibilidade, segurança e facilidade de manutenção ao implementar bancos de dados NoSQL em um projeto. Além disso, é crucial entender os requisitos específicos do aplicativo e escolher o banco de dados NoSQL mais adequado para atender a esses requisitos.

## **9.5 Tipos**

Existem vários tipos de bancos de dados NoSQL, cada um projetado para atender a diferentes necessidades de modelagem de dados e casos de uso específicos. Os principais tipos de bancos de dados NoSQL incluem:

### **9.5.1 Documentos**

Armazenam dados em documentos semelhantes a JSON, onde cada documento pode ter estruturas diferentes.

Exemplos populares incluem MongoDB, Couchbase e CouchDB.

### **9.5.2 Chave-Valor**

Armazenam dados em um formato simples de chave-valor, onde cada valor é associado a uma chave única. São eficientes para operações simples de armazenamento e recuperação.

Exemplos incluem Redis, Riak e DynamoDB.

### **9.5.3 Coluna Larga (Wide Column)**

Armazenam dados em uma estrutura de colunas, onde os dados são agrupados por coluna em vez de linha. São eficientes para consultas analíticas e agregações em grande escala.

Exemplos incluem Apache Cassandra, HBase e ScyllaDB.

### **9.5.4 Grafos**

Armazenam dados na forma de grafos, com nós representando entidades e arestas representando relacionamentos entre elas. São eficientes para consultas que envolvem relacionamentos complexos entre entidades.

Exemplos incluem Neo4j, Amazon Neptune e ArangoDB.

### **9.5.5 Orientado a Objetos**

Armazenam dados de forma semelhante a objetos em linguagens de programação, preservando as relações de herança e composição. São eficientes para aplicativos orientados a objetos e mapeamento objeto-relacional. Exemplos incluem db4o e ObjectDB.

Esses são os tipos principais de bancos de dados NoSQL, cada um com suas próprias características, vantagens e casos de uso específicos. A escolha do tipo certo de banco de dados NoSQL depende das necessidades e requisitos específicos do projeto,

incluindo o modelo de dados, a escalabilidade, o desempenho e a consistência necessários.

## **10 Banco de Dados de Documentos**

Um banco de dados de documentos é um tipo de banco de dados NoSQL (Not Only SQL) que armazena e organiza dados em documentos semiestruturados ou estruturados, geralmente no formato JSON (JavaScript Object Notation), BSON (Binary JSON) ou similar. Ao contrário dos bancos de dados relacionais, que organizam dados em tabelas com linhas e colunas, os bancos de dados de documentos permitem que os dados sejam armazenados em documentos individuais, que podem conter campos e valores variados e aninhados.

Cada documento em um banco de dados de documentos é identificado por uma chave exclusiva e pode conter dados relacionados a um único item ou entidade. Os documentos são flexíveis e podem ser facilmente atualizados e expandidos sem a necessidade de alterar a estrutura do banco de dados.

Os bancos de dados de documentos são adequados para uma ampla variedade de casos de uso, incluindo aplicativos da web, aplicativos móveis, análise de big data, IoT (Internet das Coisas) e muito mais. Eles oferecem escalabilidade horizontal, capacidade de lidar com dados semiestruturados e flexibilidade no esquema de dados, o que os torna uma escolha popular para aplicativos modernos que lidam com volumes variados e dinâmicos de dados.

Exemplos populares de bancos de dados de documentos incluem MongoDB, Couchbase, CouchDB e Firebase Firestore. Esses sistemas de banco de dados são amplamente utilizados por empresas e desenvolvedores para armazenar, recuperar e manipular dados de forma eficiente e escalável em uma variedade de aplicativos e ambientes.

### **10.1 Componentes principais**

Os componentes principais de um banco de dados de documentos geralmente incluem:

#### **10.1.1 Documento**

O documento é a unidade básica de armazenamento em um banco de dados de documentos. Ele é representado em um formato semiestruturado, como JSON ou BSON, e contém os dados relacionados a uma entidade específica. Os documentos podem conter campos e valores variados, incluindo strings, números, arrays, objetos aninhados e outros tipos de dados.

### **10.1.2 Coleção**

Uma coleção é um agrupamento lógico de documentos dentro de um banco de dados de documentos. Cada coleção pode conter um conjunto de documentos relacionados que compartilham um esquema semelhante ou pertencem a um determinado tipo de entidade. As coleções são análogas a tabelas em bancos de dados relacionais.

### **10.1.3 Banco de Dados**

Um banco de dados de documentos pode conter múltiplas coleções e é usado para organizar e gerenciar os dados em um ambiente isolado. Cada banco de dados pode ter suas próprias permissões de acesso e configurações de segurança.

### **10.1.4 Chave Primária**

Em um banco de dados de documentos, cada documento é identificado por uma chave única chamada de chave primária. Essa chave é usada para recuperar, atualizar ou excluir documentos específicos de uma coleção.

### **10.1.5 Operações CRUD**

As operações CRUD (Create, Read, Update, Delete) são usadas para interagir com os dados em um banco de dados de documentos. Isso inclui a criação de novos documentos, a recuperação de documentos existentes, a atualização de documentos existentes e a exclusão de documentos.

### **10.1.6 Consultas e Índices**

Os bancos de dados de documentos geralmente suportam consultas flexíveis para recuperar dados com base em critérios específicos. Eles também podem usar índices para melhorar o desempenho de consultas e operações de busca.

#### **10.1.7 Replicação e Escalabilidade**

Muitos bancos de dados de documentos suportam replicação de dados para garantir alta disponibilidade e tolerância a falhas. Eles também oferecem recursos de escalabilidade horizontal para lidar com cargas de trabalho crescentes, distribuindo dados entre vários servidores.

Esses são alguns dos componentes principais comuns em um banco de dados de documentos. No entanto, a implementação exata e os recursos disponíveis podem variar de um sistema de banco de dados para outro.

### **10.2 Tipos de Banco de Dados em Documento**

Existem vários tipos de bancos de dados de documentos disponíveis atualmente, cada um com suas próprias características e casos de uso específicos. Aqui estão alguns dos tipos mais populares:

#### **10.2.1 MongoDB**

MongoDB é um dos bancos de dados de documentos mais conhecidos e amplamente utilizados. Ele oferece uma estrutura flexível para armazenar e consultar documentos JSON, suportando consultas complexas, índices, replicação e escalabilidade horizontal.

#### **10.2.2 Couchbase**

Couchbase é outro banco de dados de documentos popular que oferece alta disponibilidade, escalabilidade e desempenho. Ele combina a flexibilidade de documentos JSON com a velocidade e a eficiência do armazenamento de chave-valor.

#### **10.2.3 CouchDB**

CouchDB é um banco de dados de documentos de código aberto que enfatiza a facilidade de uso, a tolerância a falhas e a replicação bidirecional. Ele é projetado para funcionar bem em ambientes distribuídos e descentralizados.

#### **10.2.4 Firebase Firestore**

Firestore é um banco de dados de documentos totalmente gerenciado que faz parte da plataforma Firebase do Google. Ele oferece armazenamento em tempo real, sincronização automática entre dispositivos e integração perfeita com outros serviços do Firebase.

#### **10.2.5 Amazon DynamoDB**

DynamoDB é um banco de dados de documentos e de valor-chave totalmente gerenciado oferecido pela Amazon Web Services (AWS). Ele fornece escalabilidade automática, baixa latência e alta disponibilidade para aplicativos de qualquer tamanho.

#### **10.2.6 RethinkDB**

RethinkDB é um banco de dados de documentos distribuído, de código aberto, que permite consultas em tempo real. Ele foi projetado para facilitar o desenvolvimento de aplicativos em tempo real, fornecendo uma API de consulta expressiva e uma estrutura de dados flexível.

Esses são apenas alguns exemplos de bancos de dados de documentos disponíveis atualmente. Cada um tem suas próprias vantagens, recursos e casos de uso específicos, e a escolha do melhor banco de dados depende dos requisitos e das necessidades do seu aplicativo.

### **10.3 Usos**

#### **10.3.1 Desenvolvimento Web e Aplicativos Móveis**

Bancos de dados de documentos são frequentemente utilizados no desenvolvimento de aplicativos da web e móveis devido à sua capacidade de lidar com dados semiestruturados e à sua flexibilidade de esquema. Eles

são ideais para armazenar dados de usuário, configurações de aplicativos, conteúdo dinâmico e muito mais.

### **10.3.2 Gestão de Conteúdo**

Sistemas de gerenciamento de conteúdo (CMS) e plataformas de blogs frequentemente usam bancos de dados de documentos para armazenar e gerenciar artigos, postagens, páginas da web e outros tipos de conteúdo digital. A flexibilidade dos documentos JSON permite que diferentes tipos de conteúdo sejam armazenados de forma eficiente.

### **10.3.3 Análise de Big Data**

Bancos de dados de documentos são utilizados em cenários de análise de big data devido à sua capacidade de armazenar e processar grandes volumes de dados semiestruturados. Eles são usados para coletar e analisar dados de sensores, logs de servidor, registros de eventos e outras fontes de dados não estruturados.

### **10.3.4 Gestão de Sistemas de IoT**

Dispositivos de Internet das Coisas (IoT) frequentemente geram grandes volumes de dados semiestruturados que precisam ser armazenados e processados de forma eficiente. Bancos de dados de documentos são utilizados para armazenar dados de sensores, dispositivos e eventos IoT, facilitando a análise e o monitoramento em tempo real.

### **10.3.5 Aplicações de E-Commerce**

Plataformas de comércio eletrônico utilizam bancos de dados de documentos para armazenar informações de produtos, inventário, pedidos e transações de forma flexível. Isso permite uma personalização eficiente, recomendações de produtos e uma experiência de compra personalizada para os usuários.

### **10.3.6 Aplicações de Jogos**



Em jogos digitais, os bancos de dados de documentos são usados para armazenar informações de perfil do jogador, progresso do jogo, conquistas desbloqueadas e outros dados relacionados ao jogo. Eles permitem uma integração suave entre dispositivos e plataformas e suportam recursos sociais e colaborativos.

Esses são apenas alguns exemplos dos diversos usos dos bancos de dados de documentos em diferentes setores e domínios. Sua flexibilidade e capacidade de lidar com dados semi estruturados os tornam uma escolha popular para uma ampla variedade de aplicativos e cenários de dados.

## **10.4 Vantagens**

### **10.4.1 Flexibilidade de Esquema**

Os bancos de dados de documentos permitem que os desenvolvedores armazenem dados semi estruturados em formato JSON ou similar, o que proporciona flexibilidade para lidar com dados com estruturas variáveis e em evolução ao longo do tempo.

### **10.4.2 Modelagem de Dados Intuitiva**

O modelo de dados baseado em documentos é intuitivo e reflete mais naturalmente as estruturas de dados do mundo real do que o modelo de dados tabular dos bancos de dados relacionais. Isso simplifica o processo de desenvolvimento e manutenção de aplicativos.

### **10.4.3 Escalabilidade Horizontal**

Os bancos de dados de documentos são altamente escaláveis horizontalmente, o que significa que podem lidar com volumes crescentes de dados distribuindo-os entre vários servidores. Isso permite que os aplicativos dimensionem facilmente à medida que crescem em demanda.

### **10.4.4 Desempenho**

Os bancos de dados de documentos geralmente oferecem desempenho rápido e eficiente, especialmente para consultas que envolvem documentos relacionados. Eles são otimizados para acesso rápido aos dados, o que resulta em tempos de resposta mais curtos.

#### **10.4.5 Agilidade no Desenvolvimento**

A flexibilidade de esquema dos bancos de dados de documentos permite que os desenvolvedores reitem rapidamente o desenvolvimento de aplicativos, sem a necessidade de ajustes frequentes no esquema de banco de dados. Isso acelera o ciclo de desenvolvimento e permite uma resposta mais rápida às mudanças nos requisitos do aplicativo.

#### **10.4.6 Suporte a Consultas Complexas**

Muitos bancos de dados de documentos oferecem recursos avançados de consulta que permitem consultas complexas, incluindo agregação, filtragem, projeção e junção de documentos. Isso facilita a análise e a recuperação de dados de maneira eficiente.

#### **10.4.7 Integração com Linguagens de Programação**

Os bancos de dados de documentos geralmente oferecem bibliotecas e drivers para uma ampla variedade de linguagens de programação, facilitando a integração com aplicativos desenvolvidos em diferentes tecnologias.

### **10.5 Desvantagens**

#### **10.5.1 Menor Suporte a Transações Complexas**

Em comparação com bancos de dados relacionais, os bancos de dados de documentos geralmente oferecem suporte limitado a transações complexas que envolvem várias operações atômicas. Isso pode ser uma limitação em casos de uso que exigem transações altamente consistentes e complexas.

### **10.5.2 Desempenho de Consulta em Estruturação Profunda**

Consultas que envolvem estruturas profundamente aninhadas em documentos podem ser menos eficientes em bancos de dados de documentos em comparação com bancos de dados relacionais. Isso ocorre porque os bancos de dados de documentos não são otimizados para operações que requerem acesso a várias coleções ou documentos aninhados.

### **10.5.3 Consumo de Espaço de Armazenamento**

Em certos casos, os bancos de dados de documentos podem consumir mais espaço de armazenamento em comparação com bancos de dados relacionais, especialmente se houver duplicação de dados ou se os documentos contiverem campos excessivamente grandes ou aninhados.

### **10.5.4 Maior Complexidade em Casos de Uso Relacionais**

Em cenários onde há muitos relacionamentos entre entidades e onde é necessário realizar junções complexas de dados, os bancos de dados de documentos podem ser menos adequados. Modelar e consultar relacionamentos complexos pode exigir mais esforço e habilidade do desenvolvedor.

### **10.5.5 Falta de Padronização**

A falta de um padrão universal para bancos de dados de documentos pode levar à fragmentação do ecossistema e à dificuldade de interoperabilidade entre diferentes sistemas. Isso pode complicar a integração de aplicativos que usam diferentes implementações de bancos de dados de documentos.

### **10.5.6 Curva de Aprendizado para Desenvolvedores**

A transição de um modelo de dados relacional para um modelo de documentos pode exigir uma curva de aprendizado para desenvolvedores acostumados com bancos de dados relacionais tradicionais. Isso pode

aumentar o tempo necessário para desenvolver e manter aplicativos que usam bancos de dados de documentos.

Apesar dessas desvantagens, os bancos de dados de documentos continuam sendo uma escolha popular para uma variedade de aplicativos, especialmente aqueles que exigem flexibilidade de esquema, escalabilidade horizontal e desenvolvimento ágil. Como em qualquer tecnologia, é importante avaliar cuidadosamente os requisitos específicos do aplicativo antes de decidir pelo uso de um banco de dados de documentos.

## **10.6 MongoDB**

O MongoDB é um sistema de gerenciamento de banco de dados (SGBD) NoSQL, orientado a documentos e de código aberto. Ele foi desenvolvido para lidar com o armazenamento e a recuperação de dados de forma flexível e escalável, especialmente em ambientes onde a estrutura dos dados pode variar significativamente entre diferentes registros. Em vez de usar tabelas e linhas, como em bancos de dados relacionais tradicionais, o MongoDB armazena dados em documentos JSON (JavaScript Object Notation), o que permite uma modelagem de dados mais dinâmica e adaptável.

### **10.6.1 Arquitetura**

#### **10.6.1.1 Modelo de Dados Orientado a Documentos**

No MongoDB, os dados são armazenados em documentos BSON (Binary JSON), que são semelhantes a objetos JSON. Esses documentos são agrupados em coleções, que são análogas a tabelas em bancos de dados relacionais.

#### **10.6.1.2 Sharding**

O MongoDB suporta sharding, que é a divisão dos dados em partes menores chamadas shards, distribuídas em vários servidores. Isso permite que o MongoDB escale horizontalmente, aumentando a capacidade de armazenamento e processamento conforme necessário.

#### **10.6.1.3 Replicação**

O MongoDB suporta replicação, onde os dados são replicados em vários servidores para garantir alta disponibilidade e tolerância a falhas. Um conjunto de réplicas consiste em vários servidores MongoDB que mantêm cópias idênticas dos dados.

#### **10.6.1.4 Balanceamento de Carga Automático**

O MongoDB possui um balanceador de carga automático que redistribui os dados entre os shards para garantir uma distribuição uniforme da carga de trabalho.

#### **10.6.1.5 Arquitetura Cliente-Servidor**

Os clientes se conectam aos servidores MongoDB para acessar os dados. Os servidores podem ser configurados como nós de dados (responsáveis pelo armazenamento e processamento de dados) ou como nós de consulta (responsáveis por rotear consultas para os nós de dados apropriados).

### **10.6.2 Características**

#### **10.6.2.1 Flexibilidade do Esquema**

O MongoDB não exige um esquema de dados rígido como os bancos de dados relacionais. Isso permite que os desenvolvedores armazenem dados de diferentes estruturas em uma única coleção, facilitando a adaptação a mudanças nos requisitos de dados.

#### **10.6.2.2 Consultas Poderosas**

O MongoDB oferece uma ampla gama de operadores e funcionalidades de consulta, incluindo consultas ad hoc, agregações, índices e suporte a geoespacial, permitindo consultas complexas e eficientes.

#### **10.6.2.3 Indexação**

O MongoDB suporta indexação de campos para melhorar o desempenho de consultas. Os índices podem ser criados em campos individuais, arrays e subdocumentos.

#### **10.6.2.4 Suporte a Transações**

A partir da versão 4.0, o MongoDB oferece suporte a transações multi-documento, permitindo operações atômicas em várias coleções ou documentos.

#### **10.6.2.5 Escalabilidade Horizontal**

O MongoDB é altamente escalável horizontalmente, permitindo adicionar mais servidores para aumentar a capacidade de armazenamento e processamento sem interrupções.

#### **10.6.2.6 Documentação Rica**

O MongoDB possui uma documentação abrangente e uma comunidade ativa que oferece suporte e recursos valiosos para desenvolvedores e administradores.

### **10.6.3 Linguagem Python**

O Python é a linguagem primária para interagir com o MongoDB, e a biblioteca oficial PyMongo permite que os desenvolvedores acessem facilmente todas as funcionalidades do MongoDB em aplicativos Python. Isso inclui operações de CRUD (criar, ler, atualizar, excluir), consultas complexas, indexação, replicação, sharding e muito mais.

A sintaxe limpa e expressiva, tornando-o uma escolha popular para desenvolvedores. Sua simplicidade e facilidade de aprendizado combinam bem com a abordagem flexível de modelagem de dados do MongoDB, permitindo que os desenvolvedores criem e manipulem documentos BSON de forma intuitiva.

O Ecossistema de Ferramentas que o Python possui, suas bibliotecas e frameworks podem ser integrados facilmente com o MongoDB. Isso inclui frameworks

web como Flask e Django, bibliotecas de ciência de dados como Pandas e scikit-learn, e ferramentas de visualização como Matplotlib e Plotly. A integração do MongoDB com Python permite aos desenvolvedores criar aplicativos completos, desde APIs RESTful até aplicativos de análise de dados.

Podemos contar também com suporte da comunidade do Python, o que significa que há uma ampla gama de recursos disponíveis para desenvolvedores que trabalham com o MongoDB em Python. Isso inclui documentação detalhada, tutoriais, exemplos de código, fóruns de discussão e muito mais.

O Python é a linguagem preferida para muitos projetos de aprendizado de máquina e inteligência artificial, e o MongoDB é uma escolha popular para armazenar e analisar grandes volumes de dados usados em aplicativos de IA. A combinação de Python e MongoDB permite que os desenvolvedores construam pipelines de dados completos, desde a coleta e armazenamento de dados até a análise e visualização.

Desta forma, a importância do Python no ecossistema do MongoDB reside na sua facilidade de uso, flexibilidade, vasto ecossistema de ferramentas e suporte da comunidade, tornando-o uma escolha popular para desenvolvedores que buscam criar aplicativos modernos e escaláveis que utilizam o MongoDB como seu banco de dados.

## Referências Bibliográficas

MONGODB - MongoDB PyMongo. New York: MONGODB, 2007. Disponível em: <https://www.mongodb.com/pt-br/docs/drivers/pymongo/>. Acesso em 30 de abr. 2024.

MONGODB - MongoDB Cheat Sheet. New York: MONGODB, 2007. Disponível em: [https://mongodb-devhub-cms.s3.us-west-1.amazonaws.com/Mongo\\_DB\\_Shell\\_Cheat\\_Sheet\\_1a0e3aa962.pdf/](https://mongodb-devhub-cms.s3.us-west-1.amazonaws.com/Mongo_DB_Shell_Cheat_Sheet_1a0e3aa962.pdf/). Acesso em 30 de abr. 2024.

MONGOSH - MongoDB Cheat Sheet. New York: MONGOSH, c 2007. Disponível em: <https://www.mongodb.com/developer/products/mongodb/cheat-sheet/#connect-mongodb-shell/>. Acesso em 30 de abr. 2024.

NASCIMENTO, Filipe. Database Application. Aula 02. 13 abr. 2024. Apresentação do Power Point. Disponível em: 3º Período - TI - Turma A T.I Noite 2024.1. Acesso em 13 abr. 2024.

PANIZ, David. NoSQL. Como armazenar os dados de uma aplicação moderna. s.l, p.1-37, v.1. n.1, s.d. 2016. Disponível em Acesso em: 13 abr. 2024.