

Documentação do 1º trabalho de Banco de Dados

Prefácio

Esta documentação apresenta o sistema de banco de dados e a interface de consultas desenvolvidos para armazenar e acessar informações sobre produtos. Utilizando PostgreSQL, o projeto garante a integridade dos dados e a normalização por meio da Terceira Forma Normal (3FN). A interface de consulta foi desenvolvida para ser utilizada diretamente no terminal.

Desenvolvido pelos Alunos: Jhonnatha Luiz Carvalho e Paulo Ricardo Lima.

Para a matéria de Banco de Dados, ministrada pelo Professor Altigran Soares.

Sumário

1. [Introdução](#)
2. [Esquema do Banco de Dados](#)
 - [Tabela: Produto](#)
 - [Tabela: ProdutoSimilar](#)
 - [Tabela: Categoria](#)
 - [Tabela: CategoriaProduto](#)
 - [Tabela: Review](#)
3. [Diagrama de Relacionamento entre Entidades](#)
4. [Normalização](#)
5. [Inserção de Dados](#)
 - [Estrutura do Script tp1_3.2.py](#)
 - [Funções Principais](#)
 - [Funções Secundárias](#)
 - [Imports](#)
6. [Consultas ao Banco de Dados](#)
 - [Estrutura do Script tp1_3.3.py](#)
 - [Funções de Consulta](#)
7. [Método de Uso](#)
 - [Povoar com tp1_3.2.py](#)
 - [Acessando o Dashboard com tp1_3.3.py](#)
8. [Conclusão](#)
9. [Referências](#)

1. Introdução

Este documento descreve um projeto de um sistema de banco de dados relacional, desenvolvido para armazenar e consultar informações sobre produtos utilizando um banco de dados PostgreSQL. O sistema foi projetado em conformidade com as regras da Terceira Forma Normal (3FN). A implementação inclui scripts para povoar as tabelas e uma interface de consulta para execução no terminal.

2. Esquema do Banco de Dados

2.1 Tabela: **Produto**

- **Descrição:** Contém informações sobre os produtos, como identificador, título, ranking de vendas e grupo ao qual pertencem.
- **Atributos:**
 - **ProductID:** **VARCHAR(50)**, chave primária, NOT NULL, Identificador único do produto.
 - **Title:** **VARCHAR(500)**, NULL, Título do produto.
 - **SalesRank:** **INTEGER**, NULL, Ranking de vendas do produto.
 - **ProductGroup:** **VARCHAR(50)**, NULL, Grupo ao qual o produto pertence.
- **Restrições de Integridade:**
 - Chave Primária: **ProductID**
 - **ProductID** é referenciado em **ProdutoSimilar.ProductID**, **CategoriaProduto.ProductID** e **Review.ProductID**.

2.2 Tabela: **ProdutoSimilar**

- **Descrição:** Relaciona produtos similares, referenciando produtos existentes na tabela **Produto**.
- **Atributos:**
 - **ProductID:** **VARCHAR(50)**, chave estrangeira, NOT NULL, Referencia **Produto.ProductID**.
 - **SimilarProductID:** **VARCHAR(50)**, chave estrangeira, NOT NULL, Referencia **Produto.ProductID**.
- **Restrições de Integridade:**
 - Chave Primária: (**ProductID**, **SimilarProductID**)
 - **ProductID** e **SimilarProductID** referenciam **Produto.ProductID**.

2.3 Tabela: **Categoria**

- **Descrição:** Armazena as categorias de produtos, com identificador único e nome.
- **Atributos:**
 - **CategoryID:** **VARCHAR(50)**, chave primária, NOT NULL, Identificador único da categoria.
 - **CategoryName:** **VARCHAR(200)**, NULL, Nome da categoria.
- **Restrições de Integridade:**
 - Chave Primária: **CategoryID**

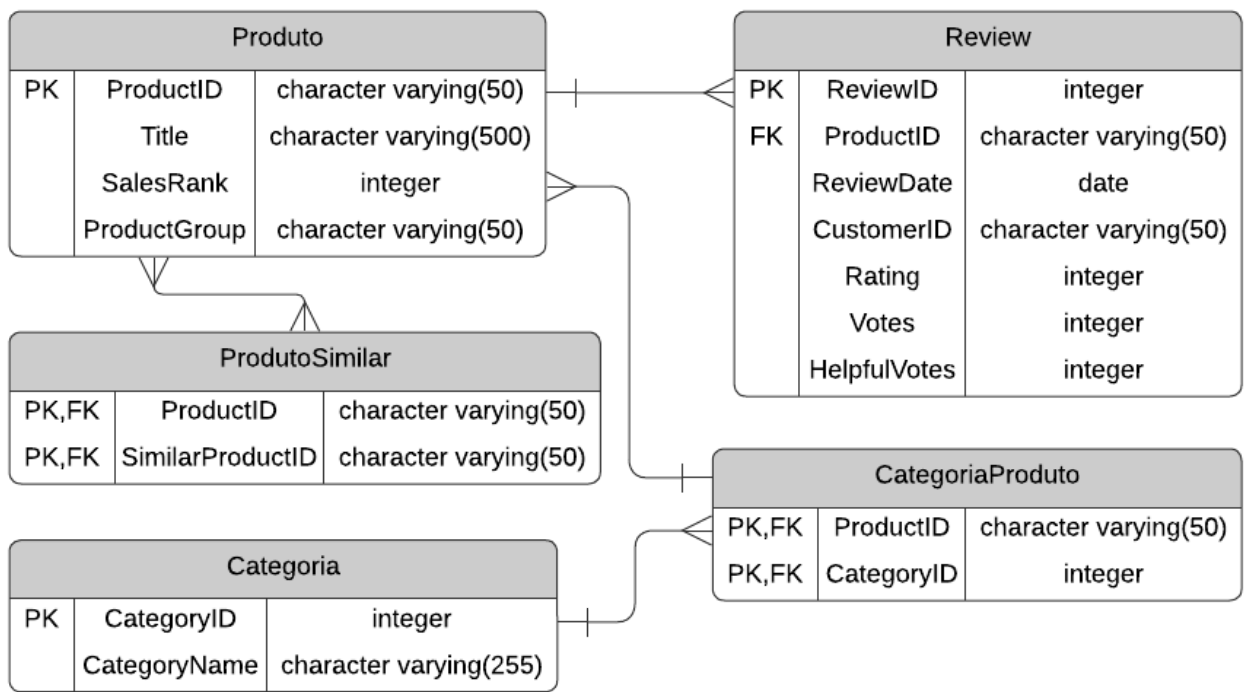
2.4 Tabela: **CategoriaProduto**

- **Descrição:** Relaciona produtos a categorias, associando **ProductID** a **CategoryID**.
- **Atributos:**
 - **ProductID:** **VARCHAR(50)**, chave estrangeira, NOT NULL, Referencia **Produto.ProductID**.
 - **CategoryID:** **VARCHAR(50)**, chave estrangeira, NOT NULL, Referencia **Categoria.CategoryID**.
- **Restrições de Integridade:**
 - Chave Primária: (**ProductID**, **CategoryID**)
 - **ProductID** referencia **Produto.ProductID**
 - **CategoryID** referencia **Categoria.CategoryID**

2.5 Tabela: **Review**

- **Descrição:** Armazena avaliações dos clientes para os produtos.
- **Atributos:**
 - **ReviewID:** **SERIAL**, chave primária, NOT NULL, Identificador único da avaliação.
 - **ProductID:** **VARCHAR(50)**, chave estrangeira, NOT NULL, Referencia **Produto.ProductID**.
 - **ReviewDate:** **DATE**, NULL, Data da avaliação.
 - **CustomerID:** **VARCHAR(50)**, NULL, Identificador do cliente que fez a avaliação.
 - **Rating:** **INTEGER**, NULL, Nota da avaliação.
 - **Votes:** **INTEGER**, NULL, Número de votos recebidos pela avaliação.
 - **HelpfulVotes:** **INTEGER**, NULL, Número de votos úteis recebidos pela avaliação.
- **Restrições de Integridade:**
 - Chave Primária: **ReviewID**
 - **ProductID** referencia **Produto.ProductID**

3. Diagrama de Relacionamento entre Entidades



4. Normalização

O banco de dados foi projetado em conformidade com a Terceira Forma Normal (3FN), assegurando que todos os atributos dependem unicamente da chave primária de suas respectivas tabelas, o que minimiza redundâncias e previne anomalias de atualização.

5. Inserção de Dados

O script `tp1_3.2.py` é responsável pelo povoamento das tabelas no banco de dados. O código utiliza a biblioteca `psycopg2` para estabelecer a conexão com o PostgreSQL e executar comandos SQL de inserção. Os dados são extraídos de um arquivo e inseridos nas tabelas correspondentes. O arquivo é tratado para coletar os dados em lotes, diminuindo o tempo de povoamento.

5.1 Estrutura do Script `tp1_3.2.py`

5.1.1 Funções Principais

- `load_config()`: Carrega as configurações de conexão com o banco de dados.
- `connect(config)`: Estabelece a conexão com o banco de dados usando as configurações fornecidas.
- `create_database(config)`: Cria um banco de dados com base nas configurações fornecidas.
- `create_tables(conn)`: Cria as tabelas necessárias no banco de dados usando a conexão fornecida.
- `inserir_dados_produto(conn, product_data)`: Insere dados na tabela `Produto`.
- `inserir_dados_categoria(conn, category_data)`: Insere dados na tabela `Categoria`.
- `inserir_dados_review(conn, review_data)`: Insere dados na tabela `Review`.
- `inserir_dados_produto_similar(conn, similar_data)`: Insere dados na tabela `ProdutoSimilar`.
- `inserir_dados_categoria_produto(conn, category_product_data)`: Insere dados na tabela `CategoriaProduto`.

5.1.2 Funções Secundárias

- As funções `process_insertion(file_path, config)` e `produto_thread(file_path, config)`: Realizam o processamento de funções em paralelo.
- `dividir_arquivo()`: Divide o arquivo em duas partes para reduzir o tempo de povoamento.
- `deletar_partes()`: Remove os arquivos criados.

5.1.3 Imports

- `import argparse`: Permite criar interfaces de linha de comando, facilitando a análise de argumentos fornecidos ao script.
- `import os`: Utilizada para interagir com o sistema operacional, permitindo manipulação de arquivos e diretórios, além de acessar variáveis de ambiente.
- `import psycopg2`: É uma biblioteca para conectar e interagir com bancos de dados PostgreSQL em Python, permitindo realizar operações como consultas e inserções de dados.
- `import re`: Usada para trabalhar com expressões regulares, permitindo buscar, manipular e validar strings com padrões complexos.
- `import time`: Esta biblioteca fornece funções para trabalhar com tempo, foi utilizada para medir o tempo de execução do código.
- `from concurrent.futures import ThreadPoolExecutor`: Permite executar tarefas em paralelo utilizando threads, facilitando a execução de funções simultaneamente para melhorar o desempenho.

6. Consultas ao Banco de Dados

O script `tp1_3.3.py` é responsável pela execução de consultas ao banco de dados. As consultas disponíveis são listadas abaixo, e o script utiliza a biblioteca `psycopg2` para conectar e consultar o PostgreSQL.

6.1 Estrutura do Script `tp1_3.3.py`

6.1.1 Funções de Consulta

- `listar_comentarios_produto(product_id)`: Lista os 5 comentários com maior e menor avaliação para um produto específico, com base nos votos úteis.
- `listar_similares_maior_venda(product_id)`: Mostra produtos similares ao especificado com melhor classificação de vendas.
- `evolucao_media_avaliacao(product_id)`: Exibe uma tabela da evolução das avaliações médias ao longo do tempo para um produto específico.
- `listar_lideres_venda_por_grupo()`: Lista os 10 produtos líderes de venda em cada grupo de produtos.
- `listar_produtos_melhores_avaliacoes()`: Exibe os 10 produtos com a maior média de avaliações úteis.
- `listar_melhores_categorias()`: Mostra as 5 categorias com a maior média de avaliações úteis.
- `listar_clientes_por_grupo()`: Lista os 10 clientes que mais comentaram por grupo de produto.

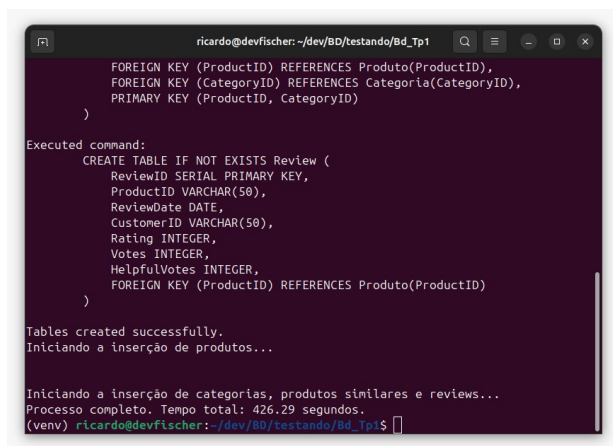
7. Método de uso

7.1 Povoar com `tp1_3.2.py`

Para povoar o banco de dados, execute o seguinte comando:

```
python3 tp1_3.2.py <nome_arquivo.txt>
```

Ao executar esse comando, o sistema começará a inserir os dados coletados no arquivo especificado nas respectivas tabelas do banco de dados.



```
ricardo@devfischer: ~/dev/BD/testando/BD_Tp1
FOREIGN KEY (ProductID) REFERENCES Produto(ProductID),
FOREIGN KEY (CategoryID) REFERENCES Categoria(CategoryID),
PRIMARY KEY (ProductID, CategoryID)
)

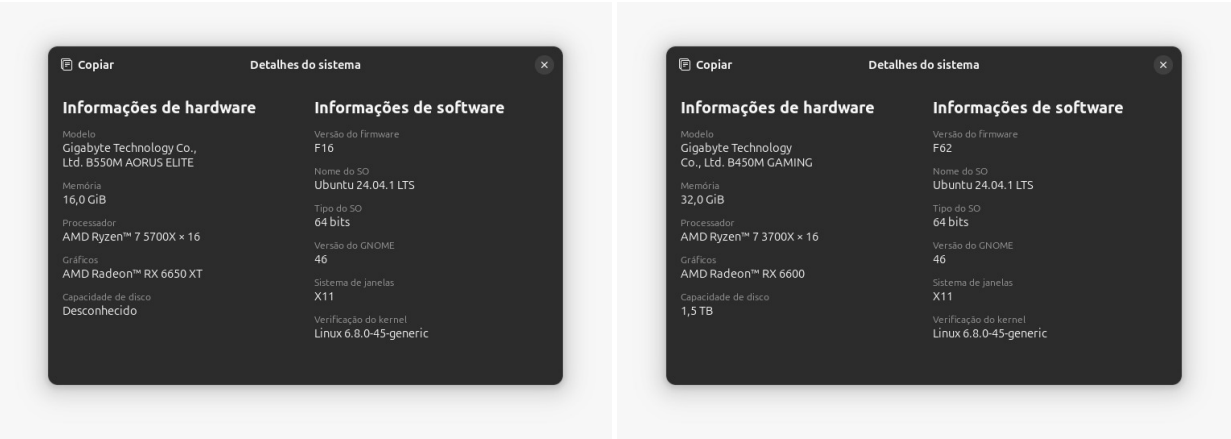
Executed command:
CREATE TABLE IF NOT EXISTS Review (
  ReviewID SERIAL PRIMARY KEY,
  ProductID VARCHAR(50),
  ReviewDate DATE,
  CustomerID VARCHAR(50),
  Rating INTEGER,
  Votes INTEGER,
  HelpfulVotes INTEGER,
  FOREIGN KEY (ProductID) REFERENCES Produto(ProductID)
)

Tables created successfully.
Iniciando a inserção de produtos...

Iniciando a inserção de categorias, produtos similares e reviews...
Processo completo. Tempo total: 426.29 segundos.
(venv) ricardo@devfischer:~/dev/BD/testando/BD_Tp1$
```

O tempo médio para completar o povoamento é de aproximadamente 578 segundos, embora isso possa variar dependendo da quantidade de dados e da configuração do sistema.

Para fornecer uma visão mais clara do desempenho, aqui estão as especificações dos computadores utilizados nos testes:



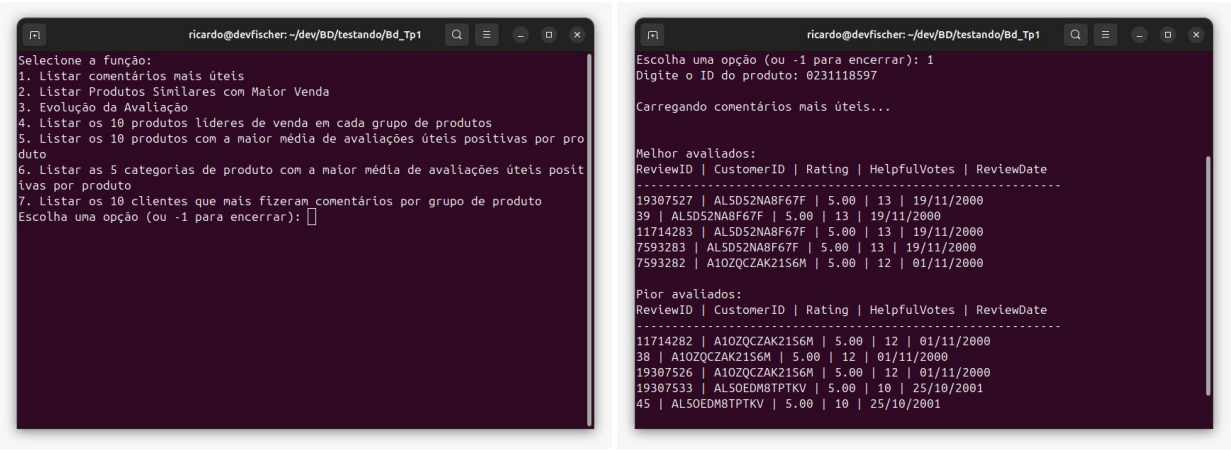
- PC 1 (esquerda): Levou 426 segundos para completar o povoamento.
- PC 2 (direita): O tempo registrado foi de 730 segundos.

7.2 Acessando dashboard com `tp1_3.3.py`

Para acessar o Dashboard depois de concluir o povoamento:

```
python3 tp1_3.3.py
```

Esse comando iniciará o Dashboard, que oferece uma interface para a realização de consultas sobre os dados armazenados.



As respostas às consultas são exibidas diretamente no terminal.

8. Conclusão

O projeto demonstrou a capacidade de estruturar um banco de dados relacional, garantindo a normalização e a integridade dos dados. Com uma estrutura que segue a Terceira Forma Normal (3FN), o projeto visa garantir a integridade dos dados e minimizar redundâncias. A interface de consultas possibilitou um acesso rápido às informações armazenadas.

9. Referências

1. Elmasri, R., & Navathe, S. B. (2011). **Sistemas de Banco de Dados** (6ª ed.). Pearson.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). **Sistemas de Banco de Dados** (6ª ed.). McGraw-Hill.