Guia de Implementação: Sistema de Gerenciamento de Estoque 2.0

Atualização com Grafos e Tabelas Hash

Iago Flávio Hertz Rafael Cauã Wendel

Table of contents

1 Representação dos Produtos e Relacionamentos

- **Classe Produto:** Crie uma classe `Produto` para representar os produtos no estoque. Essa classe pode conter atributos como `id`, `nome`, `categoria`, `quantidade`, etc.
- **Grafo para Produtos:** Use uma estrutura de grafo para modelar as relações entre os produtos. Iremos criar nossa própria classe `graph.py` para isso.

2 Construção e Atualização do Grafo e da Tabela Hash

2.1 Estrutura Geral da Classe `Graph`

- **Atributos Principais:**
  - **Tabela Hash (`hash_table`):** Um dicionário que mapeia os identificadores únicos (`id`) dos produtos para os objetos `Produto`.
  - **Grafo (`graph`):** Um dicionário separado que mapeia os `ids` dos produtos para listas (ou conjuntos) de `ids` de outros produtos com os quais estão conectados (arestas).

2.2 Inicialização da Classe (hertz)

- **Construtor:**
  - Inicialize a tabela hash (`hash_table`) como um dicionário vazio. Este dicionário armazenará todos os produtos.
  - Inicialize o grafo (`graph`) também como um dicionário vazio. Este dicionário armazenará as relações (arestas) entre os produtos.

2.3 Adicionando Nós (Produtos) ao Grafo (hertz)

- **Método `adicionar_no`:**
  - Primeiro, adicione o objeto `Produto` à tabela hash (`hash_table`) usando o `id` do produto como chave.
  - Depois, adicione o `id` do produto como uma chave no dicionário `graph`, com um valor inicial que é uma lista (ou conjunto) vazia, pronta para armazenar conexões futuras.

2.4 Adicionando Arestas (Relações) entre Produtos (iago)

- **Método `adicionar_aresta`:**
  - Receba como entrada dois `ids` de produtos que devem ser conectados.
  - Verifique se ambos os `ids` existem na tabela hash (`hash_table`).
  - Adicione uma conexão no grafo (`graph`):
    - Insira o segundo `id` na lista de conexões do primeiro `id`.
    - Insira o primeiro `id` na lista de conexões do segundo `id`.
  - Se necessário, trate a adição de peso para a aresta, armazenando essa informação junto da conexão no `graph`.

2.5 Removendo Nós e Arestas (iago e cauã)

- **Método `remover_no`:** (iago)
  - Ao remover um nó (produto), primeiro remova o `id` da tabela hash (`hash_table`).
  - Depois, remova todas as arestas que conectam esse `id` a outros nós no grafo (`graph`).
  - Finalmente, remova a entrada correspondente ao `id` no `graph`.
- **Método `remover_aresta`:** (cauã)
  - Para remover uma aresta entre dois produtos:
    - Remova o segundo `id` da lista de conexões do primeiro `id`.
    - Remova o primeiro `id` da lista de conexões do segundo `id`.

2.6 Consultas e Operações no Grafo (os 3 mosqueteiros)

- **Método `obter_no`:** (iago)
  - Dado um `id` de produto, retorne o objeto `Produto` correspondente consultando a tabela hash (`hash_table`).
- **Método `adjacentes`:** (cauã)
  - Dado um `id` de produto, retorne a lista (ou conjunto) de `ids` de produtos conectados a ele no grafo (`graph`).
- **Método `existe_aresta`:** (hertz)
  - Verifique se existe uma aresta entre dois nós (`ids`) consultando suas respectivas listas de adjacência no grafo (`graph`).

2.7 Funções Avançadas (Apenas se der tempo, deixar por último)

- **Métodos de Busca e Análise:**
  - **Busca em Largura (BFS) ou Profundidade (DFS):** Implementar essas buscas para explorar o grafo, usando os `ids` como pontos de partida, e recorrendo à tabela hash (`hash_table`) para acessar os dados dos produtos durante a busca.
  - **Sugestão de Produtos:** Utilize as conexões no grafo para sugerir produtos que frequentemente aparecem juntos, baseando-se nos `ids` conectados ao produto de interesse e acessando os dados via tabela hash.

2.8 Persistência e Manutenção

- **Atualização de Relações:**
  - Sempre que uma nova compra é feita, atualize o grafo incrementando o peso das arestas correspondentes ou criando novas arestas se os produtos nunca foram comprados juntos antes.
- **Salvamento e Carregamento:** (Verificar a necessidade real)
  - Desenvolva métodos para salvar o estado do grafo e da tabela hash em arquivos, e para carregar esses dados de volta na memória quando o sistema for reiniciado.
- **Manutenção de Integridade:**
  - Garanta que a tabela hash (`hash_table`) e o grafo (`graph`) estejam sempre em sincronia. Por exemplo, ao remover um nó, certifique-se de que tanto o nó quanto as conexões associadas sejam removidos corretamente.

3 Resumo Estrutural

1. **Tabela Hash (`hash_table`):** Armazena os produtos com `id` como chave.
2. **Grafo (`graph`):** Armazena as relações (arestas) entre produtos, com `ids` como chaves e listas/conjuntos de `ids` como valores.
3. **Métodos:** Adicionar e remover nós e arestas, consultar conexões, verificar existência de arestas, e realizar buscas no grafo.
4. **Manutenção:** Persistir o grafo e a tabela hash, manter a integridade entre as estruturas.

4 Fluxo de Implementação

- **Inicialização:** No início, você poderia carregar os produtos e relações iniciais (por exemplo, a partir de um banco de dados ou arquivo) para popular o grafo e a tabela hash.
- **Interação com Usuários:** Quando um usuário interagir com o sistema (por exemplo, adicionando um produto ao carrinho), o sistema pode consultar a tabela hash para localizar o produto, usar o grafo para identificar relações relevantes, e sugerir produtos adicionais com base nessas relações.