RELATÓRIO: BINARY SEARCH TREE

Curso: Tópicos em Ciência de Dados A (MAE013)

Professor: Milton R Ramirez

Integrantes

Leonardo Domingos (DRE: 120168324);

Binary Search Tree

1. Introdução

Este projeto teve como objetivo desenvolver um sistema simplificado de gerenciamento de banco de dados que utiliza uma Árvore Binária de Busca (ABB) como estrutura de índice para otimizar as operações de busa. A abordagem escolhida combina uma estrutura de dados linear para armazenamento físico dos registros com uma ABB para indexação, permitindo operações de busca com complexidade O(log n) em vez de O(n), como seria em uma busca sequencial.

O sistema implementado permite:

- Inserção de novos registros com CPF como chave primária
- Busca rápida de registros por CPF
- Remoção lógica de registros
- Listagem ordenada dos registros
- Criação de versões ordenadas do arquivo de dados

A solução foi desenvolvida em Python com princípios de programação orientada a objetos.

2. Metodologia

O projeto foi estruturado nas seguintes quatro classes:

2.1 Classe Registro

Forma a base do sistema, representando a entidade Pessoa com seus atributos essenciais como CPF, nome, data de nascimento e endereço. Esta classe implementa operadores de comparação baseados no CPF, que servem como chave primária, garantindo a ordenação correta na estrutura de índices.

2.2 Classe NoABB

Representa os nós da Árvore Binária de Busca.

Funcionalidades:

- Armazena um objeto Registro e o índice correspondente na EDL
- Mantém referências para os nós filhos esquerdo e direito
- Serve como estrutura básica para construção da ABB

2.3 Classe ABB

Implementa as operações da Árvore Binária de Busca.

Funcionalidades:

- Inserção, remoção e busca de registros
- Percursos (pré-ordem, em ordem, pós-ordem e em largura)

- Construtores para diferentes formas de inicialização
- Cópia profunda da estrutura
- Manutenção da propriedade fundamental da ABB

2.4 Classe EDL (Estrutura de Dados Linear)

Simula o arquivo físico de registros.
Funcionalidades:

- Armazenamento sequencial dos registros
- Operações básicas de inserção, busca por índice e remoção lógica
- Criação de versão ordenada a partir da ABB

2.5 Classe SGBD (Sistema Gerenciador de Banco de Dados)

Integra a ABB (índice) e a EDL (dados).
Funcionalidades:

- Inserção de novos registros (atualizando tanto EDL quanto ABB)
- Busca eficiente por CPF
- Remoção de registros
- Geração de relatórios ordenados

3. Comentários sobre o Desenvolvimento

Um dos primeiros obstáculos foi entender bem como os métodos de comparação *(__lt__, __eq__)* funcionam na prática. Na teoria parece simples, mas na prática, quando tentei usar os objetos na árvore, sem esses métodos definidos, nada funcionava direito. Foi necessário entender bem como implementar isso com base no CPF, e só depois disso a árvore finalmente começou a se comportar como o esperado.

Outro desafio foi remover nós da árvore, especialmente aqueles com dois filhos, o que exige mais cuidado. De forma geral, a lógica entre conectar *nodes* de estruturas de dados é algo que sempre dá bastante dificuldade de acertar sem só olhar a resposta.

Além disso, manter tudo sincronizado entre a estrutura de índice (ABB) e o armazenamento real dos dados (EDL) deu trabalho. Depois de sofrer um pouco, decidi concentrar essa lógica na classe SGBD, que virou o centro de controle do sistema.

Também tive dificuldade em implementar os percursos da árvore. Os em profundidade (pré, em e pós-ordem) foram mais tranquilos, mas o percurso em largura exigiu uma abordagem diferente, com fila, e demorou mais para funcionar.

Por fim, teve toda a parte de testes e validação, e assim como em outros trabalhos criei uma função de testes para servir de benchmark para o projeto.