Relatório de ABB e Índices de SBD

integrantes : Felipe Rodrigues Ribeiro DRE:119052031

Igor Torres Torres da Costa DRE:119034669

Este estudo aborda a criação de uma solução com base na programação orientada a objetos para implementar uma Árvore Binária de Busca (ABB), aplicada à construção de um sistema de indexação de acesso inspirado em Sistemas Gerenciadores de Banco de Dados (SBD). O objetivo principal foi idealizar uma estrutura eficiente para manter e localizar registros de dados utilizando uma chave única, como o CPF, mimetizando o funcionamento de um índice em bancos de dados.

A implementação foi estruturada em três classes essenciais: a classe `Registro`, que representa cada pessoa ou objeto armazenado, contendo campos como CPF, nome e data de nascimento; a classe `NoArvore`, encarregada de abranger cada nó da árvore, mantendo o objeto `Registro` e o índice de sua posição em uma estrutura de dados linear; e por fim a classe `ArvoreBinariaBusca`, que engloba os métodos primários para inserção, busca, remoção e trajetórias na árvore.

A classe `Registro` foi moldada para consentir comparação direta entre objetos, utilizando o operador especial `__lt__`, o que simplifica a organização e as comparações durante a inserção e a busca na ABB. Essa adaptabilidade igualmente viabiliza que a chave de ordenação eleita, neste caso o CPF, possa ser facilmente trocada por outros campos, se preciso.

Um aspecto relevante foi a distinção entre os dados reais, mantidos em uma estrutura linear (neste caso, uma lista Python que simula um arquivo ou "arquivo de registros"), e o índice de acesso, executado pela ABB. Cada vez que um novo registro é inserido, ele é adicionado ao final da lista (EDL), o que assegura uma complexidade constante O(1) para essa operação. Simultaneamente, o CPF é inserido na ABB como chave de ordenação, e no nó relativo também se retém a posição (índice) do registro na lista.

Durante o desenvolvimento, uma das maiores dificuldades foi implementar corretamente a remoção de nós na ABB, garantindo que a árvore se mantenha balanceada e consistente mesmo após exclusões. Foi crucial lidar com os diversos casos de remoção: nós sem filhos, nós com um único filho e nós com dois filhos, requerendo a substituição do nó pelo menor valor da subárvore à direita. Outro desafio foi sincronizar a exclusão na árvore com a marcação do registro como "deletado" na lista, sem deslocar ou alterar os índices dos demais elementos, para não comprometer a correspondência entre a árvore e a lista.

A implementação também abrangeu rotinas para explorar a árvore em três modos: pré-ordem, em ordem e pós-ordem. Essas abordagens são cruciais para diversas ações, como gerar uma nova relação de cadastros classificados por CPF, replicando a criação de um arquivo organizado para fins de pesquisa ou geração de relatórios. Adicionalmente, foi viável aplicar um indicador de exclusão em cada cadastro, mantendo o dado fisicamente no arquivo, mas desconsiderando-o em pesquisas e iterações, espelhando a ideia de deleção lógica comum em bancos de dados atuais.

Em Sistemas Gerenciadores de Banco de Dados, a ABB atua como um índice complementar, otimizando a velocidade das buscas ao reduzir o tempo de consulta de O(n) (busca sequencial) para O(log n) em média. Essa tática aumenta consideravelmente a eficiência em operações de leitura e modificação, simulando a operação de um índice em um SGBD relacional.

Ao combinar a árvore com a lista linear, foi possível desenvolver um sistema de gestão de cadastros que não apenas viabiliza a inserção e busca eficientes, mas também produz uma visão organizada dos dados quando necessário. Nos testes, foi possível adicionar, encontrar e apagar cadastros de forma simples, e ao final gerar uma lista totalmente classificada por CPF. Essa organização permite visualizar, por exemplo, o sistema como base para um aplicativo de cadastro de clientes ou pacientes em um hospital, onde cada CPF serve como identificação principal.

Concluindo, a solução elaborada se mostrou sólida, modular e eficaz, replicando conceitos essenciais empregados em índices de bancos de dados. Apesar dos desafios encontrados, sobretudo na remoção de nós e na sincronização entre a árvore e a lista linear, a experiência foi valiosa e ilustrou na prática como estruturas de dados clássicas, como a Árvore Binária de Busca, permanecem fundamentais em aplicações reais que exigem alto desempenho e organização de grandes volumes de dados. Essa implementação pavimenta o caminho para melhorias futuras, como balanceamento automático (AVL ou Red-Black Tree), criação de múltiplos índices (por exemplo, por nome ou data de nascimento) e suporte a chaves compostas, tornando o sistema ainda mais completo e semelhante a um SGBD real.