APLICAÇÃO DA ÁRVORE B (B-TREE)

DAVI VENTURA CARDOSO PERDIGÃO
EDMILSON LINO CORDEIRO
ERIC HENRIQUE DE CASTRO CHAVES

> *Método genérico para o armazenamento e a recuperação de dados, voltado para arquivos volumosos, pois proporciona rápido acesso aos dados e possui custo mínimo de overhead.*

Boing Corporation, em **1972**.

CARACTERÍSTICAS ÁRVORE-B

- Possui uma sequência ordenada de chaves (conjunto de ponteiros). **Número de ponteiros = número de chaves + 1** (exemplo: árvore B de ordem 8,máximo de 7 chaves e 8 ponteiros). A raiz contém no mínimo **2 chaves.**
- Cada página, exceto a raiz e as folhas, têm no mínimo **m/2** descendentes.
- Desempenho proporcional a **log k^I** (I = tamanho do índice; K = tamanho da página de disco) ou melhor. O número de acessos ao disco exigidos para a maioria das operações em uma árvore B é proporcional a sua altura.
- Todas as folhas estão no mesmo nível, isso faz com que ela seja uma árvore **completamente balanceada**.
- **Problema:** a literatura não é uniforme no uso e definição dos termos.

O PROBLEMA

CONTROLE DE PONTO NO MODELO HOME OFFICE

Sabemos que, hoje em dia, o home office é uma grande realidade no mundo todo. Uma grande **vantagem** desse modelo tanto para colaboradores quanto para empresas é a possibilidade de fazer escalas mais flexíveis de trabalho, sendo possível organizar melhor o tempo do funcionário e as suas atividades. Porém, nesse mesmo benefício se encontra um grande **desafio** para as empresas empregadoras: **controlar as horas trabalhadas por cada funcionário**.

Utilizando como exemplo uma grande empresa, segundo a RAIS (Relação Anual de Informações Sociais) do Ministério do Trabalho e Previdência

Funcionários

70.000

Funcionários em Home Office

24.500 (35%)

98.000

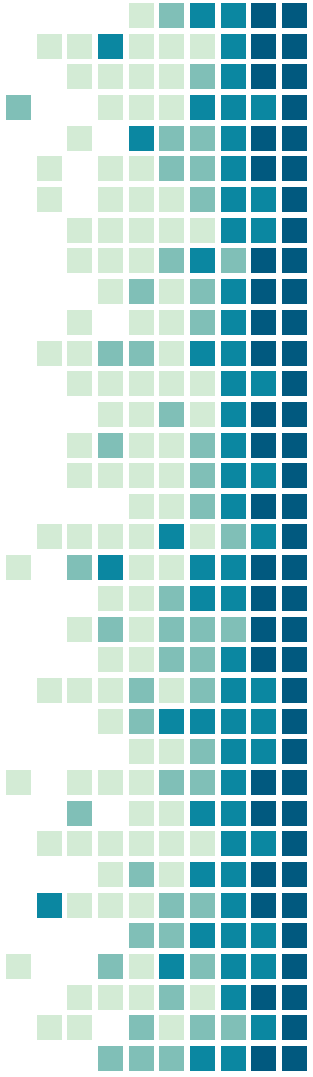
Registros Diários

490.000

Registros Semanais

1.960.000

Registros Mensais



O ALGORÍTMO E SUA COMPLEXIDADE

Em um banco de dados com gigabytes de informações usando árvore B, uma pesquisa será resolvida da seguinte forma:

Serão percorridos os nós dentro do índice e cada nó alcançado separará fisicamente dados que não estejam dentro de uma seletividade, até atingir uma profundidade máxima, será nessa profundidade máxima que estabelecerá os limites para a pesquisa. A palavra seletividade é apenas o termo que usamos quando algum nó possui algum nexo, isto é, semelhança ou aproximação com o que estamos procurando.

A seguir segue um exemplo de funções básicas do algoritmo de árvore B, ele apresenta complexidade O(t logt n) em todas as suas operações básicas.

```c
void inserir(int chave) {
    struct node *novono;
    int novachave;
    enum chaveStatus valor;
    valor = ins(raiz, chave, &novachave, &novoNo);
    if (valor == Duplicado)
        printf("chave ja disponivel\n");
    if (valor == InserirIt) {
        struct node *novaRaiz = raiz;
        raiz = (struct node *) malloc(sizeof(struct node));
        raiz->n = 1;
        raiz->chaves[0] = novaChave;
        raiz->p[0] = novaRaiz;
        raiz->p[1] = novoNo;
    }
}
void deletarNo(int chave) {
    struct node *novaRaiz;
    enum KeyStatus valor;
    valor = del(raiz, chave);
    switch (valor) {
        case PesquisaDeErro:
            printf("chave %d não está disponivel\n", chave);
            break;
        case MenoresChaves:
            novaRaiz = raiz;
            raiz = raiz->p[0];
            free(novaRaiz);
            break;
    }
}


void Destruir() {
    while (raiz != NULL) {
        DelNode(raiz->chaves[0]);
    }
}
```

```c
void buscar(int chave) {
    int pos, i, n;
    struct no *ptr = raiz;
    printf("Buscar caminho:\n");
    while (ptr) {
        n = ptr->n;
        for (i = 0; i < ptr->n; i++)
            printf(" %d", ptr->chaves[i]);
        printf("\n");
        pos = searchPos(chave, ptr->chave, n);
        if (pos < n && chave == ptr->chaves[pos]) {
            printf("Chave %d encontrada na posição %d do nó\n", chave,
i);
            return;
        }
        ptr = ptr->p[pos];
    }
    printf("Chave %d não foi encontrada\n", chave);
}
```

Exemplo da função ***inserir(int chave)*** que se refere à matrícula do funcionário que desejamos buscar as informações do registro de ponto: **51, 3477, 1320, 1755, 5600, 8375, 13899, 666, 4742, 11170**.

- **51**

- **3477**

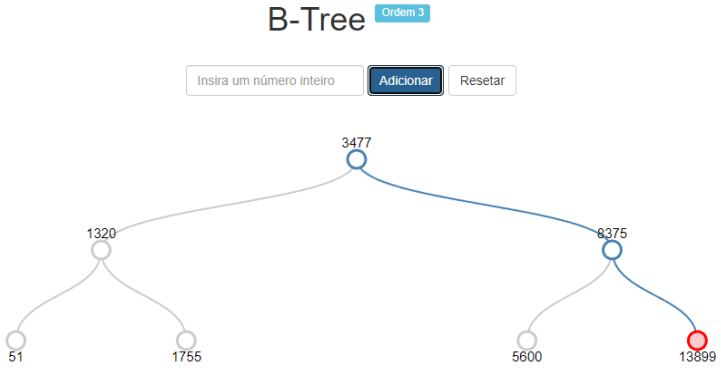
1320

1755

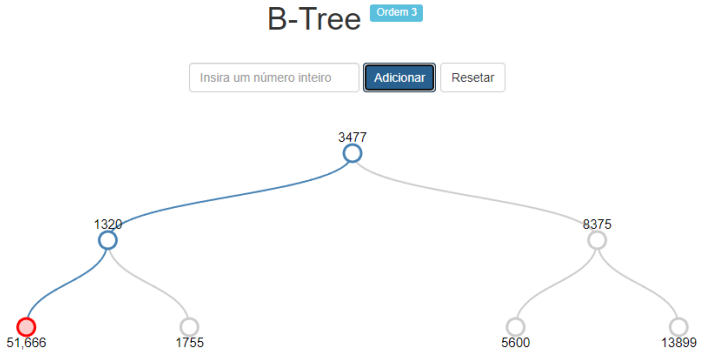
5600

8375

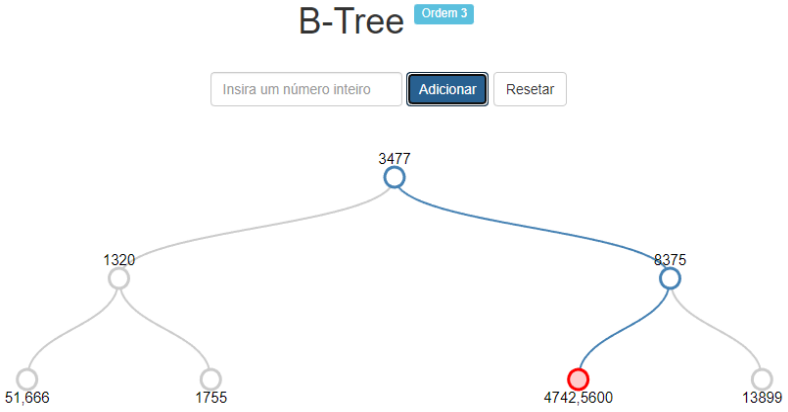
13899



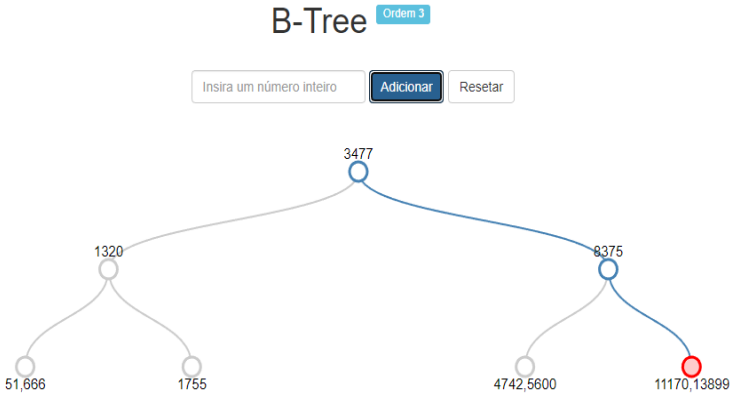
666



4742



11170



CONCLUSÃO

As árvores B são um tipo de árvores de busca que foram projetadas para minimizar o número de acessos à memória secundária. Como o número de acessos à memória secundária depende diretamente da altura da árvore, as árvores B foram projetadas para ter uma altura inferior para um dado número de chaves.

Claro, existem algumas desvantagens em usar índices árvores B, como por exemplo se o banco de dados for pequeno ele obterá menor performance em comparação com índices comuns, então recomenda-se sua aplicação apenas em bancos de dados com grandes quantidades de tabelas, campos e registros.

OBRIGADO!

Alguma Pergunta?