

# Classificação e Pesquisa de Dados

Aula 20

Árvores B e B+

B-Trees (Árvores B) [Bayer & McCreight 1970]

UFRGS

INF01124

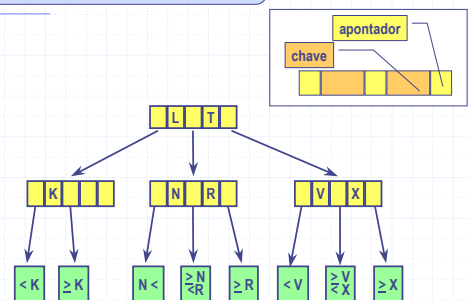
## B-Trees (Árvores B)

- ◆ Árvores de pesquisa balanceadas, projetadas para minimizar o tempo de acesso para buscas, inserções e remoções de registros em grandes arquivos de dados armazenados em disco
- ◆ Cada nodo de uma árvore B pode conter vários nodos filhos
- ◆ Generalização de árvores binárias de pesquisa
- ◆ Outras árvores, como árvores binária de pesquisa e AVL, são muito eficientes quando os dados se encontram armazenados na memória principal
- ◆ Em uma aplicação típica de árvores B, a quantidade de dados manipulados não cabe na memória principal

## Árvores B (Cont.)

- ◆ Se um nodo  $x$  de uma árvore B contém  $k$  valores de chaves, então  $x$  tem  $k+1$  nodos filhos, e tais valores de chave funcionam como separadores para  $k+1$  intervalos
- ◆ Uma árvore B com  $n$  nodos tem altura  $O(\log n)$ , apesar de sua altura ser consideravelmente menor que a de uma árvore binária
- ◆ Algoritmos precisam gerenciar a leitura (gravação) de partes do arquivo de (para) o disco, quando necessário
- ◆ Algoritmos de manipulação de árvores B precisam apenas de um número fixo de páginas (do arquivo) na memória principal em qualquer instante de tempo
- ◆ O tamanho da memória principal não limita o tamanho da árvore B que pode ser manipulada

## Exemplo de Árvore B



## Árvores B Propriedades

- ◆ Cada nodo  $x$  contém os seguintes campos:
  - Número de chaves atualmente armazenadas ( $n[x]$ )
  - As  $n[x]$  chaves armazenadas em ordem não decrescente
  - Uma variável booleana que indica se um nodo é folha ( $leaf[x]$ )
- ◆ Se  $x$  é um nodo interno (i.e., não é folha), ele conterá  $n[x]+1$  ponteiros para os seus nodos filhos
- ◆ Os valores de chave de nodo  $x$  separam os filhos de  $x$  em intervalos por valores de chaves
- ◆ Todas as folhas se encontram no mesmo nível, que corresponde à altura da árvore

## Árvores B Propriedades (Cont.)

- ◆ Existe um limite inferior e um limite superior para o número de chaves que cada nodo pode conter
  - Estes limites são definidos em termos de um valor inteiro  $t$ , chamado grau mínimo da árvore B
- ◆ Todo nodo interno, exceto a raiz, deve conter pelos menos  $t-1$  chaves. Se a árvore não for vazia, a raiz deve conter pelo menos um valor de chave
- ◆ Todo nodo pode conter no máximo  $2t-1$  chaves
- ◆ A árvore B mais simples tem  $t = 2$  (conhecida como árvore 2-3-4)

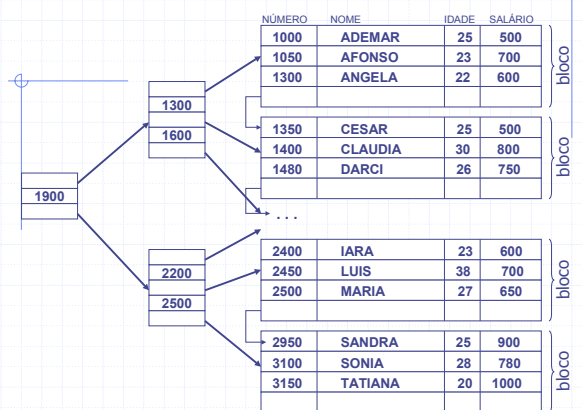
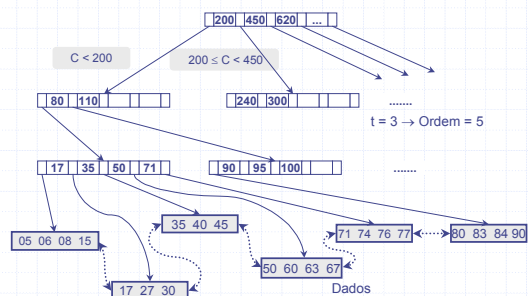
## Árvores B Número de Nodos

- Na prática, valores bem maiores de  $t$  são utilizados. Em geral, o tamanho de um nó em uma árvore B corresponde ao tamanho de uma página (bloco) em disco.
- Para arquivos grandes, o número de filhos costuma ficar entre 50 e 2000
- O número de filhos de um nó é limitado pelo tamanho das chaves e pelo tamanho da página de disco
- Uso de "fatores de derivação" elevados reduz a altura da árvore significativamente, e, como consequência, o número de acessos a disco

## Árvores B Variações

- Armazenar dados "satélite" nos nós junto com suas respectivas chaves
  - Ocupa mais espaço, reduzindo o número de filhos de cada nó, com consequente aumento na altura da árvore
- Armazena em cada nó apenas os valores de chave e ponteiros para a página de disco contendo os dados satélites
- Melo termo: armazena valores de chave e ponteiros para a página de disco no caso de nós internos. No caso de nós folhas, armazena os dados satélites diretamente
- Árvore B+:** os nós folhas são conectados utilizando uma lista duplamente encadeada
  - Valores de chaves presentes em nós internos são replicados nas folhas
  - Permite acessar o arquivo sequencialmente (ordenado pelo valor de chave) ou aleatoriamente (para um dado valor de chave)
  - Utilizado na prática para construção de arquivos de índice

## Árvore B + Exemplo

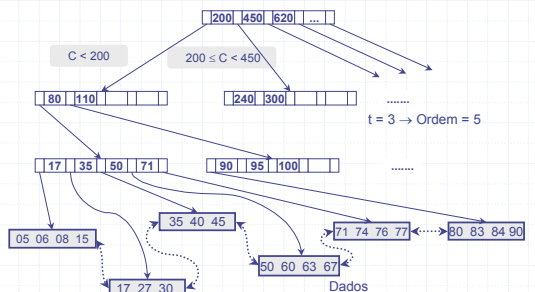


## Pesquisa em Árvores B

- Procura de um valor  $k$  em uma árvore B
- Inicie lendo a raiz da árvore a partir do disco
- Procure  $k$  dentro do nó lido (pode ser usada pesquisa binária, pois as chaves estão ordenadas dentro do nó). Se encontrou, encerra a busca; caso contrário, continue a busca, lendo o filho correspondente, a partir do disco
- Continue a busca até que  $k$  tenha sido encontrado ou você tenha feito a busca em uma folha da árvore
- O tempo de busca é proporcional a  $O(\log_t n)$ . Mais especificamente:
  - $O(t \log_t n)$  (no caso de busca sequencial no nó) ou
  - $O(\log_2 t \log_t n)$  (no caso de busca binária no nó)

## Exercício

- Pesquisar as chaves 84 e 76 na árvore B abaixo:



## Pesquisa em Árvores B

```

Proc pesquisa_árvore_B ( x, k );
    { x : nodo da árvore B }
    { k : valor de chave procurado }

begin
    i ← 1;
    while ( i ≤ n[x] and k < key[x] ) /* pesq. sequencial no nodo */
    do i ← i + 1;
    if ( i ≤ n[x] and k = key[x] ) then
        return ( x, i ); /* retorna nodo e ordem da chave */
    if ( leaf[x] ) then
        return nil; /* não encontrou */
    else
        begin
            DISK-READ(c[x]); /* lê nodo filho do disco e prossegue */
            return pesquisa_árvore_B ( c[x], k );
        end;
    end;

```

Como  $n[x] < 2t$  e a árvore tem altura  $O(\log n)$ , o custo do procedimento é  $O(t \log n)$ .

## Criação de uma Árvore B vazia

```

Proc cria_árvore_B_vazia ( T );
    { T : árvore }

begin
    x ← aloca_nodo();
    leaf[x] ← TRUE;
    n[x] ← 0;
    DISK-WRITE(x);
    root[T] ← x;
end;

```

O custo do procedimento é  $O(1)$ .

## Inserção em Árvores B

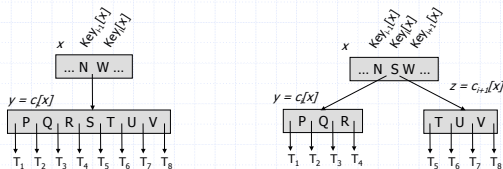
- ◆ Procedimento mais elaborado do que em árvores binárias de pesquisa
- ◆ Use o algoritmo de busca para determinar em que nó (folha) deve ser feita a inserção
- ◆ Garante que nenhum nodo conterá mais que  $2t-1$  chaves
- ◆ Se a inserção da nova chave resultar em um número de chaves naquele nó menor ou igual a  $2t-1$ , então simplesmente grave o nó no disco e encerre
- ◆ Caso o nodo  $y$  que deverá receber o novo valor de chave já esteja cheio ( $2t-1$  chaves), este deverá ser subdividido (em torno do valor mediano) em dois nodos antes que a chave possa ser inserida

## Particionamento de Nodos

- ◆ Aloque um novo nodo  $z$
- ◆ O novo nodo  $z$  adota os  $t-1$  maiores valores de chave do nodo cheio ( $y$ ), o qual mantém apenas os  $t-1$  menores valores
- ◆ O valor mediano é transferido para o nodo pai ( $x$ ) do nodo subdividido ( $y$ )
- ◆ Caso o nodo pai já se encontre cheio, o particionamento deve ser propagado para os nodos ancestrais, eventualmente chegando-se a raiz
- ◆ Caso a raiz já se encontre cheia e tenha que receber mais um valor, um novo nodo é criado para armazenar o valor mediano da raiz. Este nodo se torna a nova raiz da árvore
- ◆ A subdivisão do nodo raiz é a única forma de aumentar a altura de uma árvore B

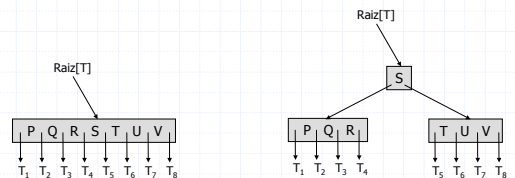
## Exemplo de Particionamento

- ◆ Particionamento de um nodo não raiz



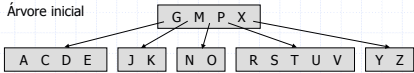
## Exemplo de Particionamento

- ◆ Particionamento do nodo raiz

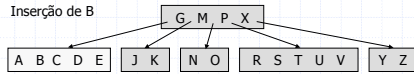


## Exemplos de Inserção

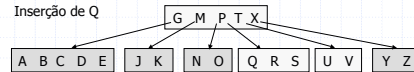
Árvore inicial



Inserção de B

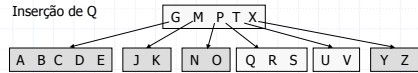


Inserção de Q

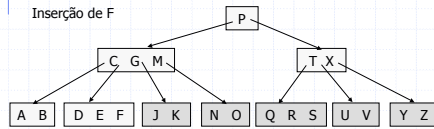


## Exemplos de Inserção

Inserção de Q



Inserção de F

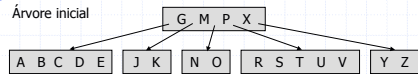


## Remoção em Árvores B

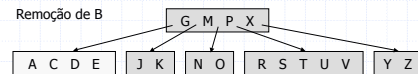
- ◆ Garante que nenhum nó, exceto a raiz, conterá menos que  $t-1$  nodos
- ◆ Se a remoção de uma chave resultar em um número de chaves maior ou igual a  $t-1$ , então remove o nó, grava o resultado no disco e encerre
- ◆ Caso o nó interno  $x$  (não raiz) que deverá ter a chave  $k$  removida
  - (1) Contenha apenas  $t-1$  chaves e o filho de  $x$  que precede/sucedo  $k$  tenha pelo menos  $t$  chaves, substitui  $k$  pelo seu predecessor/sucessor
  - (2) Contenha pelo menos  $t$  chaves e os nodos  $y$  e  $z$ , contendo o predecessor e o sucessor de  $k$ , respectivamente, contêm apenas  $t-1$  chaves, remove-se  $k$  do nó  $x$  e funde-se  $y$  e  $z$  em um único nó
- ◆ Caso  $x$  seja um nó folha
  - (3) Se um dos irmãos vizinhos (esquerda ou direita) de  $x$  contiver pelo menos  $t$  chaves, transfere-se uma chave do pai de  $x$  para  $x$ , e transfere-se o predecessor (se irmão da esquerda) ou sucessor (se irmão da direita) da chave transferida para seu lugar no nó  $x$
  - Caso ambos os irmãos vizinhos tenham apenas  $t-1$  chaves, procede-se uma fusão

## Exemplos de Remoção

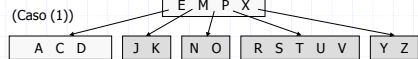
Árvore inicial



Remoção de B



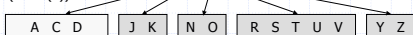
Remoção de G



## Exemplos de Remoção

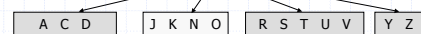
Remoção de G

(Caso (1))



Remoção de M

(Caso (2))



Remoção de Y

(Caso (3))

