Árvore B

Profa. Dra. Cristina Dutra de Aguiar Ciferri

Árvore-B

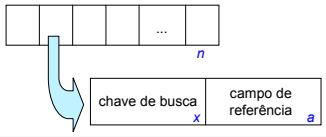
- Método genérico para o armazenamento e a recuperação de dados
  - voltado para arquivos volumosos
  - proporciona rápido acesso aos dados
  - possui custo mínimo de *overhead*
- Referência
  - Bayer, R.; McCreight, E. *Organization and Maintenance of Large Ordered Indexes*.
  - Boing Corporation, 1972.

Sistemas de Banco de Dados

- 1979
  - árvore-B é, de fato, a organização padrão para indexação
- Atualmente
  - amplamente utilizada
  - existem variantes
    - árvore-B$^+$
    - árvore-B*
    - árvore-B paralela

Características Gerais

- Organizar e manter um índice para um arquivo de acesso aleatório altamente dinâmico
- Índice
  - *n* elementos (x,a) de tamanho fixo



Características Gerais

- Índice
  - extremamente volumoso
- *Buffer-pool* pequeno
  - apenas uma parcela do índice pode ser carregada em memória principal
  - operações baseadas em disco
- Desempenho
  - proporcional a $\log_K{}^I$ *ou melhor*

- I: tamanho do índice
- K: tamanho da página de disco

Revisão

- Problemas específicos
  - é necessário encontrar uma forma de se buscar uma chave realizando-se poucos acessos a disco
  - é necessário encontrar uma forma de se realizar inserções e remoções tal que se tenham apenas efeitos “locais” no índice ao invés de se requerer uma reorganização “quase completa” do índice

Revisão

- Estruturas estudadas
  - árvore binária de busca
  - árvore AVL
  - árvore binária paginada
- Enfoque de construção de árvores paginadas
  - construção *top-down* PROBLEMA

Construção *Top-Down*

- Situação 1
  - conhecimento de todo o conjunto de chaves, antes da construção da árvore
- Etapas
  - ordenação das chaves
  - construção da árvore a partir da raiz com base na ordenação
    - chave intermediária: raiz da árvore

Inicia-se pela chave do meio para se obter uma árvore balanceada

Construção *Top-Down*

- Situação 2
  - chaves são recebidas aleatoriamente e inseridas na árvore imediatamente
- Características
  - construção da árvore a partir da raiz
  - cada vez que uma chave é inserida, a árvore dentro da página sofre uma rotação, sempre que necessário, para manter o balanceamento

Exemplo: C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

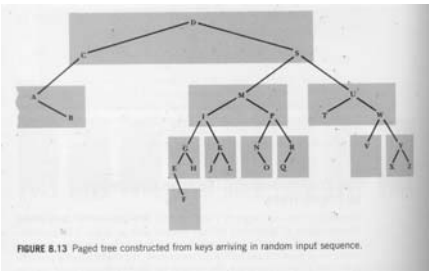


FIGURE 8.13 Paged tree constructed from keys arriving in random input sequence.

Dificuldades (Exemplo)

- Chaves iniciais devem ir para a raiz
  - C ou D
    - chaves adjacentes, inseridas em seqüência
    - chaves inapropriadas, pois causam o desbalanceamento da árvore
- Chave inapropriada no nó raiz ou em qualquer raiz de uma subárvore
  - balanceamento difícil da árvore, porque pode gerar implicações em outras páginas

Construção *Top-Down*

- Questões
  - como garantir que as chaves na página raiz são boas separadoras, i.e., dividem o conjunto de chaves de maneira balanceada?
  - como evitar o agrupamento de chaves extremas, que não deveriam estar na mesma página (como C, D e S, por exemplo)
  - como garantir que cada página contenha um número mínimo de chaves?

Árvore B

- Características
  - balanceada
  - *bottom-up* para a criação (em disco)
    - nós folhas → nó raiz
- Inovação
  - não existe a necessidade de se construir a árvore a partir do nó raiz, como é feito pelas árvores em memória principal e pelas árvores anteriormente projetadas para disco

Construção *Bottom-Up*

- Conseqüências
  - chaves "erradas" não são mais alocadas no nó raiz
    - elimina as questões em aberto de *chaves separadoras* e de *chaves extremas*
  - não é necessário tratar o problema de desbalanceamento usando algoritmos de reorganização da árvore

na árvore-B, as chaves na raiz da árvore emergem naturalmente

Características

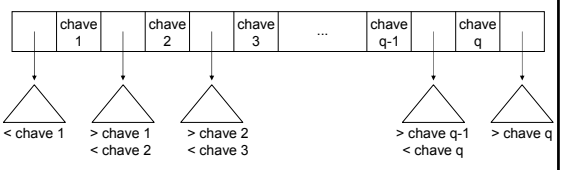
- Nó (= página de disco)
  - seqüência ordenada de chaves
  - conjunto de ponteiros
    - número de ponteiros = número de chaves + 1

➢ não há uma árvore explícita dentro de uma página (ou nó da árvore)

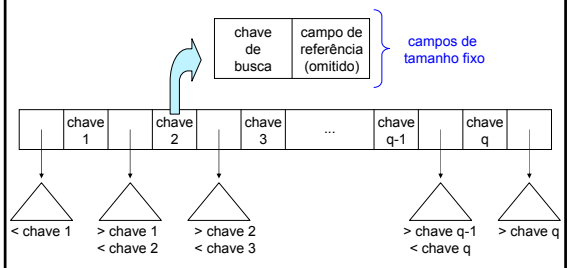
Características

- Ordem
  - número máximo de ponteiros que pode ser armazenado em um nó
  - exemplo: árvore-B de ordem 8
    - máximo de 7 chaves e 8 ponteiros
- Observações
  - número máximo de ponteiros é igual ao número máximo de descendentes de um nó
  - nós folhas não possuem filhos, e seus ponteiros são nulos

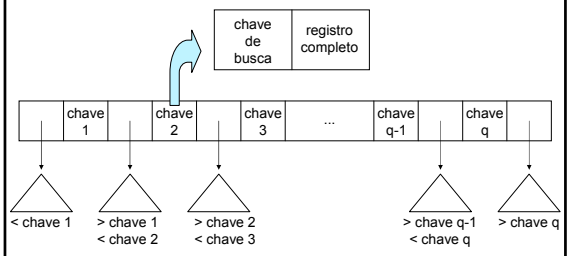
Estrutura Lógica de um Nó



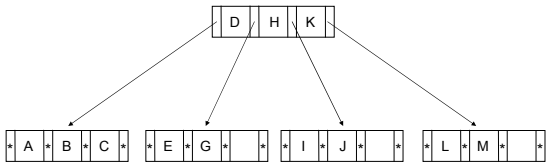
Estrutura Lógica de um Nó



Estrutura Lógica de um Nó



Exemplo



Inserção de Dados (Chave)

- Característica
  - sempre realizada nos nós folhas
- Situações a serem analisadas
  - árvore vazia
  - *overflow* no nó raiz
  - inserção nos nós folhas

Inserção: Situação Inicial

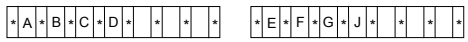
- Criação e preenchimento do nó
  - primeira chave: criação do nó raiz
  - demais chaves: inserção até a capacidade limite do nó
- Exemplo
  - nó com capacidade para 7 chaves
  - chaves: letras do alfabeto
  - situação inicial: árvore vazia

Inserção: Situação Inicial

- Chaves B C G E F D A
  - inseridas desordenadamente
  - mantidas ordenadas no nó
- Ponteiros (*)
  - nós folhas: -1 ou fim de lista (NIL)
  - nós internos: RRN do nó filho ou -1
- Nó raiz (= nó folha)

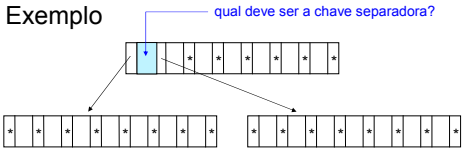
Inserção: *Overflow* Nó Raiz

- Passo 1 – particionamento do nó (*split*)
  - nó original → nó original + novo nó
    - *split* 1-to-2
  - as chaves são distribuídas uniformemente nos dois nós
    - chaves do nó original + nova chave
- Exemplo: inserção de J



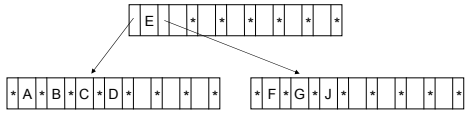
Inserção: *Overflow* Nó Raiz

- Passo 2 – criação de uma nova raiz
  - a existência de um nível mais alto na árvore permite a escolha das folhas durante a pesquisa
- Exemplo



Inserção: *Overflow* Nó Raiz

- Passo 3 – promoção de chave (*promotion*)
  - a primeira chave do novo nó resultante do particionamento é promovida para o nó raiz
- Exemplo



Inserção: Nós Folhas

- Passo 1 – pesquisa
  - a árvore é percorrida até encontrar o nó folha no qual a nova chave será inserida
- Passo 2 – inserção em nó com espaço
  - ordenação da chave após a inserção
  - alteração dos valores dos campos de referência

nó folha em memória principal

Inserção: Nós Folhas

- Passo 2 – inserção em nó cheio
  - particionamento
    - criação de um novo nó (nó original → nó original + novo nó)
    - distribuição uniforme das chaves nos dois nós
  - promoção
    - escolha da primeira chave do novo nó como chave separadora no nó pai
    - ajuste do nó pai para apontar para o novo nó
    - propagação de *overflow*

Exemplo

- Insira as seguintes chaves em um índice árvore-B
  - C S D T A M P I B W N G U R K E H O L J Y Q Z F X V
- Ordem da árvore-B: 4
  - em cada nó (página de disco)
    - número de chaves: 3
    - número de ponteiros: 4

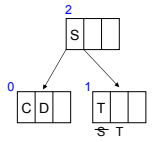
C S D T A M P I B W N G U R K ...

- Passo 1 – inserção de C, S, D
  - criação do nó raiz
    - C
    - C S
    - C D S

C S D T A M P I B W N G U R K ...

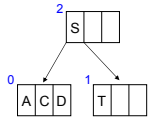
- Passo 2 – inserção de T
  - nó raiz cheio

• particionamento do nó
• criação de uma nova raiz
• promoção de S



C S D T A M P I B W N G U R K ...

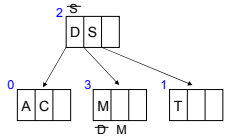
- Passo 3 – inserção de A
  - nó folha com espaço



C S D T A M P I B W N G U R K ...

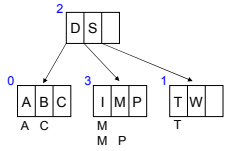
- Passo 4 – inserção de M
  - nó folha 0 cheio

• particionamento do nó
• promoção de D



C S D T A M P I B W N G U R K ...

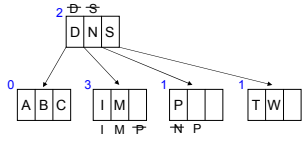
- Passo 5 – inserção de P, I, B, W
  - nós folhas com espaço



C S D T A M P I B W N G U R K ...

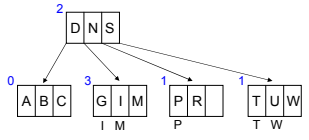
- Passo 6 – inserção de N
  - nó folha 3 cheio

• particionamento do nó
• promoção de N



C S D T A M P I B W N G U R K ...

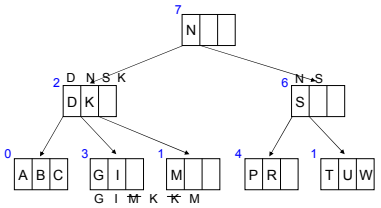
- Passo 7 – inserção de G, U, R
  - nós folhas com espaço



C S D T A M P I B W N G U R K ...

- Passo 8 – inserção de K
  - nó folha 3 cheio

• particionamento do nó 3
• promoção de K
• particionamento do nó 2
• promoção de N



... E H O L J Y Q Z F X V

- Finalizar a construção da árvore

Exercícios

- Na árvore-B do exemplo anterior, insira a chave $, sendo que $ < A.
- Insira as seguintes chaves em um índice árvore-B
  - C S D T A M P I B W N G U R K E H O L J Y Q Z F X V
    - diferentemente do exemplo anterior, escolha o último elemento do primeiro nó para promoção durante o particionamento do nó.