Árvore B*

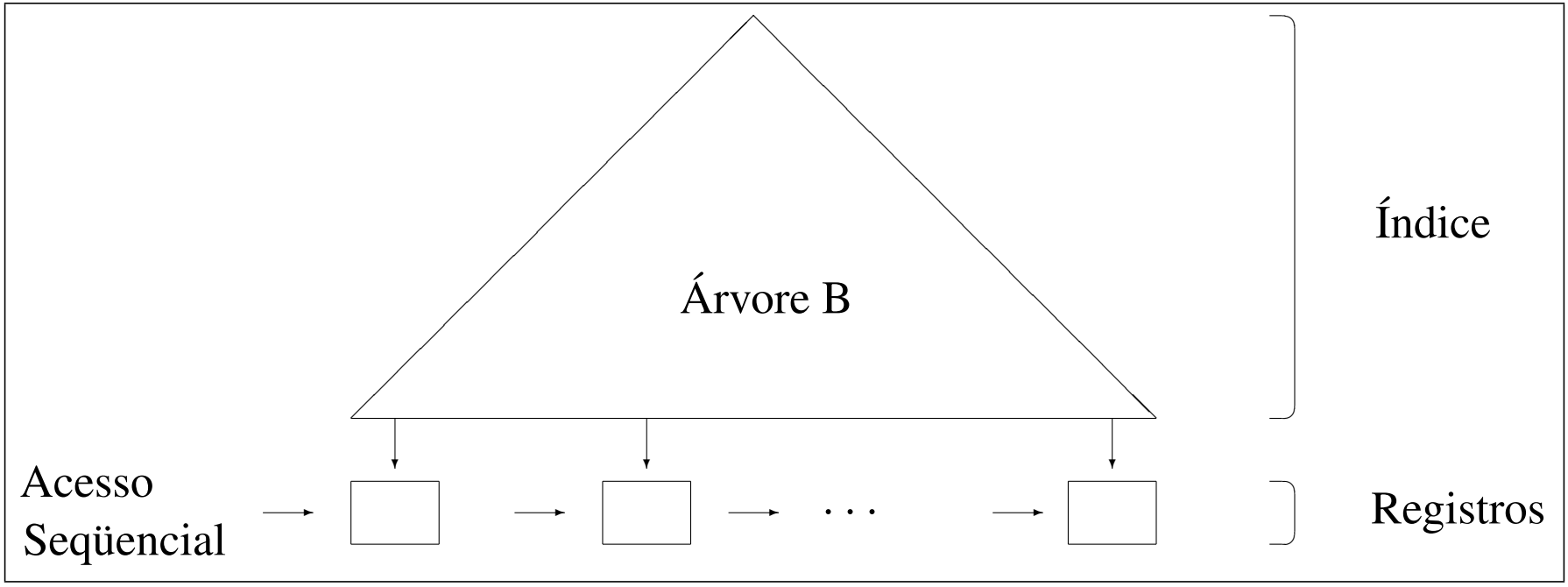
Fabiano C. Botelho

fabiano@decom.cefetmg.br

http://www.dcc.ufmg.br/~fbotelho

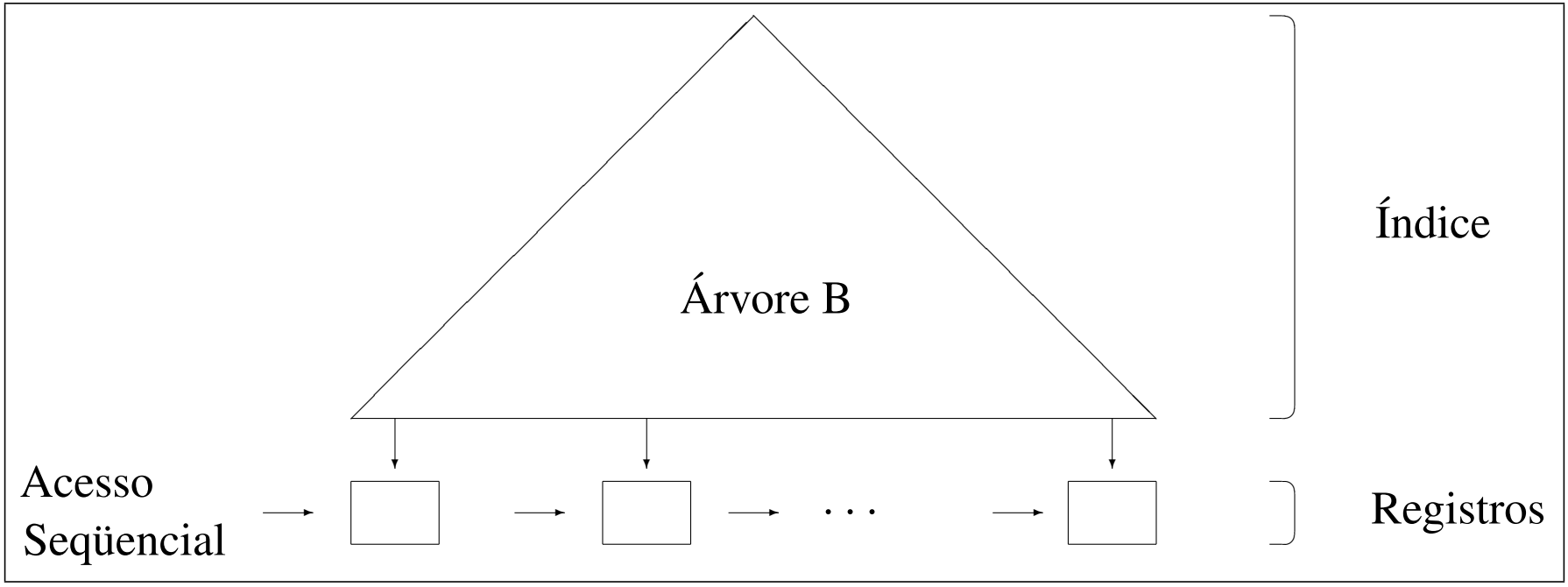
Definição

- Todos os registros são armazenados no último nível (páginas folhas).
- Os níveis acima do último nível constituem um índice organizado como uma árvore B (só armazenam as chaves).



Vantagens

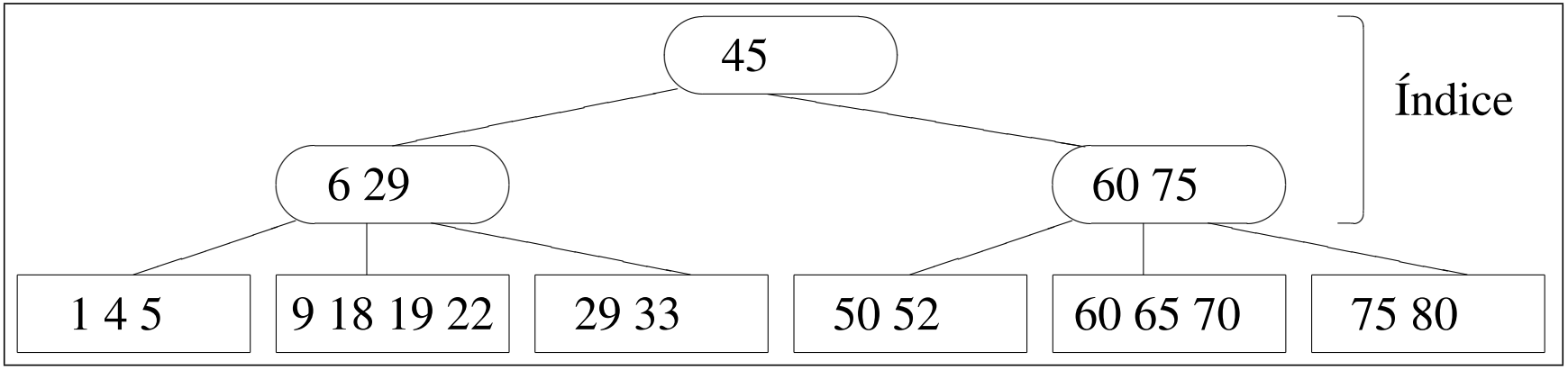
- Facilitam acesso seqüencial
- Facilitam acesso concorrente



Exemplo e Processo de Pesquisa

- O processo de pesquisa inicia-se na raiz da árvore e continua até uma página folha.

- A pesquisa não pára se a chave procurada for encontrada em uma página do índice:
  - Registros residem nas folhas
  - Os valores encontrados ao longo do caminho são irrelevantes desde que eles conduzam à página folha correta.



Armazenando mais registros em páginas folhas

- Não há necessidade do uso de apontadores nas páginas folha
- É possível utilizar este espaço para armazenar uma quantidade maior de registros em cada página folha.
- Dois valores de m são utilizados (um para o índice e outro para páginas folha)
- Isto não cria nenhum problema para os algoritmos de inserção, pois as metades de uma página que está sendo particionada permanecem no mesmo nível da página original antes da partição (algo semelhante acontece com a retirada de registros)

Estrutura de Dados

```
public class ArvoreBEstrela  {
  private static abstract class Pagina {
    int n; Item chaves[];
  }
  private static class PaginaInt extends Pagina {
    Pagina p[];
    public PaginaInt (int mm) {
      this.n = 0; this.chaves = new Item[mm];
      this.p = new Pagina[mm+1];
    }
  }
  private static class PaginaExt extends Pagina {
    Object registros[];
    public PaginaExt (int mm2) {
      this.n = 0; this.chaves = new Item[mm2];
      this.registros = new Object[mm2];
    }
  }
  private Pagina raiz;
  private int mm, mm2;
}
```

Método de Pesquisa

```
private Object pesquisa (Item chave, Pagina ap) {
  if (ap == null) return null; // Registro não econtrado
  else {
    if (this.eInterna (ap)) {
      int i = 0; PaginaInt aux = (PaginaInt)ap;
      while ((i < aux.n-1) && (chave.compara (aux.chaves[i]) > 0)) i++;
      if (chave.compara (aux.chaves[i]) < 0)
        return pesquisa (chave, aux.p[i]);
      else return pesquisa (chave, aux.p[i+1]);
    }
    else {
      int i = 0; PaginaExt aux = (PaginaExt)ap;
      while ((i < aux.n-1) && (chave.compara (aux.chaves[i]) > 0)) i++;
      if (chave.compara (aux.chaves[i]) == 0) return aux.registros[i];
      return null; // Registro não econtrado
    }
  }
}
```

Método de Inserção

- A operação de **Inserção** de um registro em uma árvore B* é essencialmente igual à inserção de um registro em uma árvore B

- A única diferença é que, quando uma folha é dividida em duas, o algoritmo promove uma cópia da chave que pertence ao registro do meio para a página pai no nível anterior, retendo o registro do meio na página folha da direita.

Exercício: Implementar esta operação

Método de Retirada

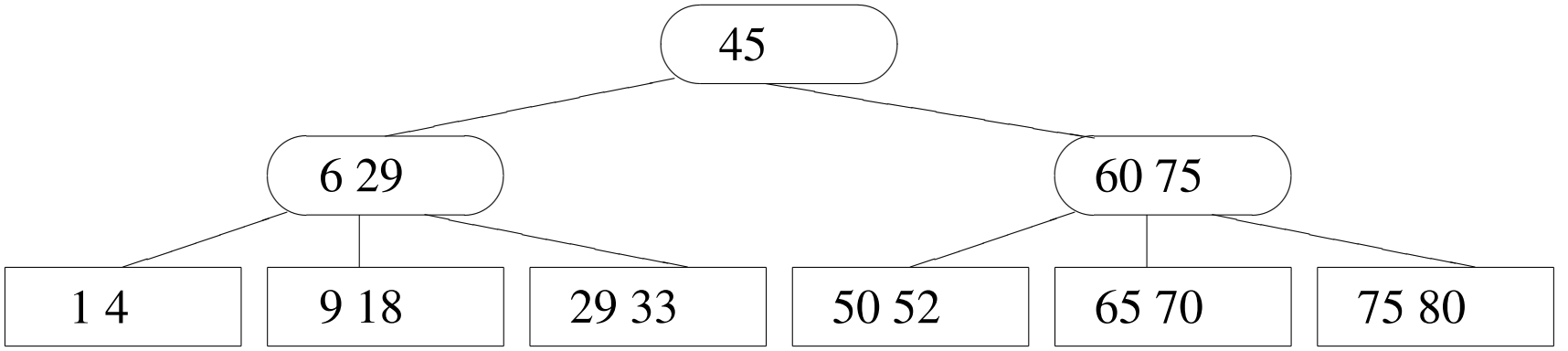
- A operação de **Retirada** em uma árvore B* é relativamente mais simples do que em uma árvore B

- O registro a ser retirado reside sempre em uma página folha, o que torna sua remoção simples, não havendo necessidade de utilização do método para localizar a chave antecessora

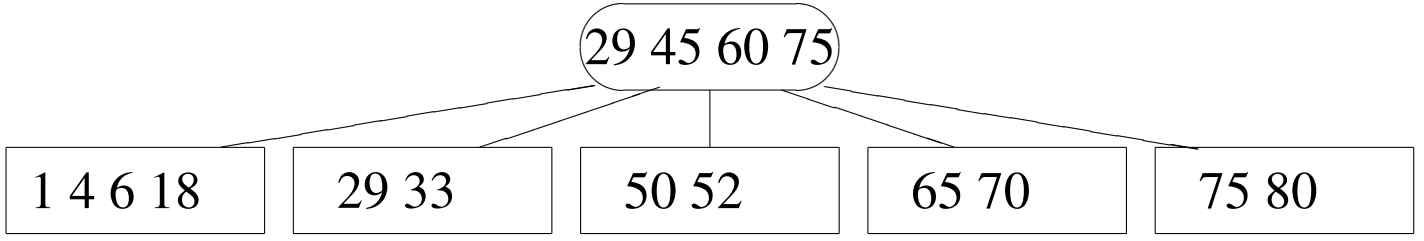
- Desde que a página folha fique pelo menos com metade dos registros, as páginas do índice não precisam ser modificadas, mesmo se uma cópia da chave que pertence ao registro a ser retirado esteja no índice

Exercício: Implementar esta operação

Exemplo de Retirada



(a) Retirada dos registros 5, 19, 22, 60 da árvore da Figura da Transparência 4



(b) Retirada do registro 9 em (a): a estrutura do índice é modificada

Acesso Concorrente em Árvores B*

- Acesso simultâneo a banco de dados por mais de um usuário.
- Concorrência aumenta a utilização e melhora o tempo de resposta do sistema.
- O uso de árvores B* nesses sistemas deve permitir o processamento simultâneo de várias solicitações diferentes.
- Necessidade de criar mecanismos chamados protocolos para garantir a integridade tanto dos dados quanto da estrutura.

Páginas Seguras Para Acesso Concorrente

- Inserção:
  - página segura se o número de chaves é igual a $2m$
- Remoção:
  - página segura se o número de chaves é maior que m
- Os algoritmos para acesso concorrente fazem uso dessa propriedade para aumentar o nível de concorrência.

Protocolos de Travamento

- Quando uma página é lida, a operação de recuperação a trava, assim, outros processos, não podem interferir com a página.

- A pesquisa continua em direção ao nível seguinte e a trava é liberada para que outros processos possam ler a página.

- Processo leitor $\longrightarrow$ executa uma operação de recuperação

- Processo modificador $\longrightarrow$ executa uma operação de inserção ou retirada.

- Tipos de Travamento:
  - Travamento para leitura $\longrightarrow$ permite um ou mais leitores acessarem os dados, mas não permite inserção ou retirada.
  - Travamento exclusivo $\longrightarrow$ nenhum outro processo pode operar na página e permite qualquer tipo de operação na página.

Considerações Práticas

- Simples, fácil manutenção, eficiente e versátil.
- Permite acesso seqüencial eficiente.
- Custo para recuperar, inserir e retirar registros do arquivo é logaritmico.
- Espaço utilizado é, no mínimo 50% do espaço reservado para o arquivo.
- Emprego onde o acesso concorrente ao banco de dados é necessário, é viável e relativamente simples de ser implementado.
- Inserção e retirada de registros sempre deixam a árvore balanceada.
- Uma árvore B de ordem m com N registros contém no máximo cerca de $\log_{m+1} N$ páginas.

Considerações Práticas

- Limites para a altura máxima e mínima de uma árvore B de ordem m com N registros:

$$\log_{2m+1}(N+1) \leq h \leq 1 + \log_{m+1}\left(\frac{N+1}{2}\right)$$

- Custo para processar uma operação de recuperação de um registro cresce com o logaritmo base m do tamanho do arquivo.

- Altura esperada: não é conhecida analiticamente.

- Há uma conjectura proposta a partir do cálculo analítico do número esperado de páginas para os quatro primeiros níveis (das folha em direção à raiz) de uma **árvore 2-3** (árvore B de ordem $m = 1$).

- Conjetura: a altura esperada de uma árvore 2-3 **randômica** com N chaves é $h(N) \approx \log_{7/3}(N+1)$

Outras Medidas de Complexidade

- A utilização de memória é cerca de $\ln 2$:
  - Páginas ocupam $\approx 69\%$ da área reservada após N inserções randômicas em uma árvore B inicialmente vazia.

- No momento da inserção, a operação mais cara é a partição da página quando ela passa a ter mais do que $2m$ chaves. Envolve:
  - Criação de nova página, rearranjo das chaves e inserção da chave do meio na página pai localizada no nível acima.
  - $\Pr\{j \text{ partições}\}$: probabilidade de que j partições ocorram durante a N-ésima inserção randômica.
  - Árvore 2-3: $\Pr\{0 \text{ partições}\} = 4/7$ e $\Pr\{1 \text{ ou mais}\} = 3/7$
  - Árvore B de ordem m: $\Pr\{0 \text{ partições}\} = 1 - 1/(2 \ln 2)m + O(m^{-2})$ e $\Pr\{1 \text{ ou mais}\} = 1/(2 \ln 2)m + O(m^{-2})$
  - Árvore B de ordem $m = 70$: 99% das vezes nada acontece em termos de partições durante uma inserção.

Acesso Concorrente (Continuação)

- Foi proposta uma técnica de aplicar um travamento na página segura mais profunda (Psmp) no caminho de inserção.

- Uma página segura é a mais profunda se não existir outra página segura abaixo dela.

- Já que o travamento da página impede o acesso de outros processos, é interessante saber qual é a probabilidade de que a página segura mais profunda esteja no primeiro nível:
  - Árvore 2-3: $\Pr\{\text{Psmp esteja no primeiro nível}\} = 4/7$ e
    $\Pr\{\text{Psmp esteja acima do primeiro nível}\} = 3/7$
  - Árvore B de ordem m:
    $\Pr\{\text{Psmp esteja no primeiro nível}\} = 1 - 1/(2 \ln 2)m + O(m^{-2})$ e
    $\Pr\{\text{Psmp esteja acima do primeiro nível}\} = 1/(2 \ln 2)m + O(m^{-2})$

Acesso Concorrente (Continuação)

- Novamente, em árvores B de ordem $m = 70$: 99% das vezes a Psmp está em uma folha. (Permite alto grau de concorrência para processos modificadores.)

- Soluções muito complicadas para permitir concorrência de operações em árvores B não trazem grandes benefícios.

- Na maioria das vezes, o travamento ocorrerá em páginas folha. (Permite alto grau de concorrência mesmo para os protocolos mais simples.)

Técnicas de Transbordamento

- Assuma que um registro tenha de ser inserido em uma página cheia, com $2m$ registros.
- Em vez de particioná-la, olhamos primeiro para a página irmã à direita.
- Se a página irmã possui menos do que $2m$ registros, um simples rearranjo de chaves torna a partição desnecessária.
- Se a página à direita também estiver cheia ou não existir, olhamos para a página irmã à esquerda.
- Se ambas estiverem cheias, então a partição terá de ser realizada.
- Efeito da modificação: produzir uma árvore com melhor utilização de memória e uma altura esperada menor.
- Produz uma utilização de memória de cerca de 83% para uma árvore B randômica.

Influência do Sistema de Paginação

- O número de níveis de uma árvore B é muito pequeno (três ou quatro) se comparado com o número de molduras de páginas.
- Assim, o sistema de paginação garante que a página raiz esteja sempre na memória principal (se for adotada a política LRU).
- O esquema LRU faz também com que as páginas a serem particionadas em uma inserção estejam automaticamente disponíveis na memória principal.
- A escolha do tamanho adequado da ordem m da árvore B é geralmente feita levando em conta as características de cada computador.
- O tamanho ideal da página da árvore corresponde ao tamanho da página do sistema, e a transferência de dados entre as memórias secundária e principal é realizada pelo sistema operacional.
- Estes tamanhos variam entre 512 bytes e 4.096 bytes, em múltiplos de 512 bytes.