SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Árvores B* e B+

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Árvores B*

Árvore B*

- Árvore-B* (B*Tree)
  - Reduz o desperdício de espaço que pode ocorrer na árvore-B

Árvore B*

- Principais características:
  - Ocupação mínima
    - Cerca de 2/3 do número máximo de chaves
    - Visa reduzir o desperdício de espaço que ocorre na árvore-B (ocupação mínima de cerca de 50%)

Árvore B*

➔ Na Inserção

1. Tenta re-distribuir as chaves antes de particionar o nó
    - Particionamento é adiado até que páginas irmãs também estejam cheias
2. Divisão do conteúdo de 2 páginas irmãs em 3 páginas (*two-to-three split*)

Árvore B*

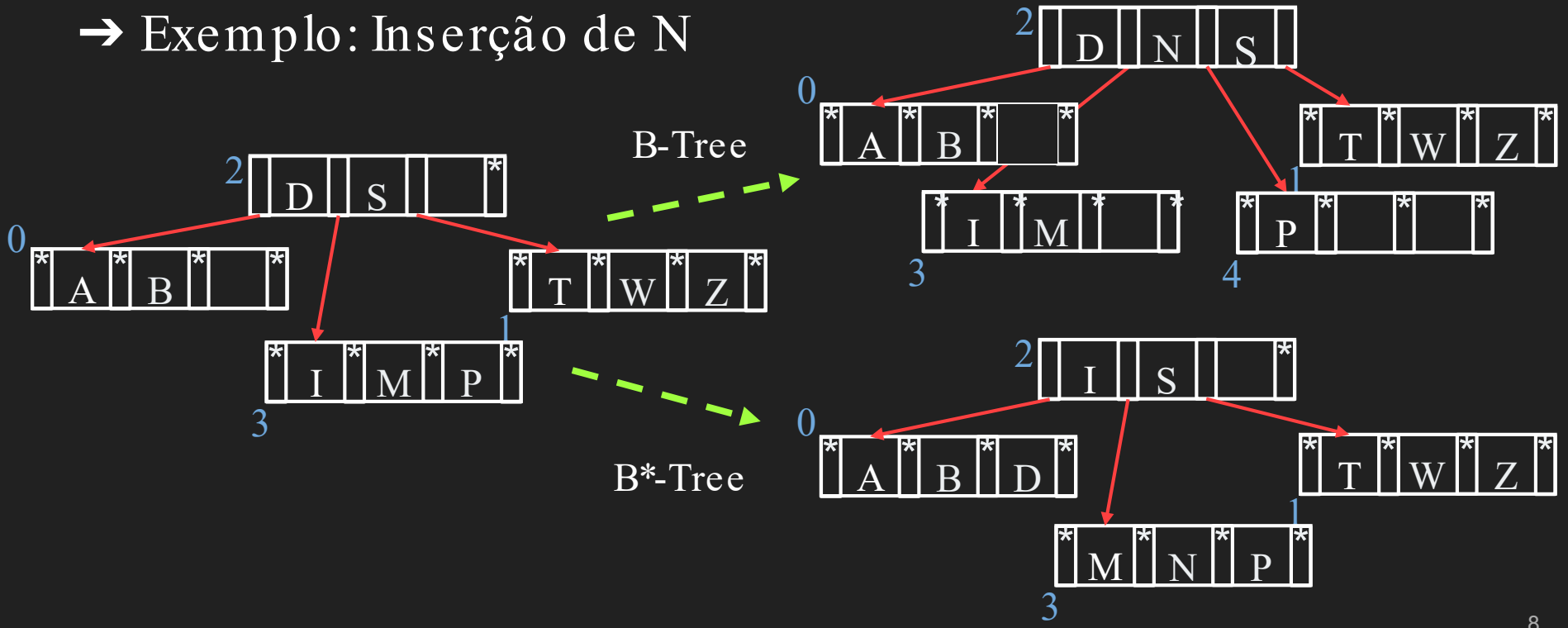
- ➔ Particionamento
  - ◆ Efeito pode se propagar
- ➔ Redistribuição
  - ◆ Efeito local
  - ◆ Pode ser utilizada na Inserção
  - ◆ Em caso de *overflow*
    - Redistribuir as chaves da página e a nova chave em páginas irmãs ao invés de particionar uma página cheia em duas páginas novas semi-vazias

Árvore B*

- ➔ Redistribuição
  - Adia particionamento
  - Melhor utilização do espaço alocado para a árvore

Redistribuição na Inserção

➔ Exemplo: Inserção de N



Redistribuição na Inserção

- ➔ Redistribuição X Particionamento
  - ◆ Depois do particionamento
    - Cada página fica ~50% vazia
    - Utilização do espaço, no pior caso, em uma árvore-B que utiliza splitting
      - ~50%
    - Em média, para árvores “grandes” a taxa de ocupação de páginas é de ~69% (valor teórico)

Redistribuição na Inserção

- ➔ Redistribuição X Particionamento
  - ◆ Depois do particionamento
    - Estudos empíricos indicam que a utilização de redistribuição pode elevar esse índice para ~ 85%
- ➔ Redistribuição pode (e deve) ser usada também na árvore-B em aplicações com grandes volumes de dados

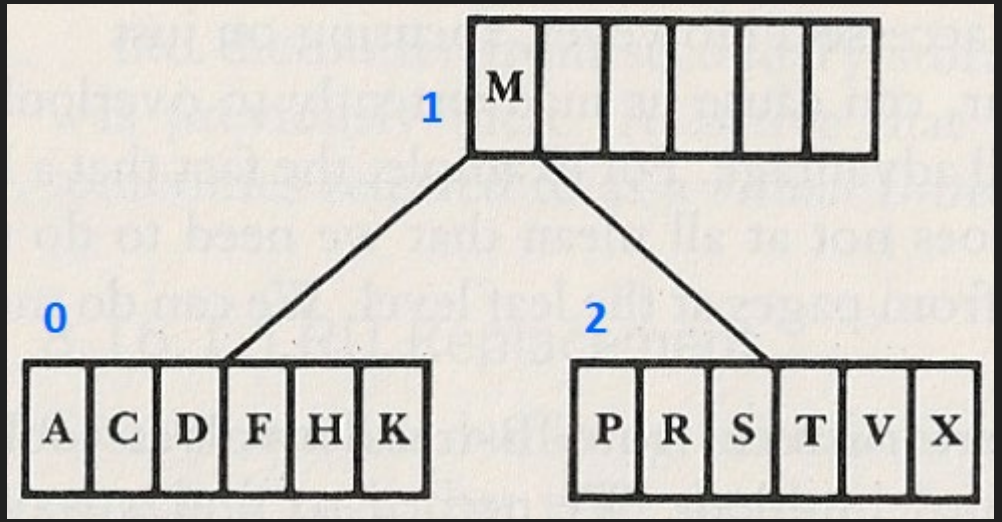
Two-to-Three Split

Two-to-Three Split

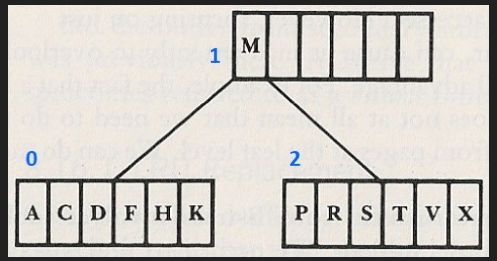
- ➔ Na Inserção
  - ◆ Particionamento é adiado até que páginas irmãs também estejam cheias
  - ◆ Quando não é mais possível realizar redistribuição
- ➔ Two-to-three split (split 2-3)
  - ◆ Divisão do conteúdo de 2 páginas irmãs + a chave nova em 3 páginas
  - ◆ Cada página com cerca de 2/3 de ocupação

Two-to-Three Split

➔ Inserção de B



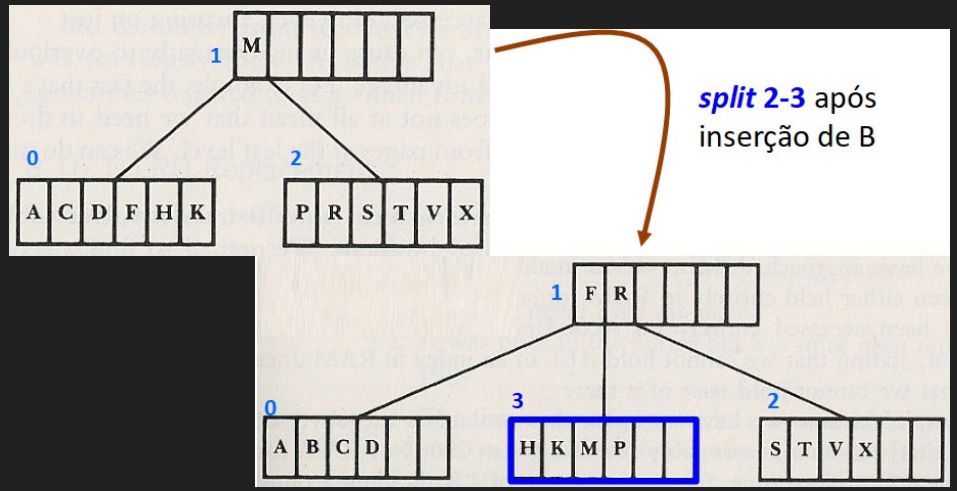
Two-to-Three Split



1. A pág. cheia da esquerda fornece 1/3 de suas maiores chaves para a nova página
2. A pág. cheia da direita fornece 1/3 de suas menores chaves para a nova página
3. A nova chave é inserida na página apropriada
   a. Se for na página esquerda, maior chave desta é deslocada para a nova página
   b. Se for na página direita, menor chave desta é deslocada para a nova página

Two-to-Three Split

4. Chave separadora do pai é rebaixada para a nova página
5. Nova página é reordenada
6. Menor e maior chaves da nova página são promovidas



Two-to-Three Split

- ➔ Two-to-tree split afeta as regras para remoção, concatenação e redistribuição
- ➔ Raiz não tem irmã
  - ◆ Requer tratamento especial na implementação
  - ◆ Solução 1: dividir a raiz usando particionamento convencional (one-to-two split)
    - Cada página com cerca de 50% de ocupação
  - ◆ Solução 2: permitir que a raiz seja um nó com maior capacidade

Propriedades da Árvore-B*

Propriedades da Árvore B*

➔ Em uma árvore de ordem m:

- Cada página tem no máximo m descendentes
- Toda página (exceto a raiz e as folhas) tem no mínimo $\lceil (2m-1)/3 \rceil$ descendentes
- A raiz tem pelo menos 2 descendentes (a menos que seja uma folha)
- Todas as folhas estão no mesmo nível

Propriedades da Árvore B*

➔ Em uma árvore de ordem m:

- Uma página não-folha com k descendentes contém $k-1$ chaves
- Uma página folha contém:
  - No mínimo $\lceil (2m-1)/3 \rceil - 1$ chaves
  - No máximo $m-1$ chaves

Árvores B+

Acesso Sequencial e Indexado

- Em muitas aplicações, é desejável que se tenha tanto acesso indexado quanto sequencial ordenado
  - Processamento co-sequencial demanda arquivos ordenados
  - Consultas por chave eficientes demandam arquivos indexados

Acesso Sequencial e Indexado

- ➔ Exemplos (ilustrativos):
  - ◆ Arquivos da Seção de Graduação
    - Processamento das matrículas (acesso sequencial ordenado)
    - Consulta a histórico escolar de um aluno pelo ID (acesso indexado)

Acesso Sequencial e Indexado

- ➔ Exemplos (ilustrativos):
  - ◆ Arquivos de Operadora de Cartão de Crédito
    - Processamento das faturas (acesso sequencial ordenado)
    - Consulta ao status do cartão (acesso indexado)

Acesso Sequencial e Indexado

- ➔ Exemplos (ilustrativos):
  - ◆ Arquivos de Montadora de Veículos
    - Consulta a dados de veículos com número de chassi num intervalo específico (acesso sequencial ordenado – consulta por intervalo)
    - Consulta a dados de um veículo específico (acesso indexado)

Porém...

Porém...

- ➔ Já sabemos que o custo de manter o arquivo de dados ordenado em função de uma chave é usualmente inaceitável (alto custo computacional) para grandes volumes de dados
- ➔ Seria viável fazer acesso ordenado (ordem lógica) usando uma árvore-B???
- ➔ Como então conseguir realizar acesso sequencial & indexado ???

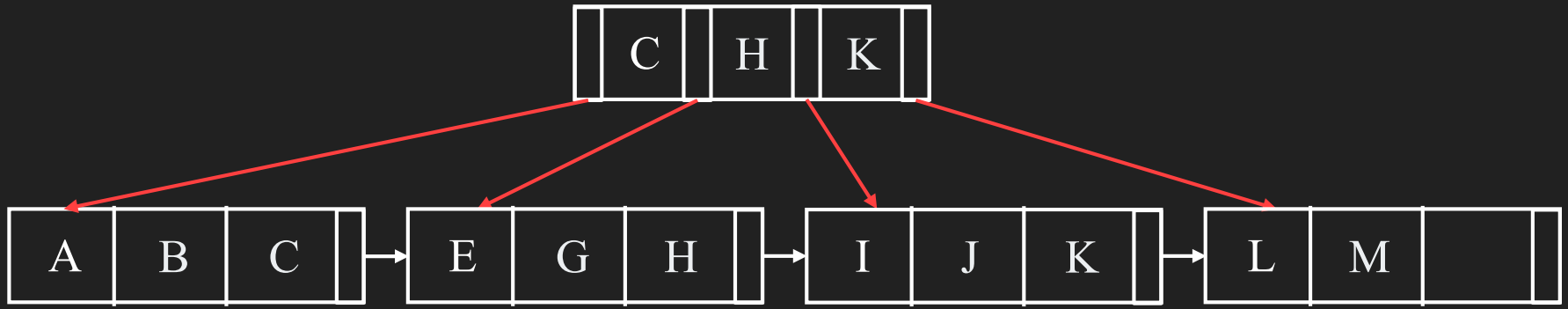
Árvores B+!

Árvores B+

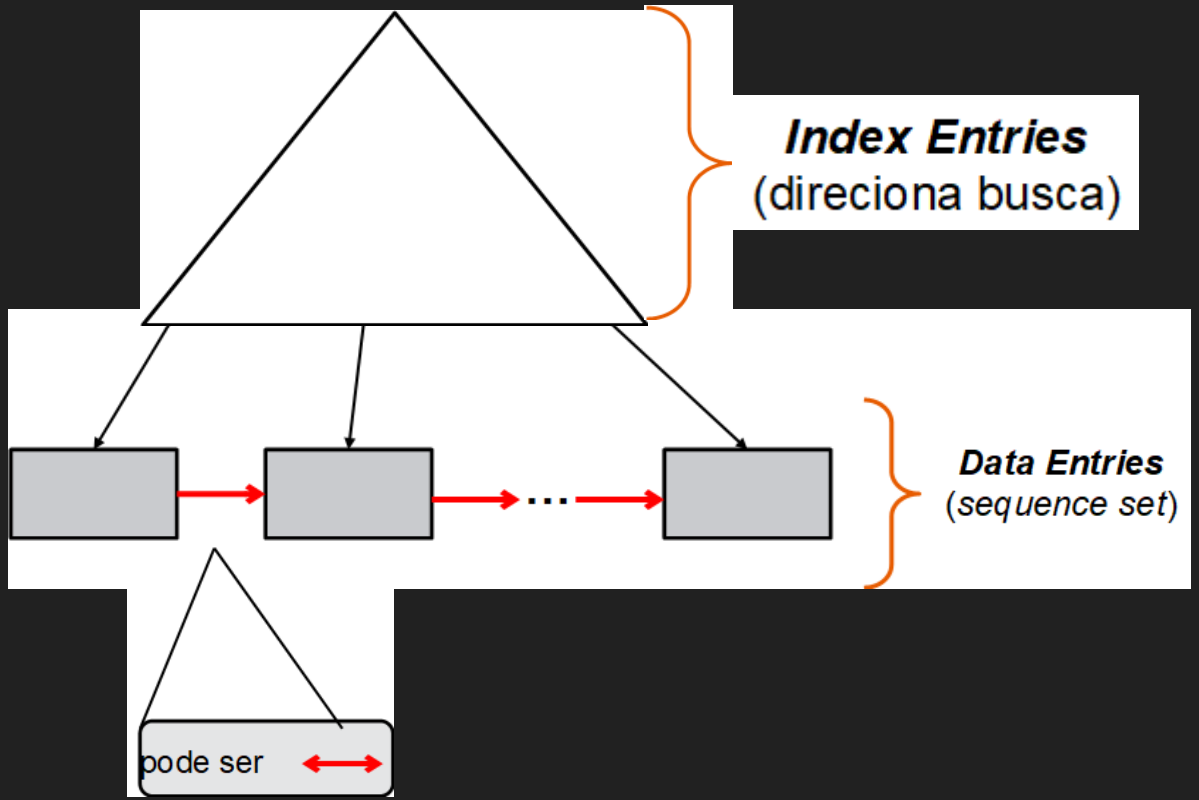
- ➔ Todas as chaves são mantidas em nós folhas e algumas são repetidas em nós não-folha (nós internos) apenas como separadores
- ➔ As folhas são “ligadas” para permitir o acesso sequencial ordenado
- ➔ Nós folha e nós internos possuem estruturas distintas

Árvores B+

- ➔ Árvores B+ são o “padrão” de Árvores-B na atualidade
- ➔ Acesso sequencial e ordenado de modo simples e eficiente + acesso indexado = Árvore-B+



Árvore-B+ - Estrutura Lógica

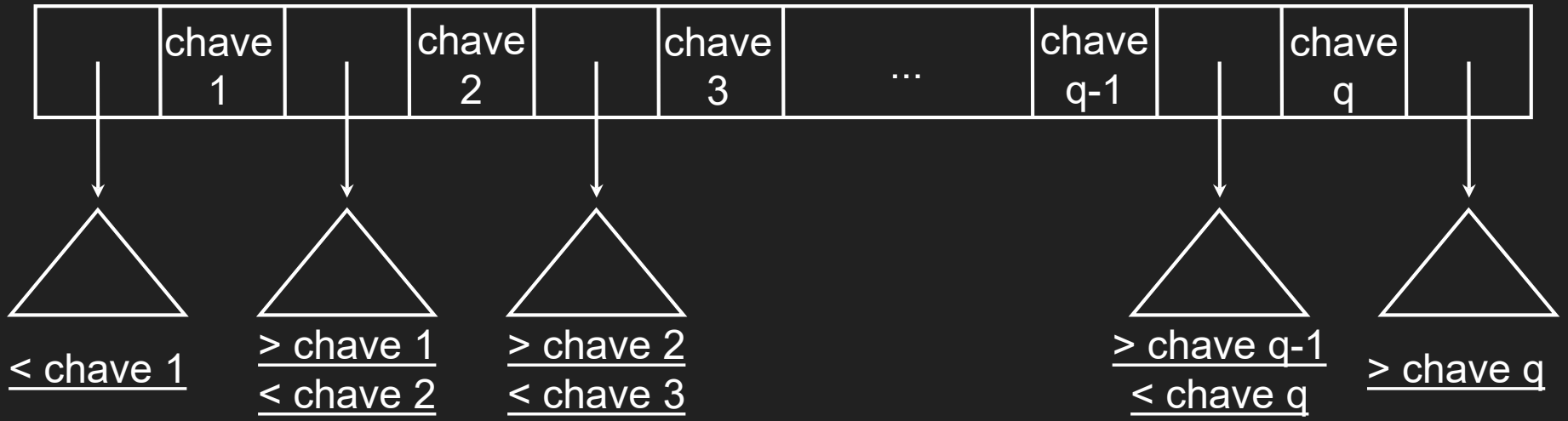


Árvores B+- Nós Internos

➔ Estrutura lógica do nó interno (não-folha)

- Página em disco – registros de tamanho fixo
- Sequência ordenada de chaves
  - Somente as chaves
    - Ponteiros para registros de dados NÃO são armazenados (diferente da árvore-B)
- “Ponteiros” para subárvores

Estrutura do Nó Interno



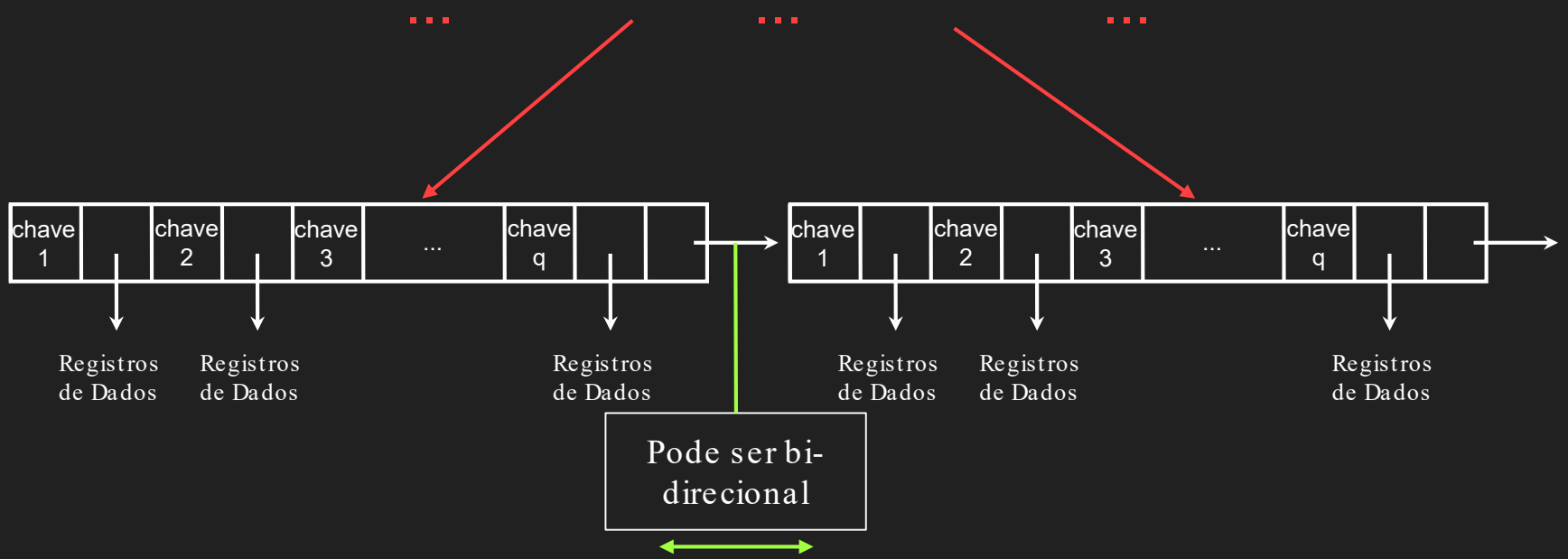
Árvores B+- Nós Folha

- Estrutura lógica dos nós folha
  - Duas abordagens principais:
    1. Páginas contendo apenas chaves e “ponteiros” para os respectivos registros completos no arquivo de dados
    2. Páginas contendo registros de dados completos

Árvores B+- Nós Folha

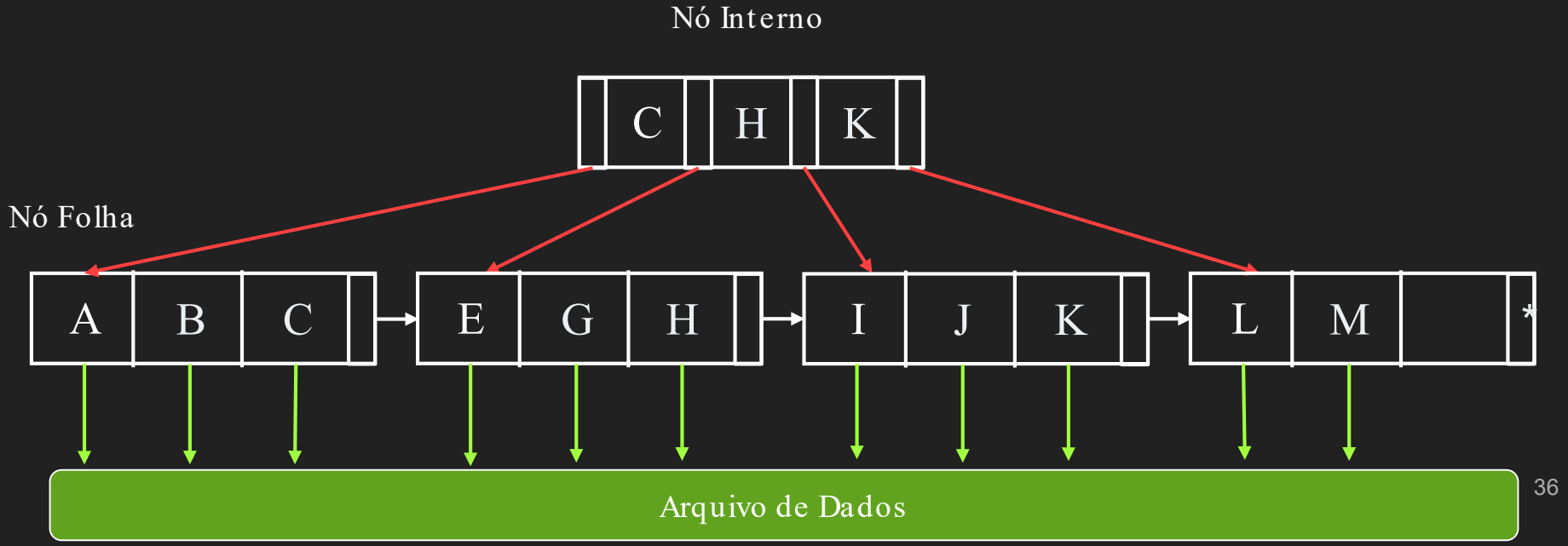
1. Páginas contendo apenas chaves e “ponteiros” para os respectivos registros completos no arquivo de dados
   - Acesso sequencial ordenado a todas as chaves, e aos registros de dados a partir dos “ponteiros”
   - Registros de tamanho fixo

ÁRVORE B+ - NÓS FOLHA



Arquivo da árvore B+ e arquivo de dados são distintos!

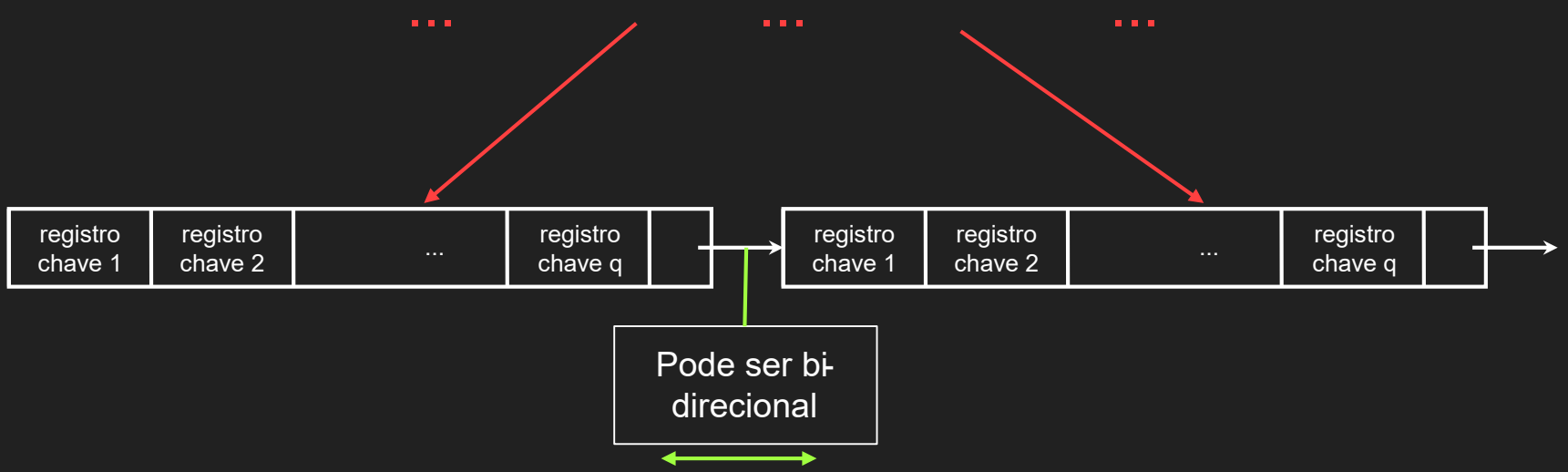
Árvores B+- Exemplo



Árvores B+- Nós Folha

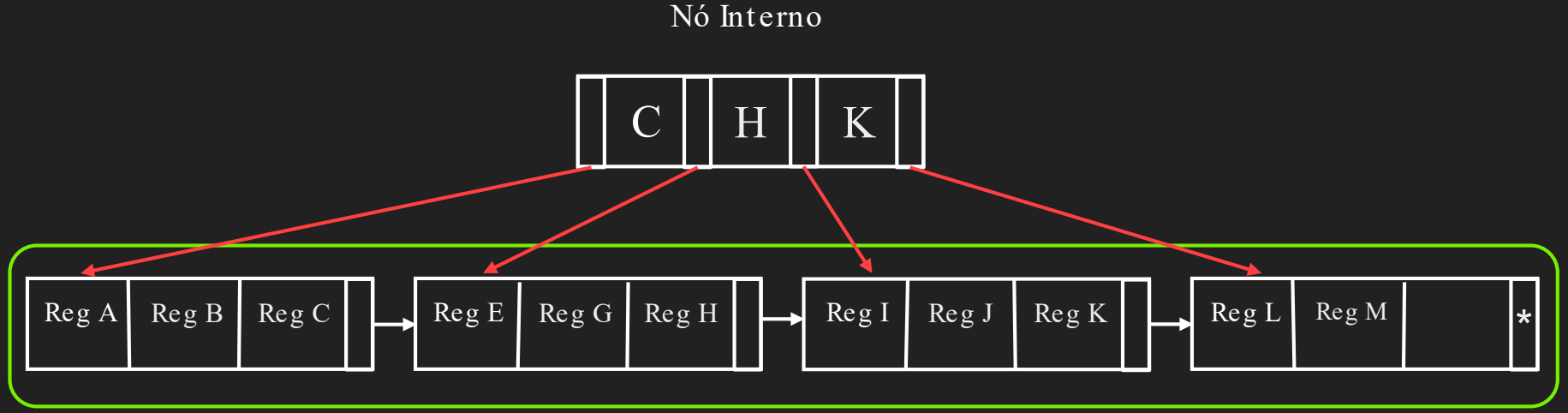
2. Páginas contendo registros de dados completos, ordenados pela chave
   - “ponteiro” para próximo nó folha permite acesso sequencial ordenado a todos os registros de dados

ÁRVORE B+ - NÓS FOLHA



Camada dos nós folha é o próprio arquivo de dados!

Árvores B+- Exemplo



Nível dos nós folha = arquivo de dados

Árvores B+- Nós Folha

2. Páginas contendo registros de dados completos
   - Registros de tamanho fixo ou variável
   - Páginas usualmente com o mesmo tamanho do bloco de disco
     - Armazenamento sequencial e ordenado dos registros no bloco do arquivo
     - Leitura da página em um único acesso
     - Ordenação lógica das páginas (blocos) no arquivo

Árvores-B+ Operações

Árvores B+ Operações

- Busca
  - Deve necessariamente ser realizada até as folhas se o objetivo for recuperar o registro de dados

Árvores B+ Operações

- ➔ Inserção
  - Sempre realizada nas folhas
  - Em caso de overflow
    - Redistribuição para adiar particionamento
    - Particionamento one-to-two
    - Chaves promovidas de nó folha para nó interno são “repetidas”

Árvores B+ Operações

- ➔ Remoção
  - ◆ Sempre realizada em nó folha
    - ● A chave também deve ser removida de nó interno se for separadora
  - ◆ Em caso de underflow
    - ● Ocupação mínima
      - ○ ~50%
- ➔ Redistribuição e/ou concatenação

Considerações

Considerações

- ➔ Até agora…
  - ◆ Chave na árvore é uma chave primária do arquivo de dados, composta por somente um atributo (um campo do registro de dados)
    - Identifica de maneira única cada registro – não há repetição de valores
    - Cada chave na árvore está associada a um único registro de dados
    - ex: NUSP, CPF, ID, ISBN, etc…

Considerações

➔ Mas é possível criar árvores-B e variantes para:

- ◆ Chaves compostas por mais de um atributo
  - ex: Label (da gravadora) + ID (da música) no Arquivo de Músicas
  - ex: Id de Estudante + Universidade num cadastro nacional

Considerações

- ➔ Chaves secundárias – há repetição de valores
  - Chave na árvore pode estar associada a múltiplos registros de dados
  - ex: nome, nacionalidade, idade, endereço, etc…

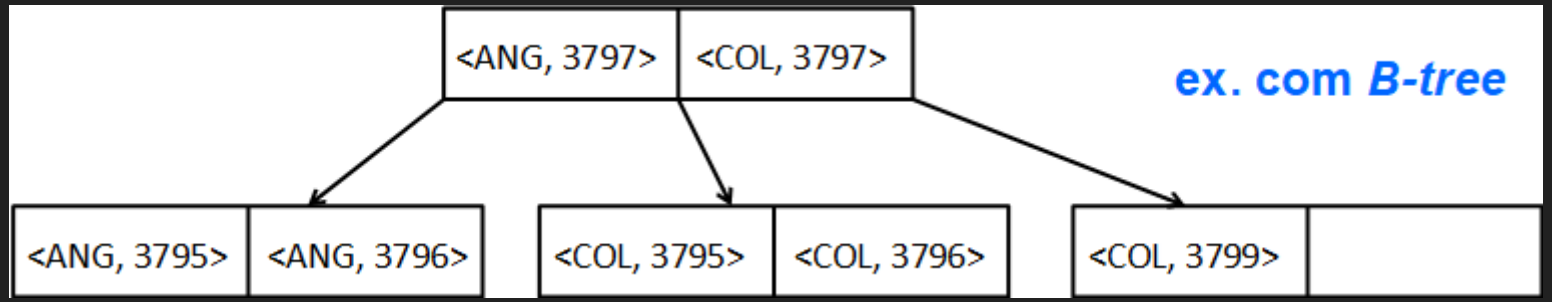
Considerações

- ➔ Chaves compostas por mais de um atributo
  - ◆ Chave é uma combinação de valores
    - ex: <ANG, 3795> (Label, ID)
    - Ordem das chaves para construção árvore é definida pela ordem lexicográfica dos valores
    - ex: <ANG, 3795> <ANG, 3796>, <ANG, 3797>, <COL, 3795>, <COL, 3796>, <COL, 3797>, <COL, 3799>, <DG, 18807> ....

Considerações

➔ Chaves compostas por mais de um atributo

- ◆ Qual a influência da ordem dos atributos na chave?
  - Há diferença entre <Label, ID> e <ID, Label>
  - Necessário analisar as consultas mais críticas realizadas a partir do índice



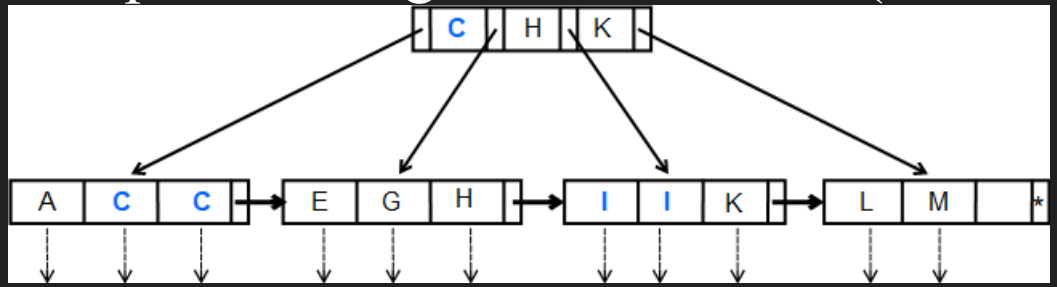
Considerações

- ➔ Chaves com repetição de valores
  - ◆ Algumas abordagens:
  - ◆ Em B+
    - Manter todas as chaves repetidas nos nós folha, respeitando a ordem
    - Cada chave com o ponteiro para o seu respectivo registro de dados (ou com o registro completo)

Considerações

➔ Chaves com repetição de valores

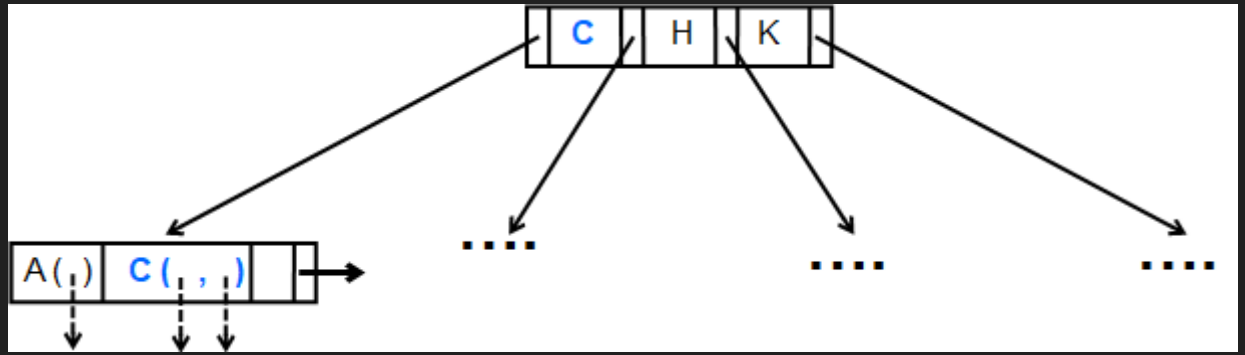
- ◆ Em B+
  - Manter todas as chaves repetidas nos nós folha, respeitando a ordem
    - Cada chave com o ponteiro para o seu respectivo registro de dados (ou com o registro



Considerações

➔ Chaves com repetição de valores em B+ (alternativa)

- Manter apenas uma "cópia" da chave na folha, com uma lista de ponteiros para os registros de dados



Considerações

➔ Chaves com repetição de valores em B, B* ou B+

- Clustered index
- Manter o arquivo de dados com uma ordenação lógica por blocos (similar ao nível de folhas da B+)
- No arquivo de índice, a chave aparece apenas uma vez associada a um ponteiro para o início do primeiro bloco no arquivo de dados que contém registros correspondentes à chave

Referências

- M. J. Folk and B. Zoellick, File Structures: A Conceptual Toolkit, Addison Wesley, 1987.
- R. Elmasri, S. Navathe. Sistemas de Banco de Dados, Person, 6a Edição, 2010.
- R. Ramakrishnan, J. Gehrke. Database Management Systems. McGraw Hill, 3rd Edition, 2003.