GBC053 - Gerenciamento de Bancos de Dados

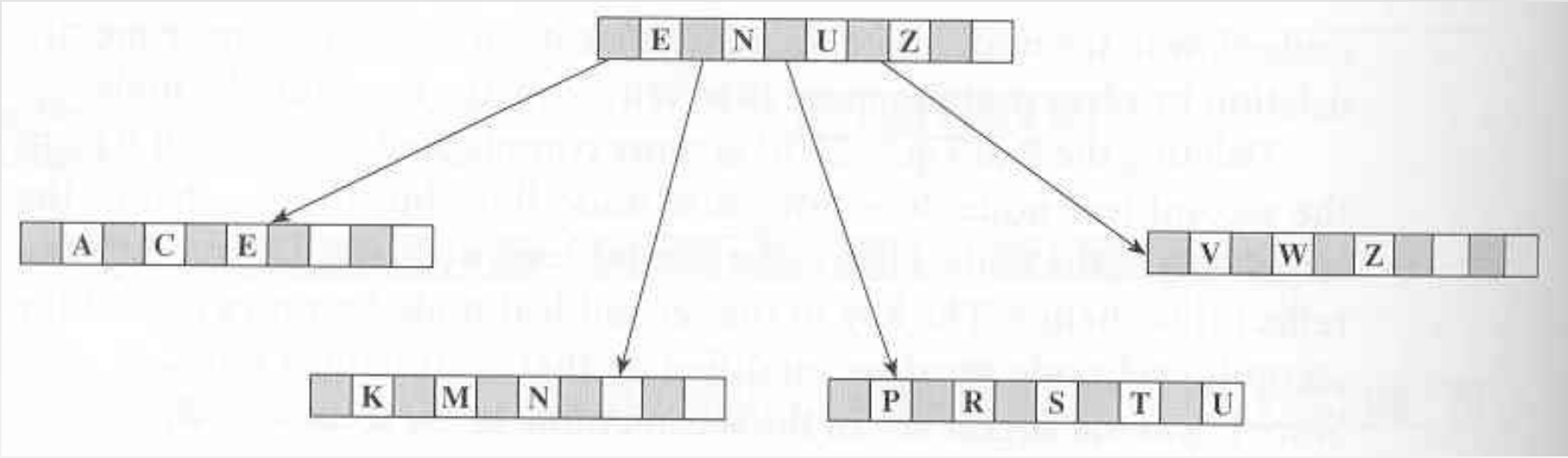
Aula 13
Árvores B e árvores Prefix B+

Humberto Razente
humberto.razente@ufu.br

Árvores B – revisão

- Uma árvore B é um índice multinível que resolve o problema do custo linear da inserção e remoção
  - não requer que páginas índice sejam completas
  - não extende overflow para próxima página
    - overflow gera divisão da página em duas, cada uma com metade dos registros
  - remoção tem estratégia semelhante
    - pode resultar na junção de páginas quando necessário

Árvores B – revisão



Árvores B – revisão

- Uma árvore B é uma árvore *n*-ária que pode ser uma estrutura de armazenamento muito eficiente e flexível
  - mantém propriedades de balanceamento mesmo após remoções e inserções
  - provém busca a qualquer chave com poucos acessos a disco

- Entretanto, não é porque uma árvore B tem 3 níveis, que toda busca tenha que fazer 3 acessos a disco
  - pode-se fazer melhor que isso!!!

Árvore B Virtual

- Buffering
  - Manter a página raiz em um buffer resulta em 1 acesso a disco a menos a cada busca
- Extensão: manter parte da árvore em memória
  - uso de um buffer de páginas para guardar um certo número de páginas da árvore B
  - ao ler uma página do disco em resposta à uma requisição do usuário, o buffer é preenchido
  - ao executar outra consulta, verifica-se se a página requisitada está no buffer
  - se a página não está em memória, ela é lida do disco para o buffer, substituindo alguma página previamente lida

Árvore B Virtual

- Substituição LRU (**L**east **R**ecently **U**sed)

- Acessar no disco com intuito de trazer uma página que não está no buffer é chamado de falha de página (*page fault*)
  - a página não foi usada
  - a página já esteve no buffer, mas foi substituída por uma página mais nova

Árvore B Virtual

- O segundo caso pode ser minimizado com o gerenciamento do buffer
  - A decisão mais crítica é qual página substituir quando o buffer se encontra cheio
- Uma estratégia bastante comum é substituir pela página menos recentemente usada (LRU)
  - LRU (**L**east **R**ecently **U**sed)
    - Página que ficou mais tempo sem ser requisitada para uso

Árvore B Virtual

- Substituição LRU
  - baseia-se no fato de que é mais comum necessitar de uma página que foi recentemente usada do que de uma página que foi usada a mais tempo
    - página menos recentemente usada é diferente de substituir pela página menos recentemente lida: o tempo é dado pela último uso da página, e não o momento em que ela foi lida
  - localidade temporal: assume algum tipo de agrupamento no uso das páginas ao longo do tempo
    - hipótese pode não ser sempre válida, mas aplica-se bem nas árvores B

Árvore B Virtual

- Efeito do uso de mais buffers com uma estratégia simples LRU (Webster, 1980):

Buffer Count	1	5	10	20
Average Accesses per Search	3.00	1.71	1.42	0.97

Number of keys = 2400
Total pages = 140
Tree height = 3 levels

Árvore B Virtual

- Substituição baseada na altura da página
  - modo mais direto que a estratégia LRU para guiar as decisões de substituição de página no buffer
  - reter no buffer as páginas que ocorrem nos primeiros níveis da árvore

Árvore B Virtual

- Substituição baseada na altura da página
  - Exemplo: 256 KB de memória e índice de 1 MB
  - Supondo utilização de 83% (1,2 MB no disco)
  - Uso de páginas de 4 KB
    - requer pouco mais de 300 páginas
  - Supondo que cada página tem 30 descendentes (em média)
    - resultado: árvore B de três níveis:
    - uma raiz
    - 9 ou 10 páginas no segundo nível
    - e as demais no nível das folhas

Árvore B Virtual

- Substituição baseada na altura da página
  - com buffer de 64 páginas
    - é possível guardar a raiz, todas as páginas do segundo nível e ainda sobraria espaço para 50 buffers que conteriam páginas folhas
      - esse espaço de sobra poderia ser gerenciado com uma estratégia LRU
    - combinar a estratégia LRU com um peso que leva em conta a altura da página reduz o número de acessos de um buffer de 10 páginas de 1,42 para 1,12 por busca (Webster, 1980)

Chaves e registros de tamanho variável

- Em várias aplicações, a informação associada com a chave varia em comprimento
  - Exemplo: lista de strings de tamanho variável
  - pode-se tratar esse problema com um número variável de registros em uma página de árvore B
    - uma árvore B com um número variável de chaves por página não possui uma ordem única fixa
    - se há mais chaves em uma página, há mais descendentes → diminuição do número de níveis
    - há necessidade de um critério diferente para determinar quando uma página está cheia ou em underflow
      - uso de um número máximo e mínimo de bytes no lugar do número de chaves

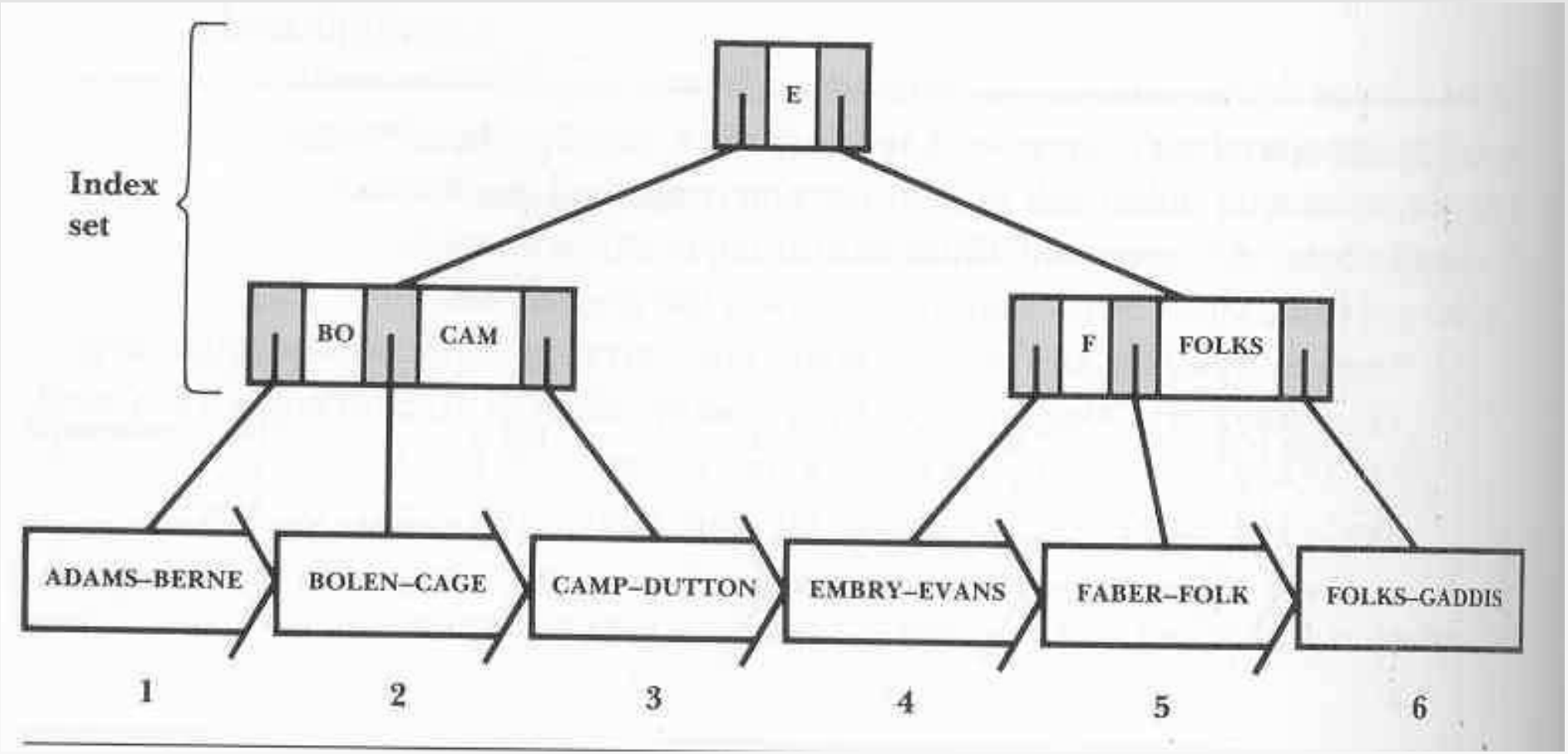
Árvore B+ de prefixo simples

- Prefixo simples → indica que o conjunto de índices contém os prefixos das chaves ao invés de cópias das chaves
  - são simples pois são simplesmente as letras iniciais das chaves
- Nos nós interiores, a última chave não é necessária
  - se o número de separadores é igual ao número de filhos (referências), o último separador é maior que qualquer chave na sub-árvore
  - nó que contém n separadores aponta para n+1 filhos

Árvore B+ de prefixo simples

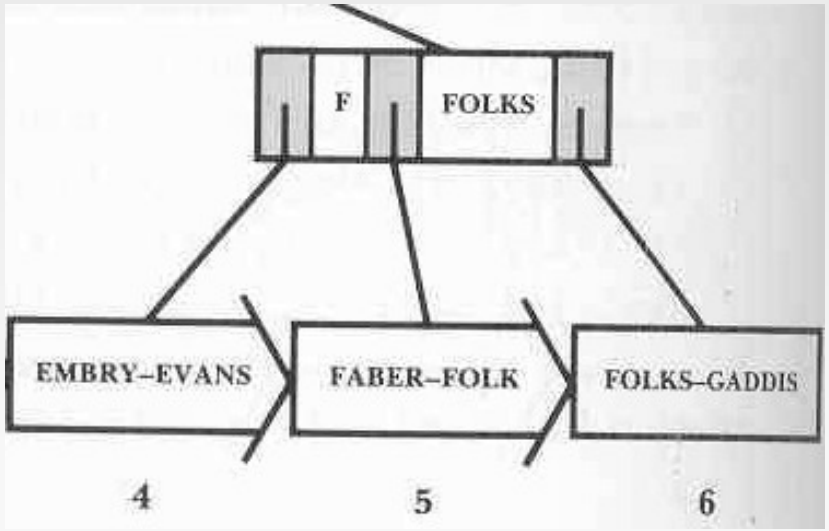
- Vantagens
  - Possibilita maior número de chaves nos índices
    - resulta em árvores com número menor de níveis
  - Possibilita comparações mais rápidas
    - comparações apenas do tamanho do prefixo ao invés do tamanho das chaves inteiras
  - Manutenção/remoção
    - podem resultar em menor número de alterações recursivas nos nós índice
  - Possui ponteiros entre os nós folha
    - em busca por faixa de valores, após encontrado o primeiro valor, não precisa voltar no índices

Árvore B+ de prefixo simples



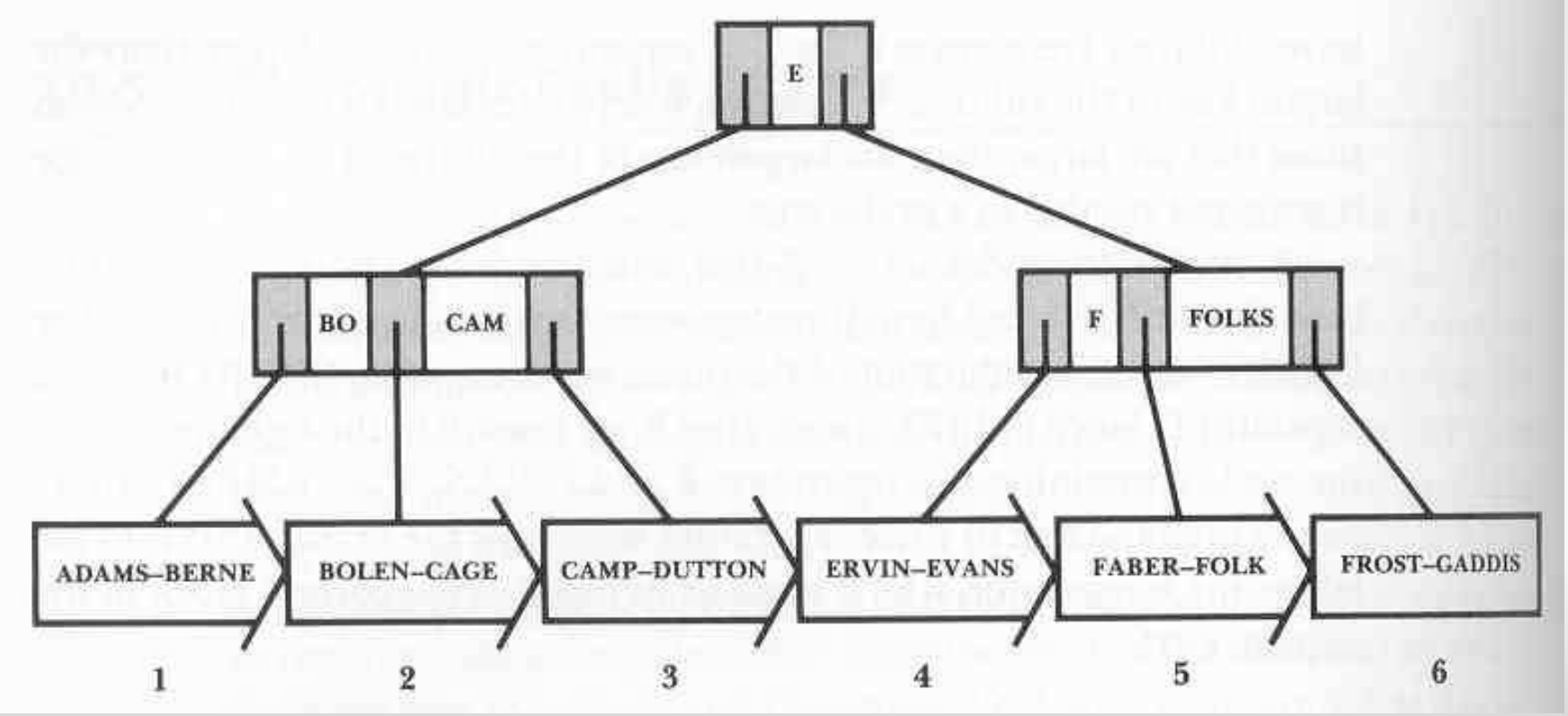
Árvore B+ de prefixo simples

- Manutenção de árvores B de prefixo simples
  - mudanças relativas à inserção ou remoção de chave podem ter efeitos localizados a um único bloco do conjunto de sequências ou envolver múltiplos blocos
    - Remoção de Folk (bloco 5) força leitura das chaves ”Folks” a ”Gaddis” (bloco 6) para atualizar índice no nível acima



Árvore B+ de prefixo simples

- Mudanças localizadas a um único bloco
  - Se uma remoção não necessita de junção ou redistribuição, os seus efeitos estão restritos a um único bloco
  - Exemplo: remoção de EMBRY



Árvore B+ de prefixo simples

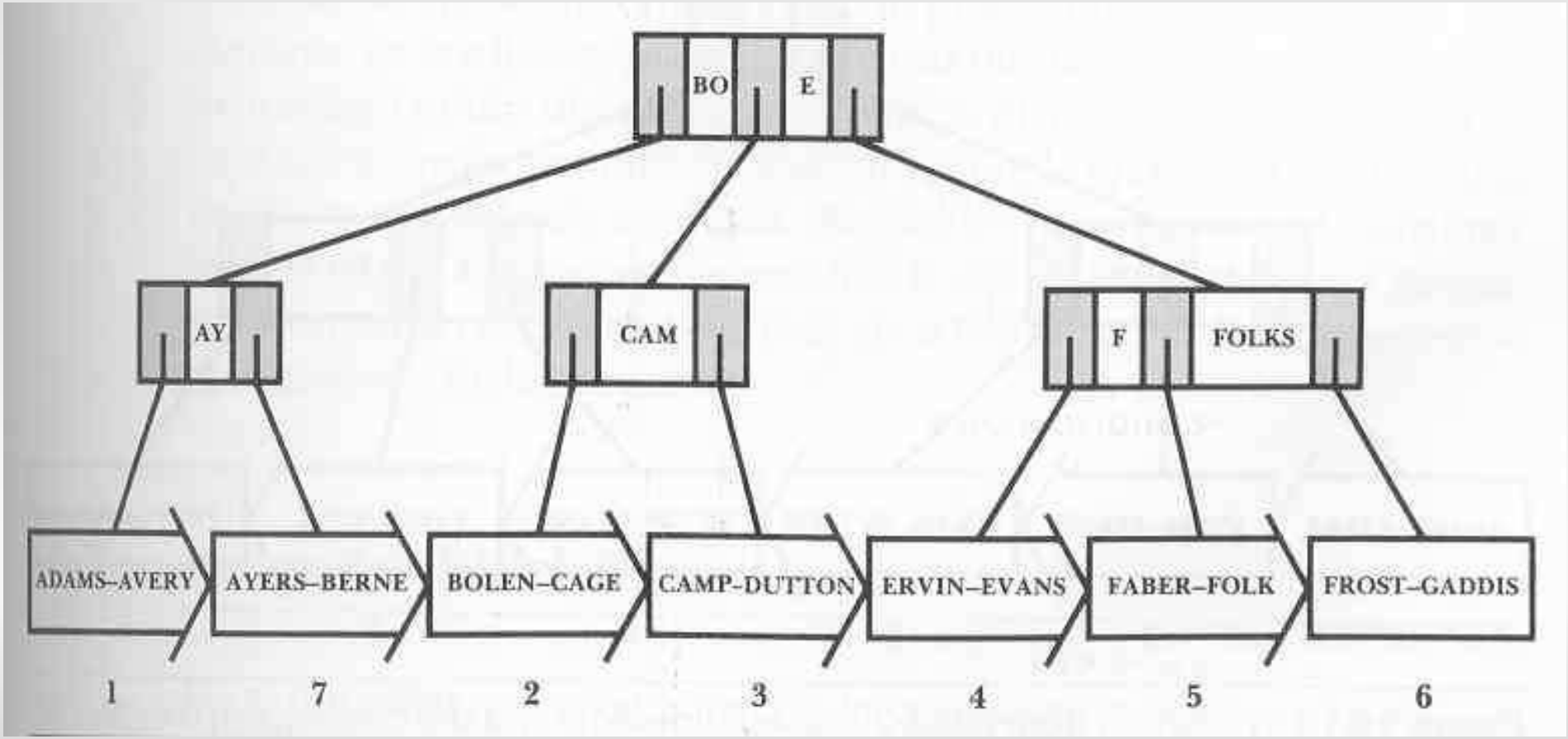
- Mudanças localizadas a um único bloco
  - conjunto de índices não foi alterado
    - os separadores continuam cumprindo seu papel
  - o efeito de uma inserção que não causa a divisão do bloco é semelhante à remoção que não resulta em junção
    - não há mudança no conjunto de índices
    - exemplo: inserção de EATON. A informação contida no conjunto de índices é suficiente para propiciar achar o registro novamente

Árvore B+ de prefixo simples

- Mudanças que envolvem múltiplos blocos
  - o que acontece se a inserção ou remoção de um registro causa um mudança no número de blocos no conjunto de seqüências?
    - mudança no número de separadores
  - inserções e remoções no conjunto de índices são gerenciadas com operações padrão em árvores B

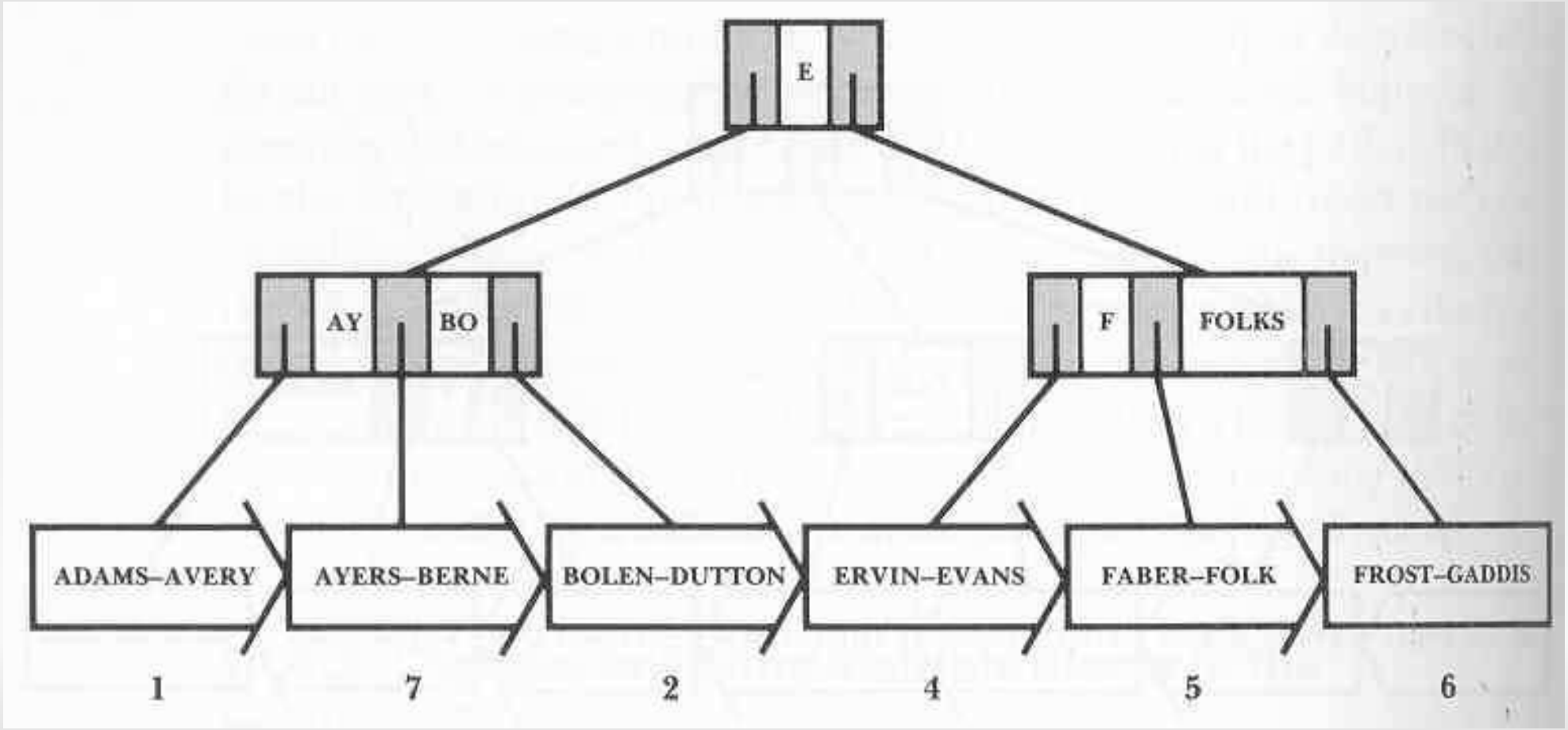
Árvore B+ de prefixo simples

- Mudanças que envolvem múltiplos blocos
  - exemplo de inserção no bloco 1 (árvore de ordem 3), que causa uma divisão do bloco



Árvore B+ de prefixo simples

- Exemplo de remoção no bloco 2 que causa *underflow* e consequente junção dos blocos 2 e 3



Árvore B+ de prefixo simples

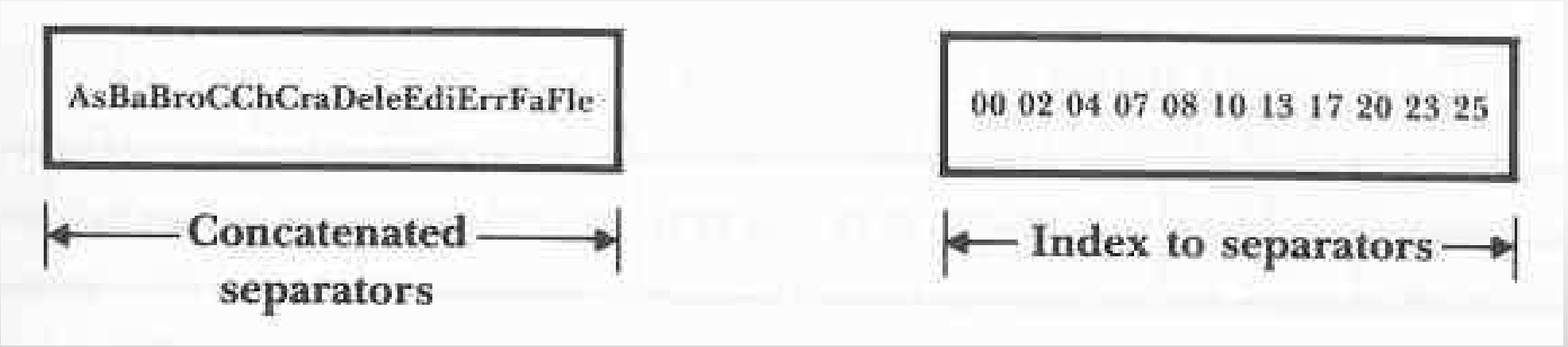
- As operações de inserção e remoção acontecem no conjunto de sequências
- Depois de realizadas no conjunto de sequências (folhas), as seguintes mudanças são necessárias nos índices
  - Se blocos são divididos no conjunto de sequências, um novo separador deve ser inserido no conjunto de índices
  - Se há junção de blocos no conjunto de sequências, um separador deve ser removido do conjunto de índices
  - Se os registros são redistribuídos entre blocos no conjunto de sequências, o valor do separador no conjunto de índices deve ser atualizado

Tamanho do bloco do conjunto de índices

- O tamanho físico do nó para o conjunto de índices é usualmente o mesmo que o tamanho físico de um bloco no conjunto de sequências
  - mesmos fatores levam à escolha do tamanho do bloco para os índices e sequências (folhas): ajuste entre tamanho do bloco, características do disco e a quantidade de memória disponível
  - o tamanho de bloco comum torna mais fácil implementar um esquema de buffering para criar uma árvore B+ virtual de prefixo simples
  - Os blocos de índices e de sequências são normalmente incorporados no mesmo arquivo → evita o acesso em dois arquivos distintos

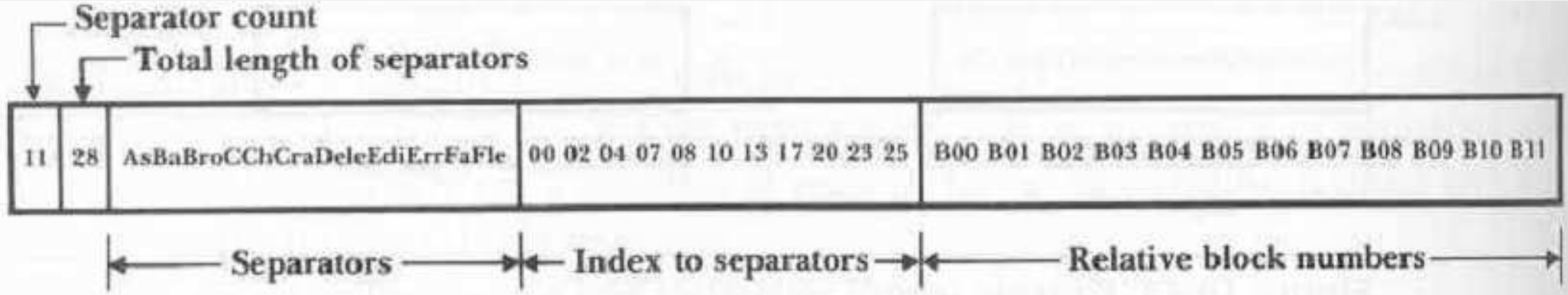
Árvores B de ordem variável

- Como são armazenados os separadores em um bloco?
  - separadores curtos abrem a possibilidade de armazenar um número maior em um nó
    - motivação desaparece nas árvores B de ordem fixa
  - mesmo com o uso de separadores de tamanho variável deve haver suporte para busca binária
- Uso de índices separados, com referências de tamanho fixo → permite o uso de busca binária



Árvores B de ordem variável

- Uma possível solução:
  - Campo contador de separadores: para indicar o elemento do meio na busca binária e o tamanho do campo de índices de separadores
  - Campo com o comprimento total dos separadores: serve para calcular o início do campo dos índices de separadores



Árvores B de ordem variável

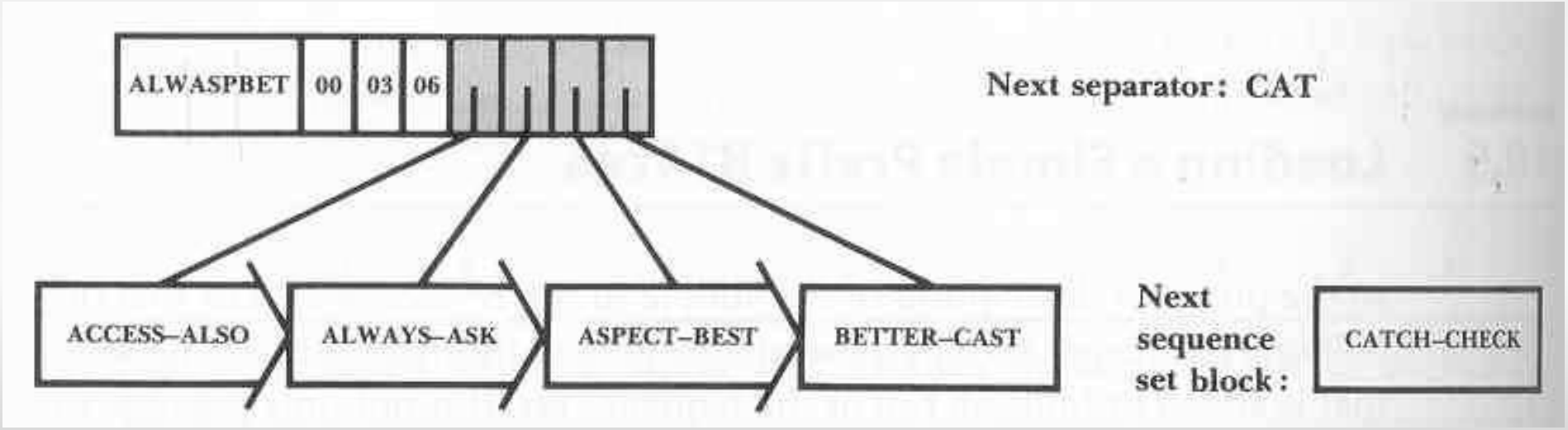
- Implicações do uso de prefixos de tamanho variável:
  - o número de separadores no bloco é limitado pelo tamanho do bloco ao invés de ter uma ordem (número de chaves por nó) pré-determinada
  - como a árvore tem ordem variável, as decisões de quando dividir, intercalar ou distribuir ficam mais complicadas

Construção: árvore B+ de prefixo simples

- Inserções sucessivas (chaves aleatórias ou não)
  - Problema: operações de divisão e redistribuição são relativamente caras
  - Uso: quando não se tem o conjunto de chaves previamente
- Carregamento sequencial (bulk loading)
  - Necessita ordenação do conjunto de chaves
    - colocam-se os registros ordenados nos blocos do conjunto de sequências, um a um, começando um novo bloco quando aquele que está sendo preenchido fica cheio. Em cada transição de bloco, determina-se o separador mais curto para os blocos
    - separadores podem ser coletados em um bloco de conjunto de índices que é construído em memória até que fique cheio

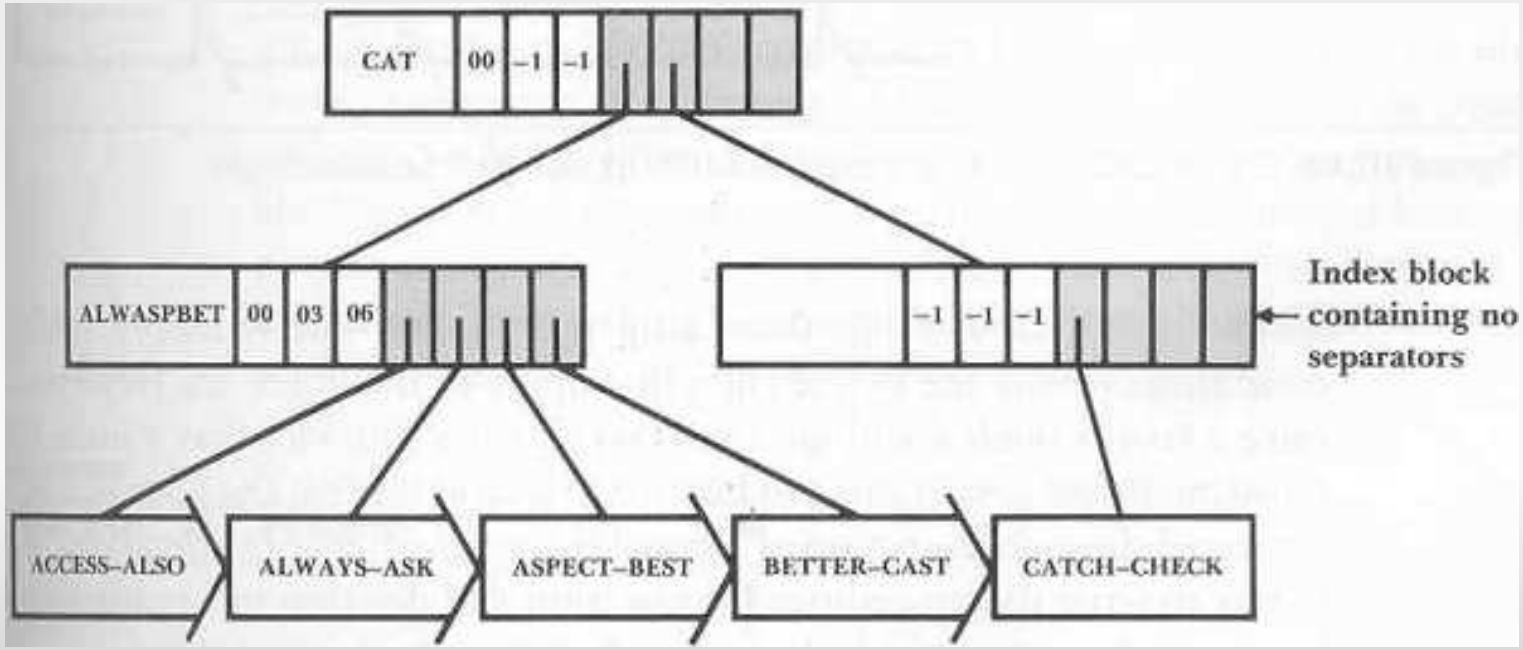
Construção: árvore B+ de prefixo simples

- Exemplo: termos para um índice de livro
  - registros: lista das ocorrências de cada termo
  - 4 blocos do conjunto de sequências foram escritos
  - bloco do conjunto de índices está cheio e deve ser escrito no disco
  - o que fazer com o separador CAT?



Construção: árvore B+ de prefixo simples

- Cria-se um novo bloco do conjunto de índices
  - CAT não pode ser inserido em um índice no mesmo nível que índice ALW, ASP e BET: não pode-se ter dois blocos no mesmo nível sem um bloco pai
  - CAT é promovido para o bloco de mais alto nível
  - Tem-se dois níveis do conjunto de índices na memória à medida que se constrói o conjunto de sequências

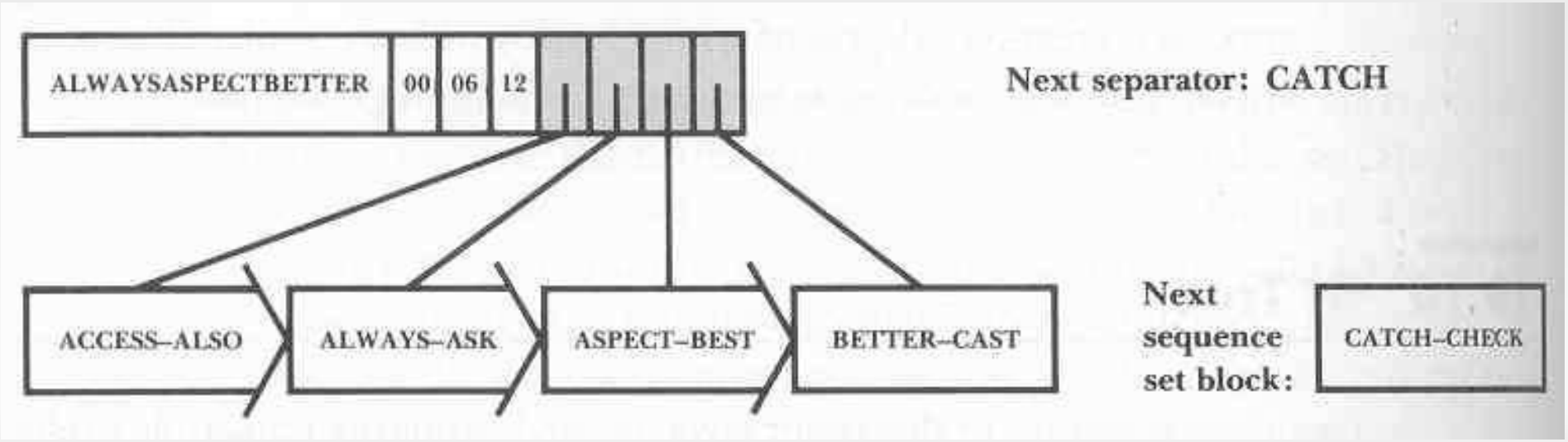


Construção: árvore B+ de prefixo simples

- Carregamento sequencial (bulk loading)
  - vantagens:
    - pode-se determinar a ocupação dos nós
    - produz árvores menos fragmentadas
      - com melhor desempenho
    - pode resultar em árvores com altura menor
    - operação de leitura sequencial, única passagem sobre os dados
      - ao invés das várias passagens associadas com inserções de forma aleatória
    - nenhum bloco precisa ser reorganizado durante o procedimento

Árvore B+ (padrão)

- Não usa prefixos como separadores
- Possui ponteiros entre os nós folha
  - em busca por faixa de valores, após encontrado o primeiro valor, não precisa voltar nos índices



Árvore B+ (padrão)

- Quando usar a árvore B+ padrão para strings
  - Se conjunto de chaves não obtém alta compressão quando o método do prefixo simples é usado
    - nesse caso não compensa overhead para manter prefixos

Conclusões

- Árvores B, B+ e B+ de prefixo simples
  - usar quando não há memória para índice inteiro
  - usar estrutura de gerenciamento de buffers
  - não usar quando há memória disponível para a tarefa. Alternativas
    - árvore binária balanceada tem desempenho melhor em memória que uma árvore B em memória
    - tabela hash

Conclusões

- Características das árv. B, B+ e B+ de prefixo simples
  - São estruturas paginadas
    - lêem blocos inteiros de informação para a memória de cada vez
  - Permitem buscar em um grande volume de chaves com poucos acessos ao disco
  - Mantêm árvores balanceadas na altura
  - As árvores crescem de baixo para cima
  - O balanceamento é mantido por meio da divisão, junção e redistribuição de blocos
  - Podem ser usadas junto com gerenciamento de buffers

Leitura complementar

- Capítulos
  - **”Multilevel Indexing and B-trees”**
  - **”Indexed Sequential File Access and Prefix B+ Trees”**
- do livro Folk et al. ”File Structures: An Object-Oriented Approach with C++”, Editora Pearson, 3ª edição, 1998