



Árvores B – Parte V

Árvores B+

Adaptado e Estendido dos Originais de:

Cristina Dutra de Aguiar Ciferri

Acesso a Arquivos

- Alternativas (até o momento)
 - **acesso indexado**
 - registros localizados por uma chave em um arquivo de índice auxiliar (p. ex. Árvore B)
 - **acesso seqüencial**
 - registros percorridos seqüencialmente ao longo do arquivo lógico. Pode ser de 2 tipos:
 - **ordenado**: se arquivo for mantido ordenado
 - **não ordenado**: caso contrário

Acesso Seqüencial & Indexado

- Em muitas aplicações, é desejável que se possa fazer ambos os acessos indexado e seqüencial ordenado
 - Processamentos co-seqüenciais demandam arquivos ordenados
 - Consultas por chave eficientes demandam arquivos indexados
- Por exemplo:
 - Arquivo da Seção de Graduação
 - Processamento das matrículas (acesso seqüencial ordenado)
 - Consulta a histórico escolar de um aluno pelo No. USP (acesso indexado)
 - Arquivo de Operadora de Cartão de Crédito
 - Processamento das faturas (acesso seqüencial ordenado)
 - Consulta ao status do cartão (acesso indexado)

Questão

- Já sabemos que o custo de manter o arquivo principal ordenado registro a registro em função de uma chave é usualmente inaceitável
- Como então conseguir realizar acesso seqüencial & indexado ???

Foco 1

- Problema
 - manter os registros ordenados logicamente em função da chave, de forma eficiente
- Solução
 - organizar registros em **blocos**

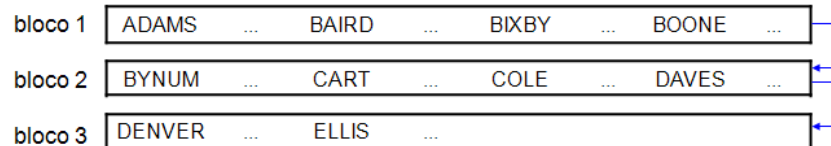
um bloco consiste na unidade básica de L/E e deve ter seu tamanho determinado de forma que leituras e escritas demandem um único acesso

Uso de Blocos

- Características
 - o conteúdo (conjunto de registros) de cada bloco está ordenado e pode ser recuperado em um acesso
 - registros podem ser de tamanho fixo ou variável
 - nesse curso discutiremos apenas o caso de tamanho fixo
 - cada bloco mantém um 'ponteiro' para o bloco antecessor e um 'ponteiro' para o bloco sucessor
 - blocos logicamente adjacentes não estão (necessariamente) fisicamente adjacentes
- Garante acesso seqüencial ao arquivo principal

Exemplo de Uso de Blocos

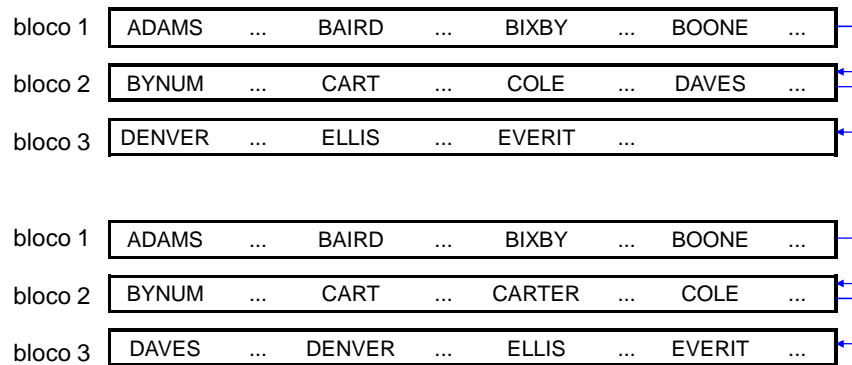
- Por simplicidade apenas as chaves estão sendo mostradas



Problema 1

- **Inserção de Registros:**
 - pode provocar **overflow** em um bloco
- **Solução 1 (redistribuição):**
 - redistribuir registros, movendo-os entre blocos logicamente adjacentes
 - desde que haja espaço disponível

Exemplo: Inserção de CARTER

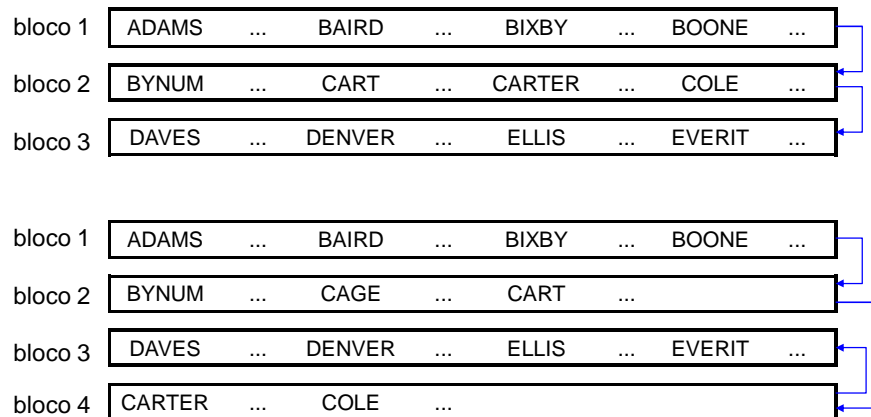


Problema 1

- **Inserção de Registros:**
 - pode provocar **overflow** em um bloco
- **Solução 2 (split):**
 - dividir o bloco
 - processo similar ao realizado em Árvores B
 - passos
 - divide os registros entre os dois blocos
 - rearranja os ponteiros

não existe
promoção !

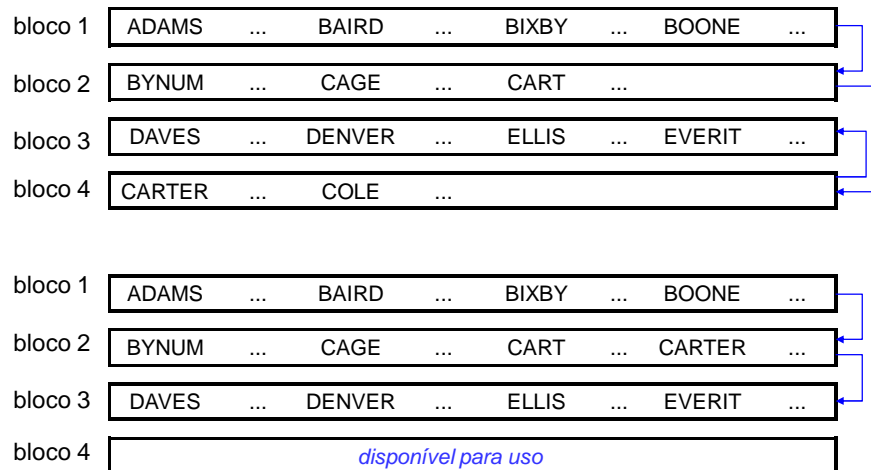
Exemplo: Inserção de CAGE



Problema 2

- **Remoção de Registros:**
 - pode provocar **underflow** em um bloco
- **Solução 1 (concatenação)**
 - concatenar o bloco com o seu antecessor ou sucessor na seqüência lógica
 - desde que um deles esteja com apenas ~metade da sua capacidade máxima

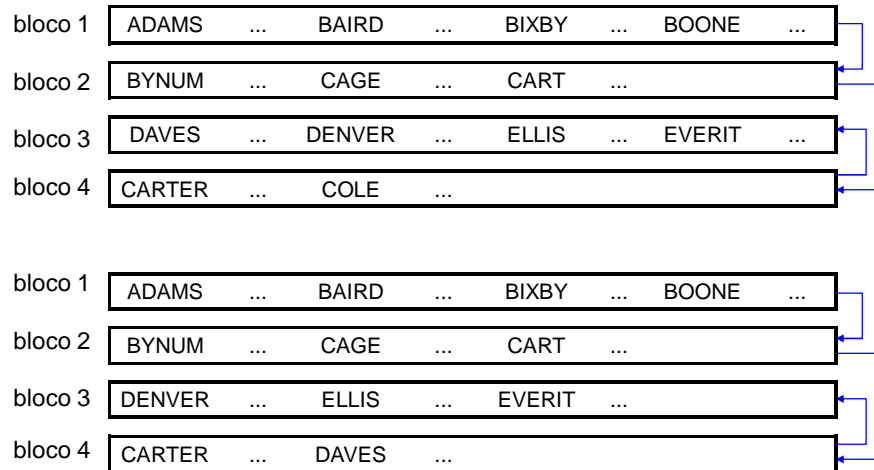
Exemplo: Remoção de COLE



Problema 2

- **Remoção de Registros:**
 - pode provocar **underflow** em um bloco
- **Solução 2 (redistribuição)**
 - redistribuir registros, movendo-os entre blocos logicamente adjacentes
 - desde que haja registros adicionais disponíveis

Exemplo: Remoção de COLE



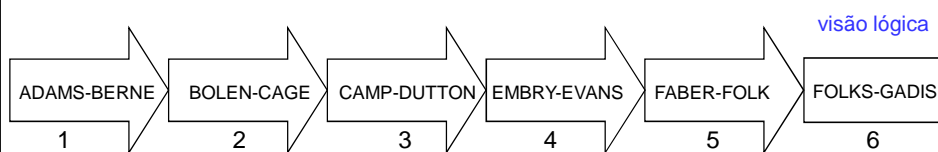
Nota

- Ao contrário do que ocorre nos arquivos de índice em Árvores B, aqui a concatenação pode ser preferida à redistribuição, porque libera espaço no arquivo principal

Foco 2

- Situação
 - os registros podem agora ser acessados seqüencialmente, em ordem (da chave)
- Problema
 - como localizar eficientemente um bloco com um registro particular, dada a sua chave ???
- Soluções
 - índice simples para referenciar os blocos
 - Árvore-B+

Índice Simples (Tabela)



Chave	RRN do Bloco
BERNE	1
CAGE	2
DUTTON	3
EVANS	4
FOLK	5
GADIS	6

Índice Linear:

- registros de tamanho fixo
- contêm chave para o último registro do bloco (chave delimitadora)

Índice Simples

- **Exercício:**

- Apresente o índice para cada um dos exemplos vistos anteriormente, antes e depois de cada inserção ou remoção

Acesso Seqüencial & Indexado Linearmente

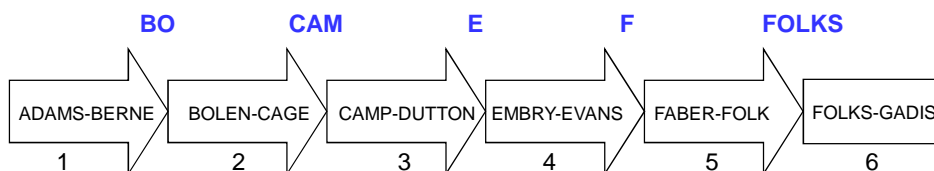
- Combina
 - registros ordenados pela chave, em blocos
 - índice linear simples para indexar os blocos
- Restrição
 - acesso indexado eficiente presume que o índice cabe todo em memória primária
 - busca binária no índice em RAM
 - atualização do índice em RAM

Observação

- Cada chave delimitadora no índice não possui o papel tradicional de chave que indexa um registro específico
 - mas apenas o papel de **separação** de blocos
- Logo, pode-se utilizar separadores mais simples ao invés das chaves em si
 - separadores mínimos

Separadores Mínimos

- No. mínimo de caracteres que diferencia entre as chaves de dois blocos adjacentes
- Exemplo:



Separadores Mínimos

- Tabela de decisão:

chave de busca x separador	decisão
chave < separador	procure à esquerda
chave = separador	procure à direita
chave > separador	procure à direita

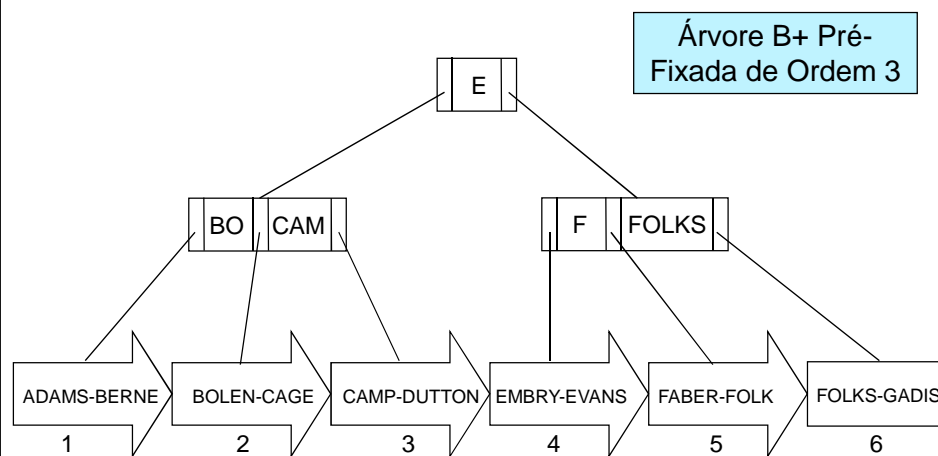
Separadores Mínimos

- No caso de índices simples o uso de separadores mínimos ao invés das chaves não traz benefícios
- Esses separadores, no entanto, podem beneficiar índices paginados que operam com quantidades variáveis de itens por páginas
 - separadores menores \Rightarrow mais separadores nas páginas
- Árvores B⁺ são um exemplo desse tipo de índice
 - \uparrow no. de separadores nas páginas \Rightarrow \downarrow altura da árvore
- Apesar disso, para discutir os fundamentos dessas árvores, consideraremos no. fixo de itens por página


Árvore B+ Pré-Fixada

- Estrutura Híbrida:
 - arquivo principal
 - blocos de registros ordenados por chave
 - separadores
 - organizados como Árvore-B

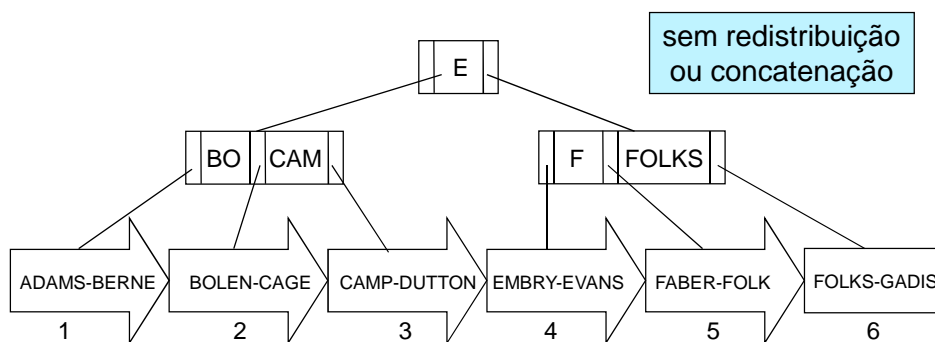
Árvore B+ Pré-Fixada



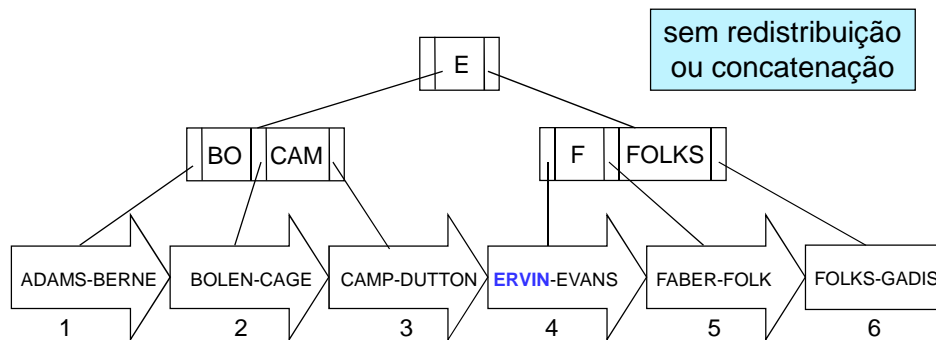
Manutenção

- Cenários
 - inserção
 - remoção
 - *overflow*
 - *underflow*
- 
- Efeitos colaterais
 - arquivo principal
 - árvore-B+

Remoção de EMBRY

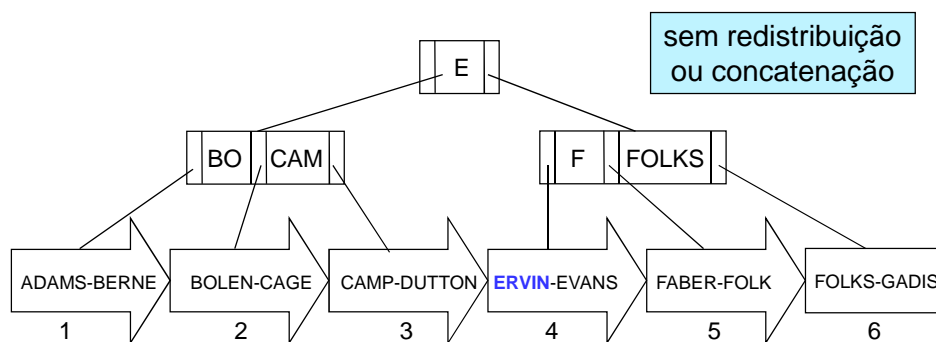


Remoção de EMBRY



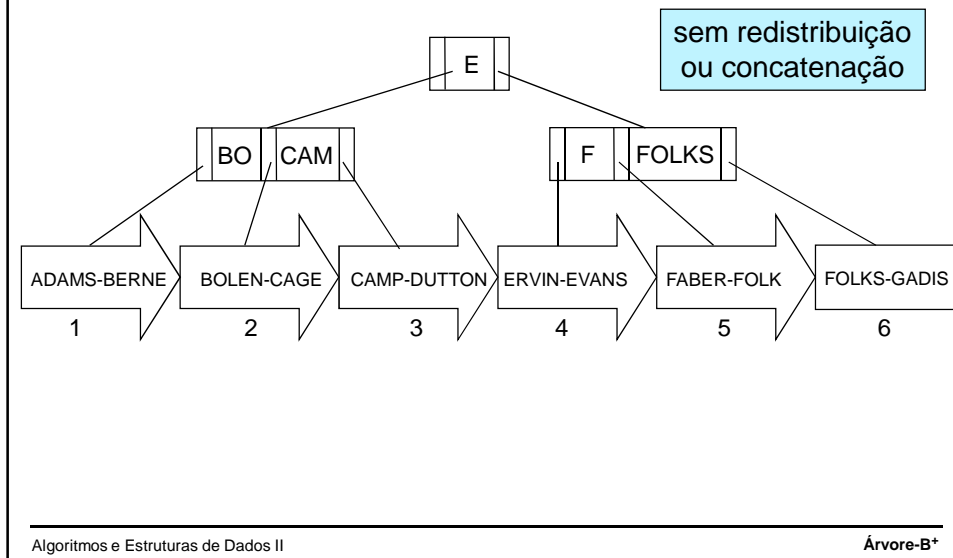
- Efeito no Arquivo Principal
 - limitado a alterações internas ao bloco 4

Remoção de EMBRY

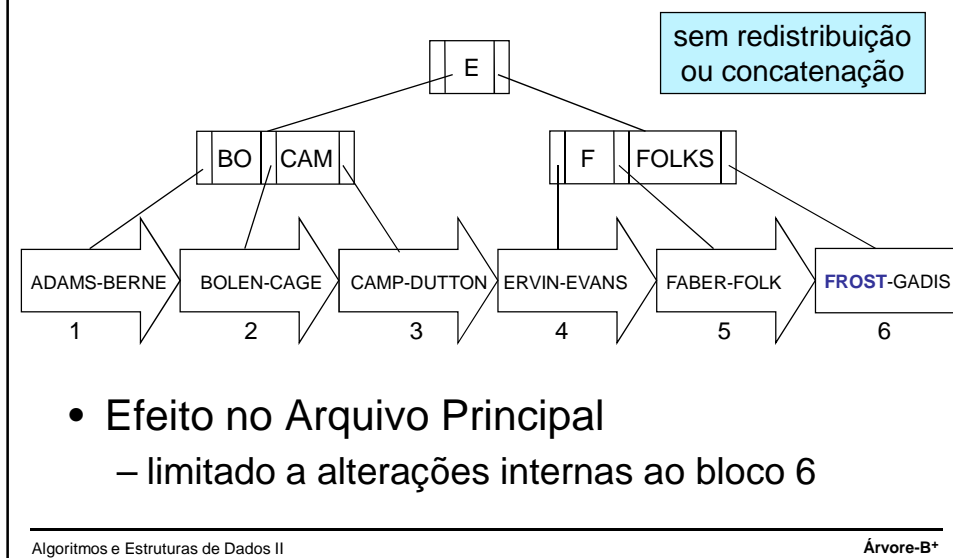


- Efeito na Árvore B+
 - nenhum: “E” permanece um bom separador

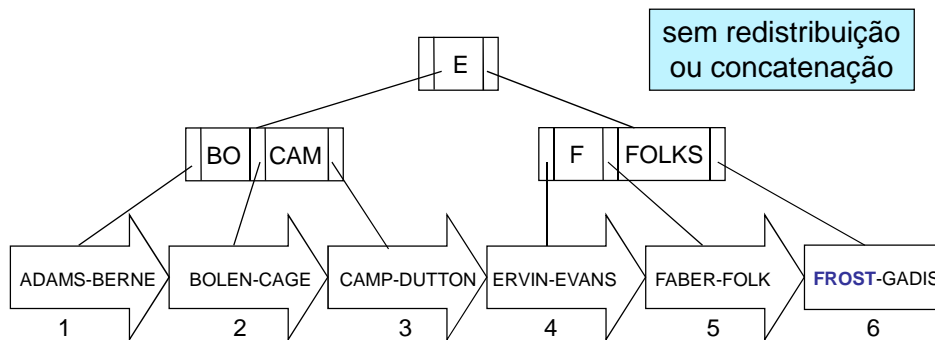
Remoção de FOLKS



Remoção de FOLKS

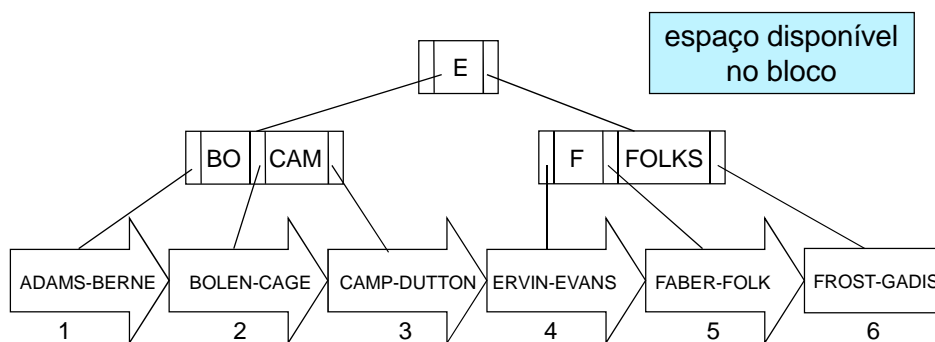


Remoção de FOLKS

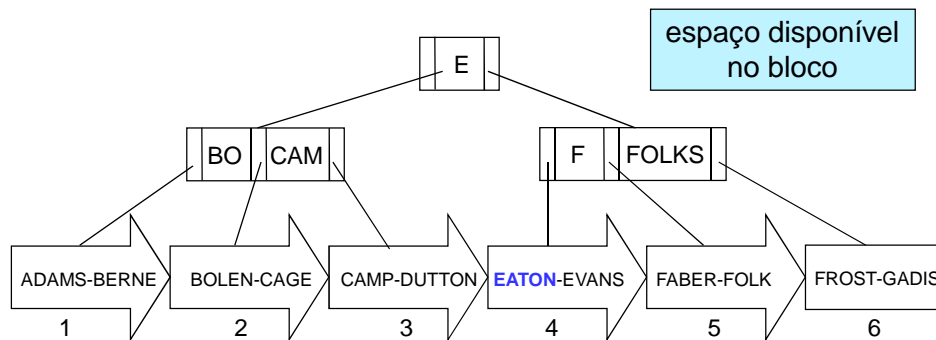


- Efeito na Árvore B+
 - nenhum: “FOLKS” permanece um bom separador

Inserção de EATON

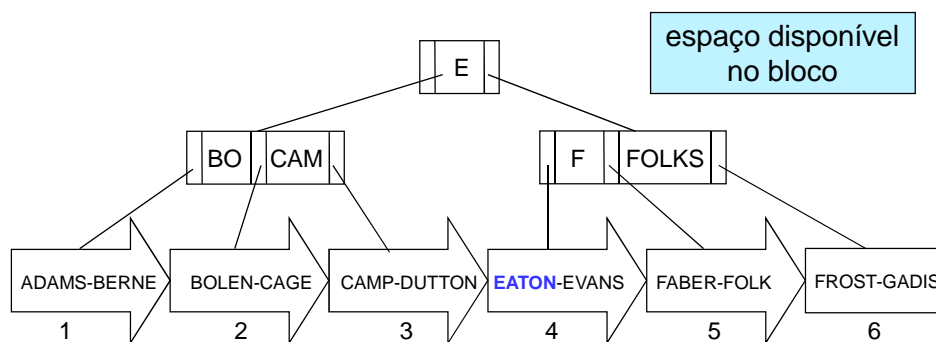


Inserção de EATON



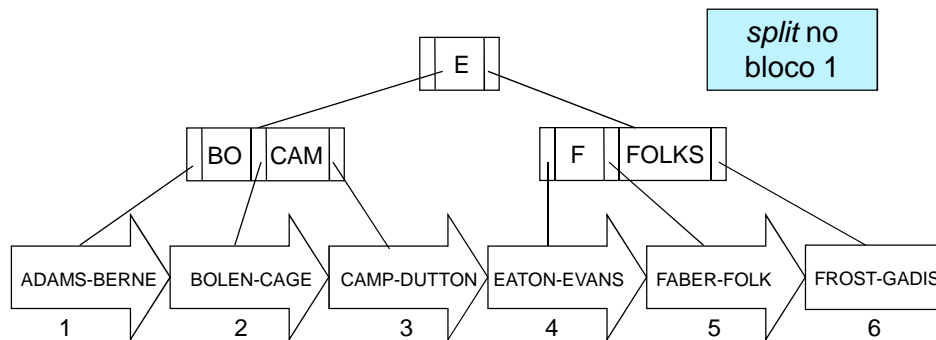
- Efeito no Arquivo Principal
 - limitado a alterações internas ao bloco 4

Inserção de EATON



- Efeito na Árvore B+
 - nenhum: “E” permanece um bom separador

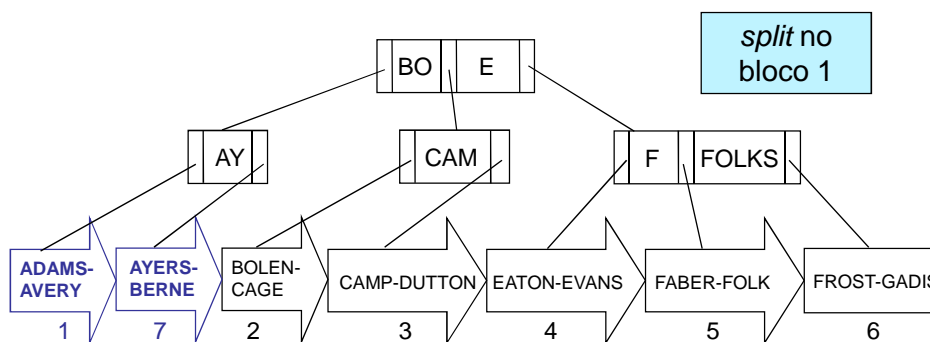
Inserção de AVERY



Algoritmos e Estruturas de Dados II

Árvore-B⁺

Inserção de AVERY

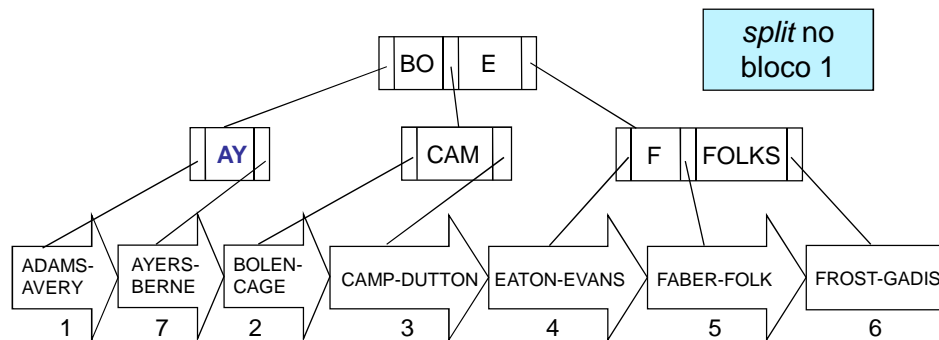


- Efeito no Arquivo Principal
 - Registros do Bloco 1 + AVERY são distribuídos entre os blocos 1 e 7

Algoritmos e Estruturas de Dados II

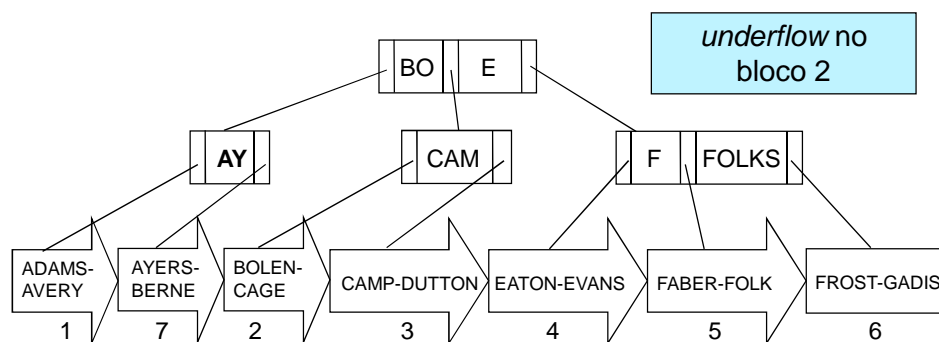
Árvore-B⁺

Inserção de AVERY

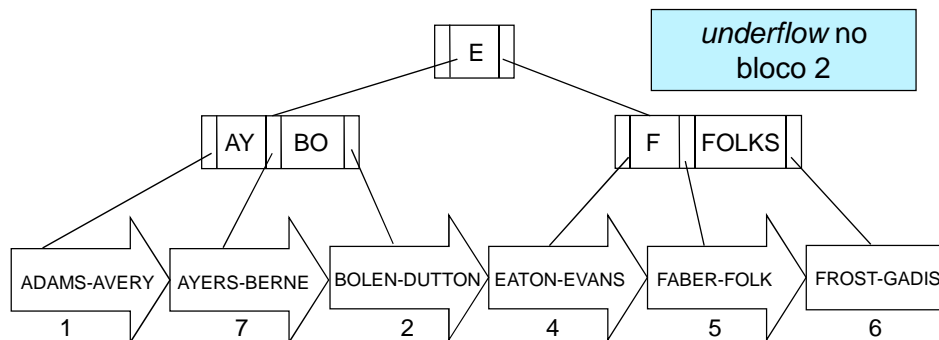


- Efeito na Árvore-B+
 - separador adicional AY inserido na Árvore

Remoção de CAEL

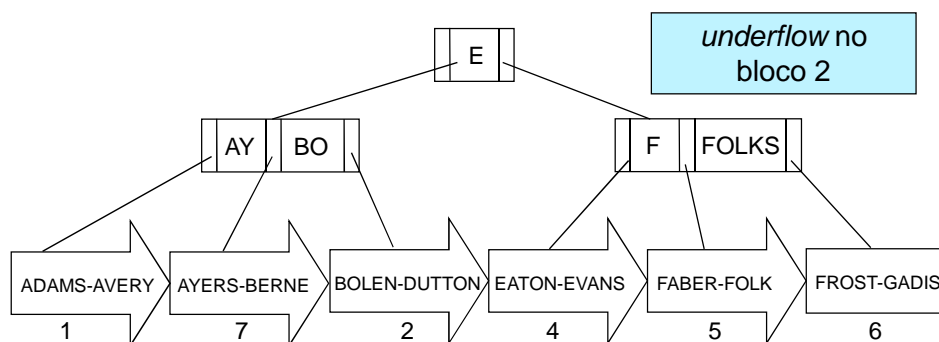


Remoção de CAEL



- Efeito no Arquivo Principal
 - concatenação dos blocos 2 e 3

Remoção de CAEL



- Efeito na Árvore B+
 - remoção de CAM e concatenação

Inserção e Remoção

- Primeiro Passo: *Arquivo Principal*
 - inserir ou remover o registro
 - tratar, caso necessário
 - *split*
 - concatenação
 - redistribuição

Inserção e Remoção

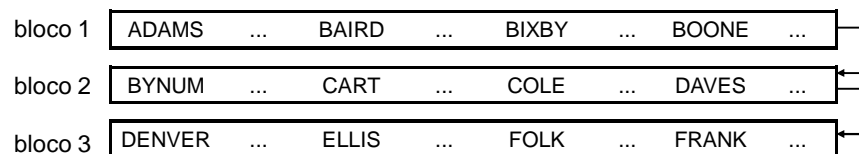
- Segundo Passo: *Árvore B⁺*
 - se *split* no arquivo principal
inserir um novo separador no índice
 - se *concatenação* no arquivo principal
remover um separador do índice
 - se *redistribuição* no arquivo principal
alterar o valor do separador no índice

Exercícios

- Árvore B+
 - Ordem $m = 3$
- Blocos do Arquivo Principal
 - número máximo de registros: 4
 - número mínimo de registros: 2
 - *underflow*: 1 registro

Exercícios

1. Quais os separadores dos blocos ?



2. Construa a Árvore B+ correspondente

3. Realize as seguintes operações

- inserção de CARTER
- inserção de DRAG
- remoção de BIXBY
- remoção de COLE



Bibliografia

- **M. J. Folk and B. Zoellick, *File Structures: A Conceptual Toolkit*, Addison Wesley, 1987.**