Árvore-B*

Profa. Dra. Cristina Dutra de Aguiar Ciferri

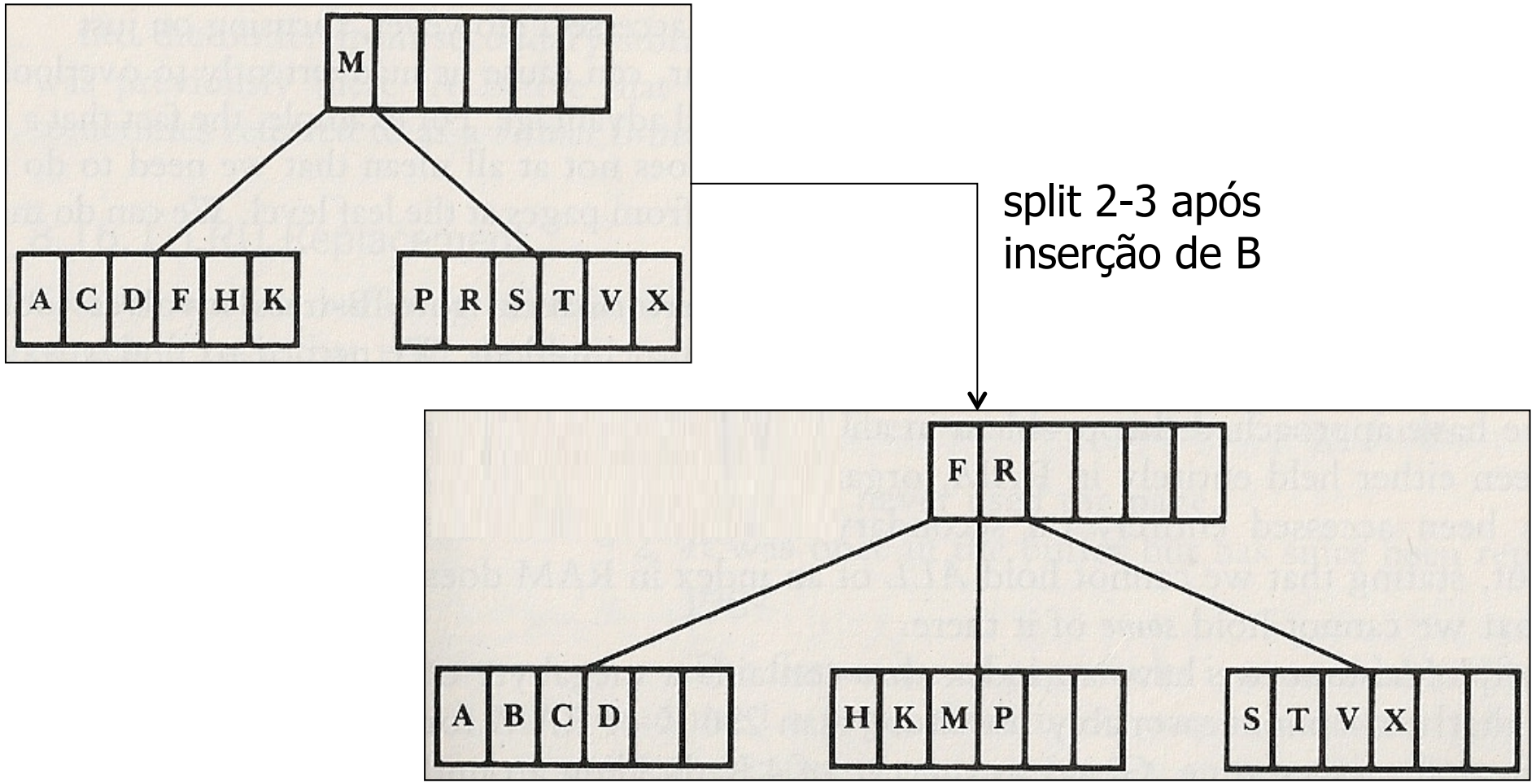
Árvore-B*

- Proposta por Knuth em 1973
  - variação de árvore-B
- Característica
  - cada nó contém, no mínimo, 2/3 do número máximo de chaves
- Posterga o *split*
  - estende a noção de redistribuição durante a inserção para incluir novas regras para o particionamento de nós

Árvore-B*

- Geração
  - utiliza uma variação do processo de subdivisão
- Características
  - a subdivisão é adiada até que duas páginas irmãs estejam cheias
  - na sequência, a divisão do conteúdo das duas páginas em três páginas (*two-to-three split*) é realizada

Split 2-to-3: Exemplo



Definição Formal

- Propriedades de uma Árvore-B*
  - cada página possui um máximo de m descendentes
  - cada página, exceto a raiz e as folhas, possui no mínimo (2m-1)/3 descendentes → taxa de ocupação
  - a raiz possui pelo menos 2 descendentes, a menos que seja um nó folha
  - todas as folhas aparecem no mesmo nível
  - uma página interna com k descendentes contém k-1 chaves
  - uma folha possui no mínimo $\lfloor (2m-1)/3 \rfloor$ chaves e no máximo m - 1 chaves → taxa de ocupação

Observações

- **Mudança na taxa de ocupação**
  - afeta as rotinas de remoção e redistribuição
- Particionamento da raiz
  - problema
    - raiz não possui nó irmão
  - soluções
    - dividir a raiz usando a divisão convencional (*1-to-2 split*); ou
    - permitir que a raiz seja maior

Árvore-B Virtual

Profa. Dra. Cristina Dutra de Aguiar Ciferri

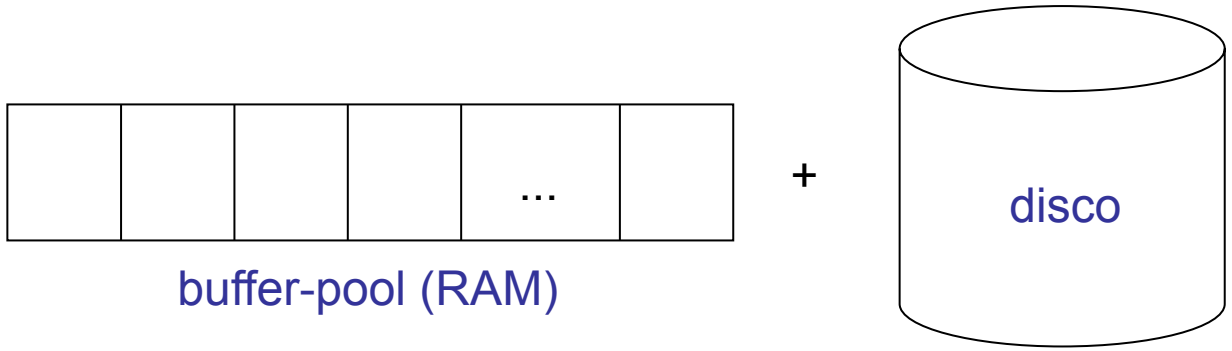
Acessos a Disco

- Problema
  - encontrar uma maneira de fazer um uso eficiente de índices que são muito grandes para serem armazenados inteiramente em memória principal (i.e., RAM)
- Objetivo
  - encontrar uma maneira de diminuir o número médio de acessos a disco para pesquisa

Melhorias de Desempenho

- Manter a página raiz em memória principal
  - ainda deixa espaço disponível em RAM
  - diminui o número de acessos a disco em 1 no pior caso
- Manter em um *buffer-pool* (i.e., em RAM) um certo número de páginas da árvore-B
  - abordagem mais genérica

Árvore-B Virtual



- Pesquisa
  - primeiro procura a página no *buffer-pool* para evitar acessos a disco
  - se a página não estiver no *buffer-pool*, o acesso é realizado em disco e a página é copiada para o *buffer-pool*

Substituição de Páginas

- ***Page Fault***
  - processo de acessar o disco para trazer uma página que não está no *buffer-pool*
  - causas
    - a página nunca foi utilizada
    - a página foi *substituída* no buffer-pool por outra página
- Decisão crítica
  - qual página deve ser substituída no *buffer-pool*, quando este encontra-se cheio?

Opções

- Política LRU (*least recently used*)
  - substitui a página que foi acessada menos recentemente
- Substituição baseada na altura da página
  - mantém as páginas que estão nos níveis mais altos da árvore (i.e., próximas à raiz)
  - utiliza a política LRU para as demais páginas (i.e., páginas mais utilizadas)

Conclusão

- Buferização deve ser incluída em qualquer situação real de utilização de árvore-B
  - para um *buffer-pool* de um certo tamanho, é possível reduzir o número médio de acessos a disco para menos do que 1 acesso
- Pesquisa de Webster (1980)
  - *buffer-pool* com capacidade de 10 páginas
    - somente LRU: 1,42 acessos a disco
    - LRU + altura da página: 1,12 acessos a disco

Árvore-B^+

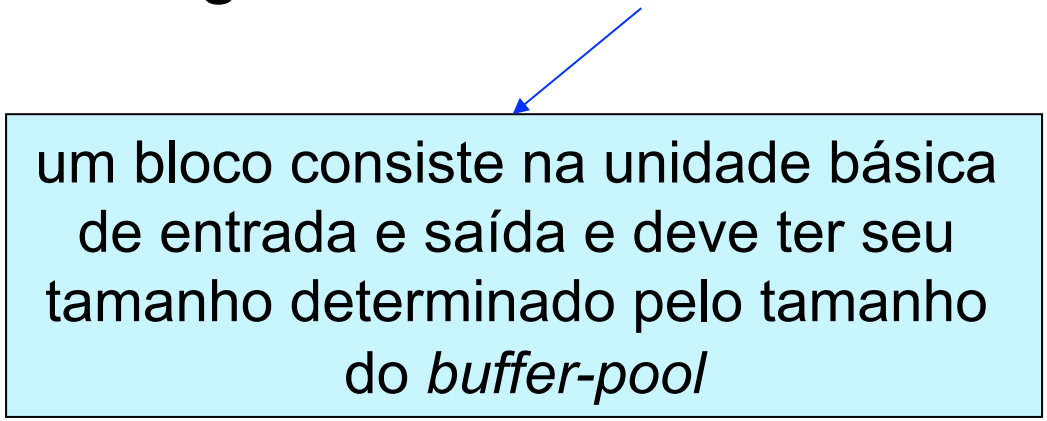
Profa. Dra. Cristina Dutra de Aguiar Ciferri

Acesso Sequencial Indexado

- Alternativas (até o momento)
  - acesso indexado
    - o arquivo pode ser visto como um conjunto de registros que são indexados por uma chave
  - acesso sequencial
    - o arquivo pode ser acessado sequencialmente (i.e., registros fisicamente contínuos)
- Ideia
  - arquivos devem permitir acesso indexado eficiente, e também acesso sequencial

Organização dos Registros

- Problema
  - manter os registros ordenados fisicamente pela chave (*sequence set*)
- Solução
  - organizar registros em blocos



Uso de Blocos

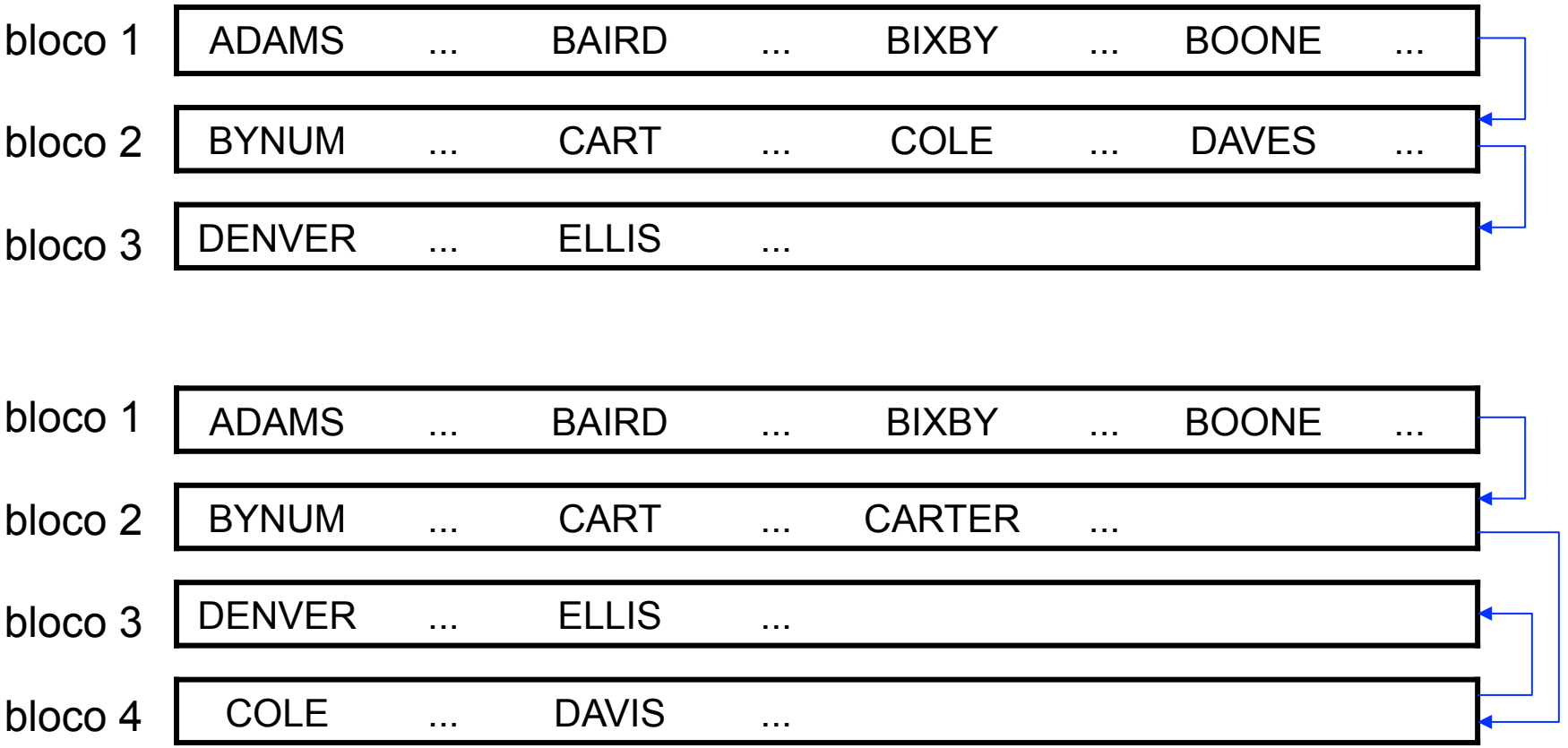
- Características
  - o conteúdo de cada bloco está ordenado, e pode ser recuperado em um acesso
  - cada bloco mantém um ‘ponteiro’ para o bloco antecessor e um ‘ponteiro’ para o bloco sucessor
  - blocos logicamente adjacentes não estão (necessariamente) fisicamente adjacentes
- Garante acesso sequencial ao arquivo

Problema 1

- Inserção de registros pode provocar *overflow* em um bloco
- Solução
  - dividir o bloco, em um processo análogo ao realizado em árvores-B
  - passos
    - divide os registros entre os dois blocos
    - rearranja os ponteiros

não existe promoção !

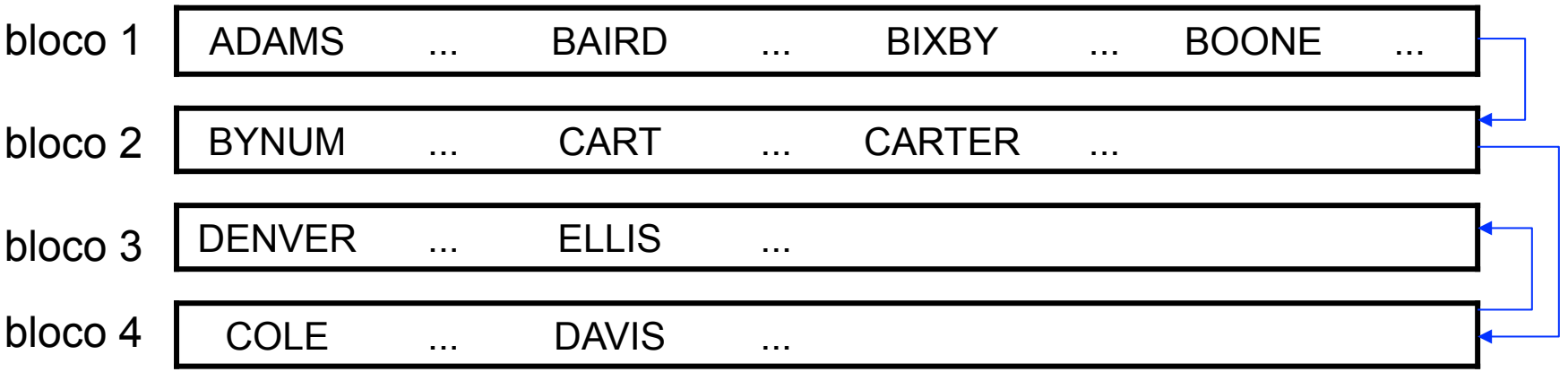
Exemplo: Inserção de CARTER

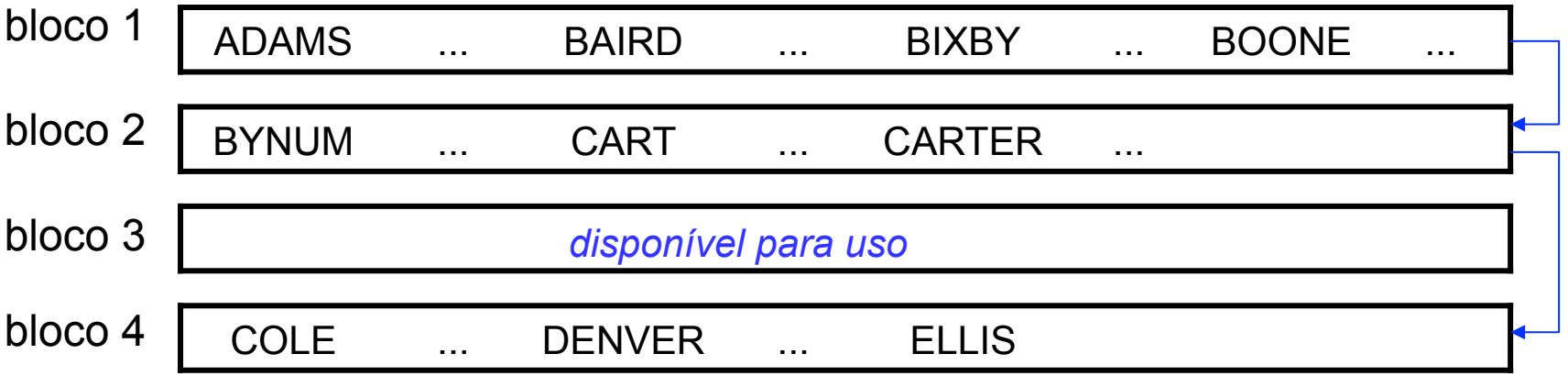


Problema 2

- Remoção de registros pode provocar *underflow* em um bloco
- Solução
  - concatenar o bloco com o seu antecessor ou sucessor na sequência lógica
  - redistribuir os registros, movendo-os entre blocos logicamente adjacentes

Exemplo: Remoção de DAVIS





Uso de Blocos

- Custos associados
  - devido à fragmentação gerada pelas inserções, o arquivo pode ocupar mais espaço do que um arquivo ordenado comum
    - melhorias incluem redistribuição antes do particionamento, *split* 2-to-3, etc
  - a ordem física dos registros não é necessariamente sequencial ao longo do arquivo

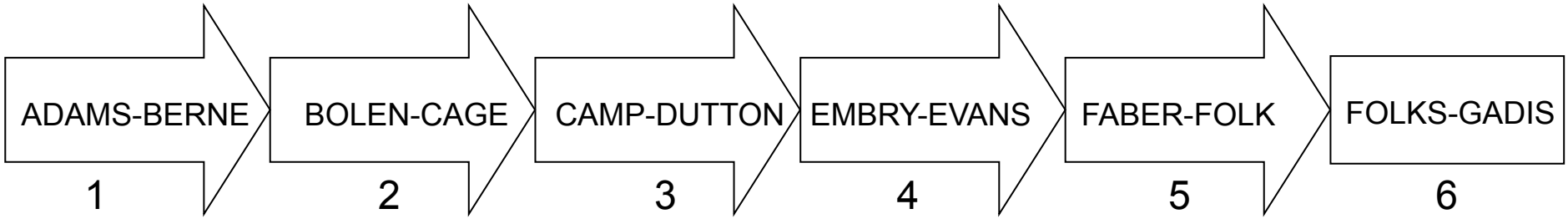
Tamanho do Bloco

- Consideração 1
  - deve permitir que diversos blocos possam ser armazenados em RAM ao mesmo tempo
- Consideração 2
  - deve permitir que um bloco possa ser acessado sem se pagar o custo de um *seek* com a operação de leitura ou escrita do bloco
    - a leitura ou a escrita de um bloco não deve consumir muito tempo

Acesso aos Registros

- Característica
  - os registros podem ser acessados em ordem, sequencialmente, pela chave
- Problema
  - localizar eficientemente um bloco com um registro particular, dado a chave do registro
- Soluções
  - índice simples para referenciar os blocos
  - árvore-B+

Índice Simples (Tabela)



visão lógica

chave	bloco
BERGE	1
CAGE	2
DUTTON	3
EVANS	4
FOLK	5
GADIS	6

Índice de 1 nível

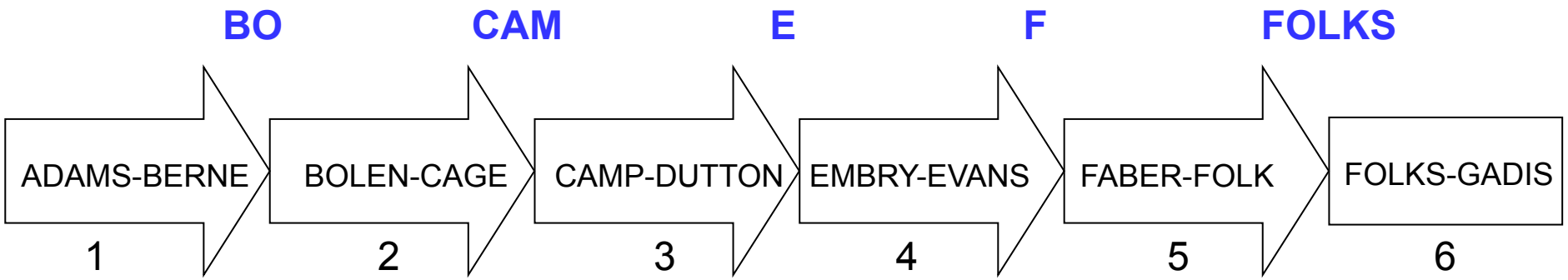
- registros de tamanho fixo
- contém a chave para o último registro no bloco

Acesso Sequencial Indexado

- Combina
  - registros ordenados fisicamente pela chave (*sequence set*)
  - índice simples para referenciar os blocos
- Restrição
  - a organização em tabela implica que o índice cabe na memória principal
    - busca binária no índice
    - atualização do índice em RAM

Separadores

- Características
  - são mantidos no índice, ao invés das chaves de busca
  - possuem tamanho variável
- Exemplo



Separadores

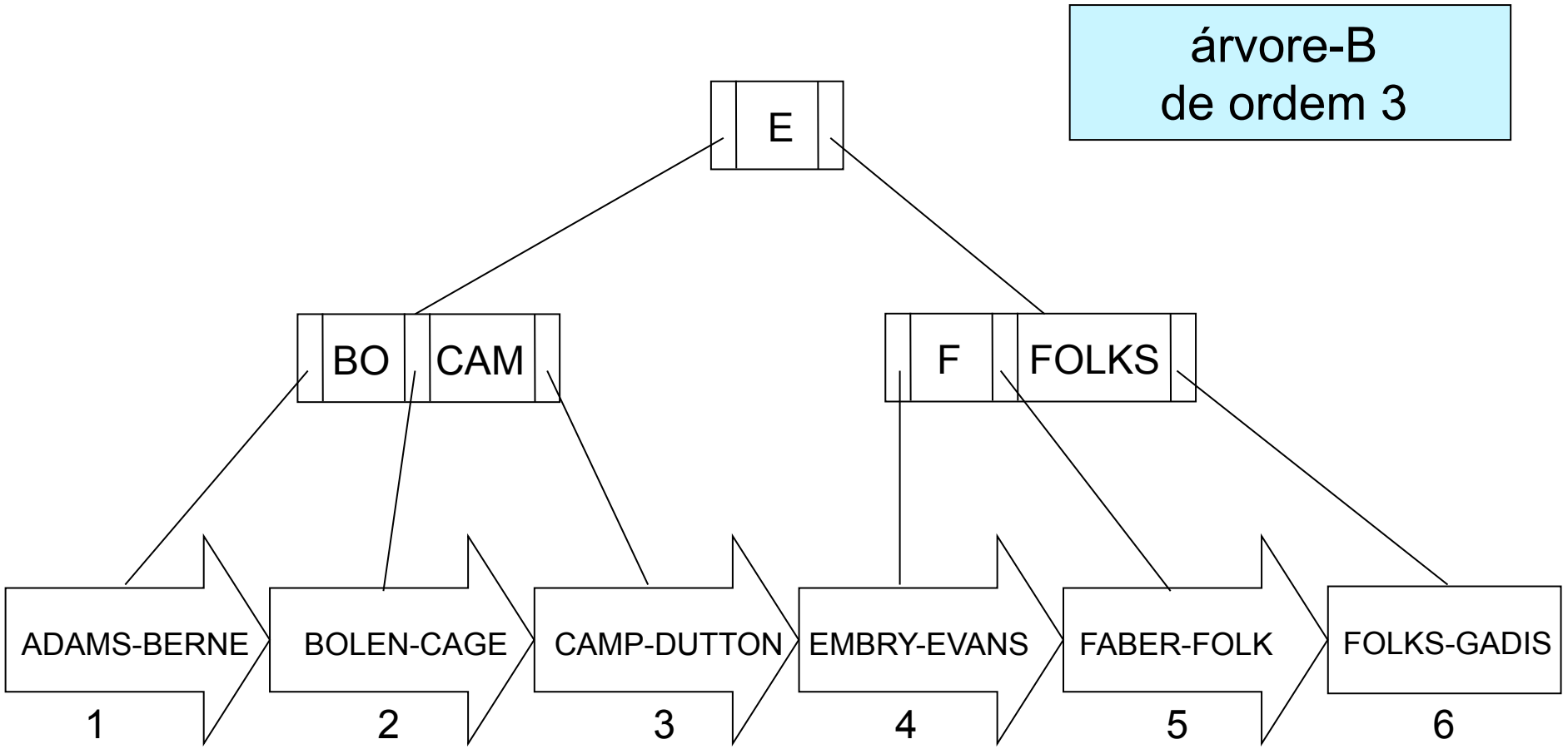
- Desafio
  - escolher o menor separador para utilizar no índice
- Tabela de decisão

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

Árvore-B+ Pré-Fixada

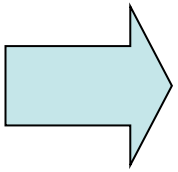
- Estrutura híbrida
  - chaves
    - organizadas como árvore-B
  - nós folhas
    - consistem em blocos de *sequence set*
- Pré-fixada simples
  - armazena na árvore as cadeias separadoras mínimas entre cada par de blocos

Árvore-B+ Pré-Fixada



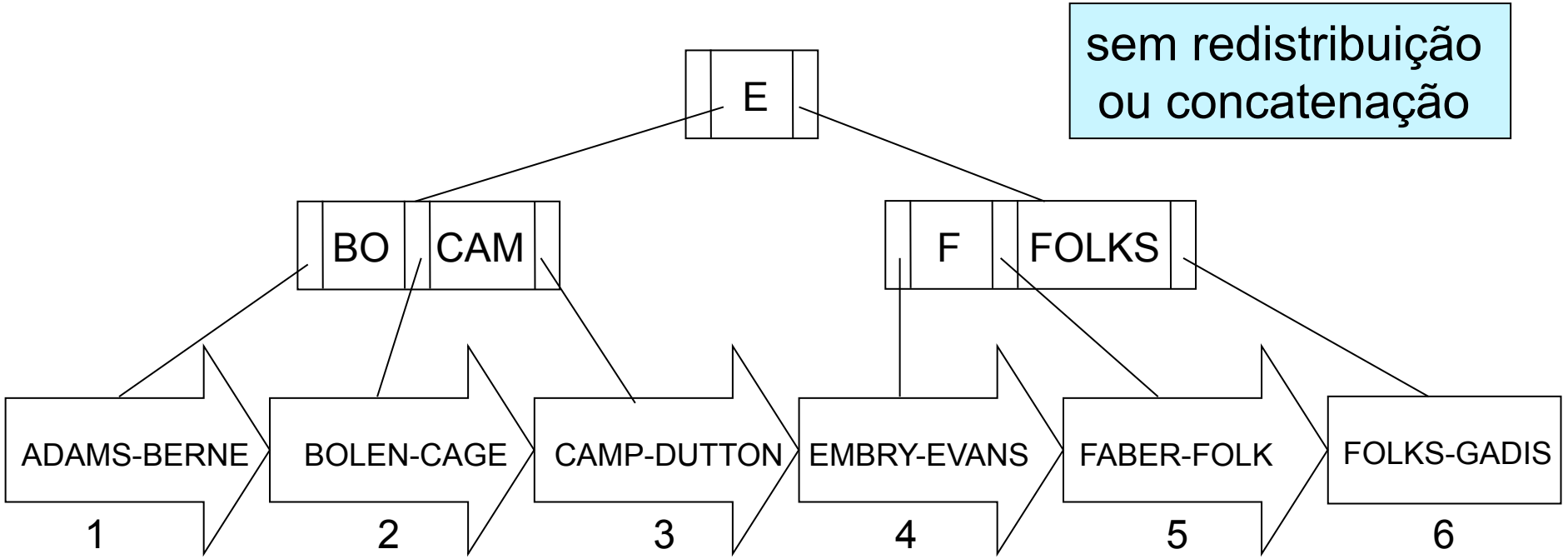
Manutenção

- Cenários
  - inserção
  - remoção
  - *overflow*
  - *underflow*

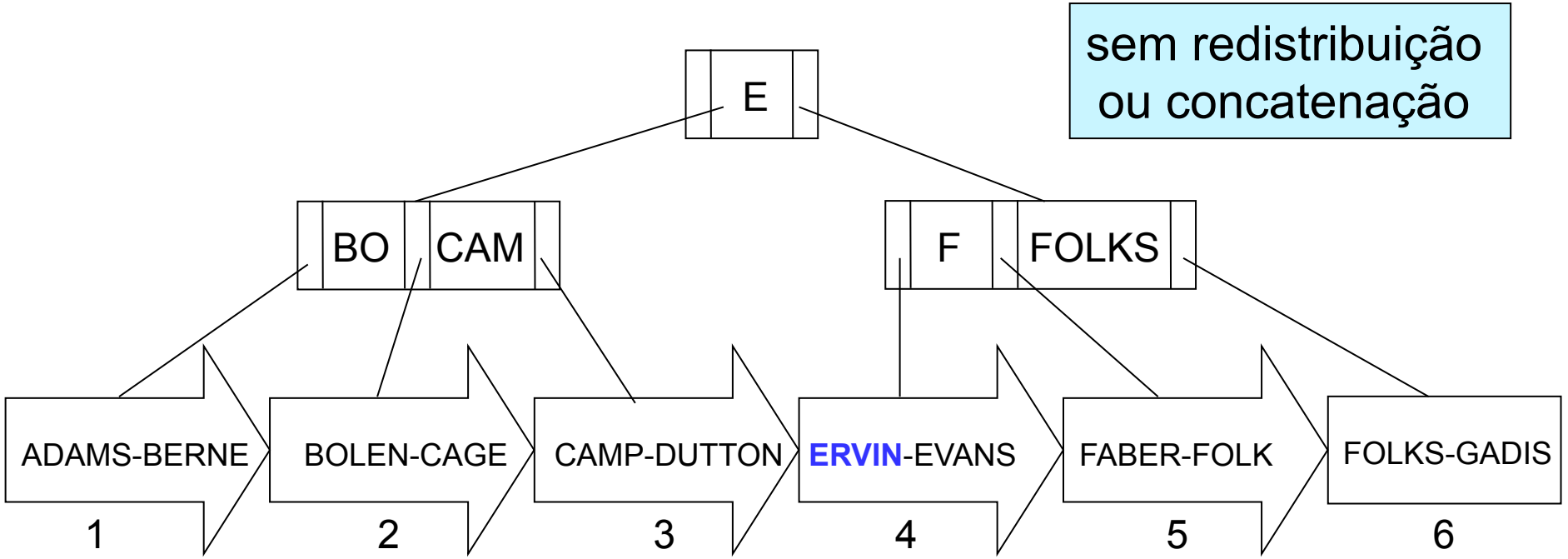


- Efeitos colaterais
  - *sequence set*
  - árvore-B+

Remoção de EMBRY

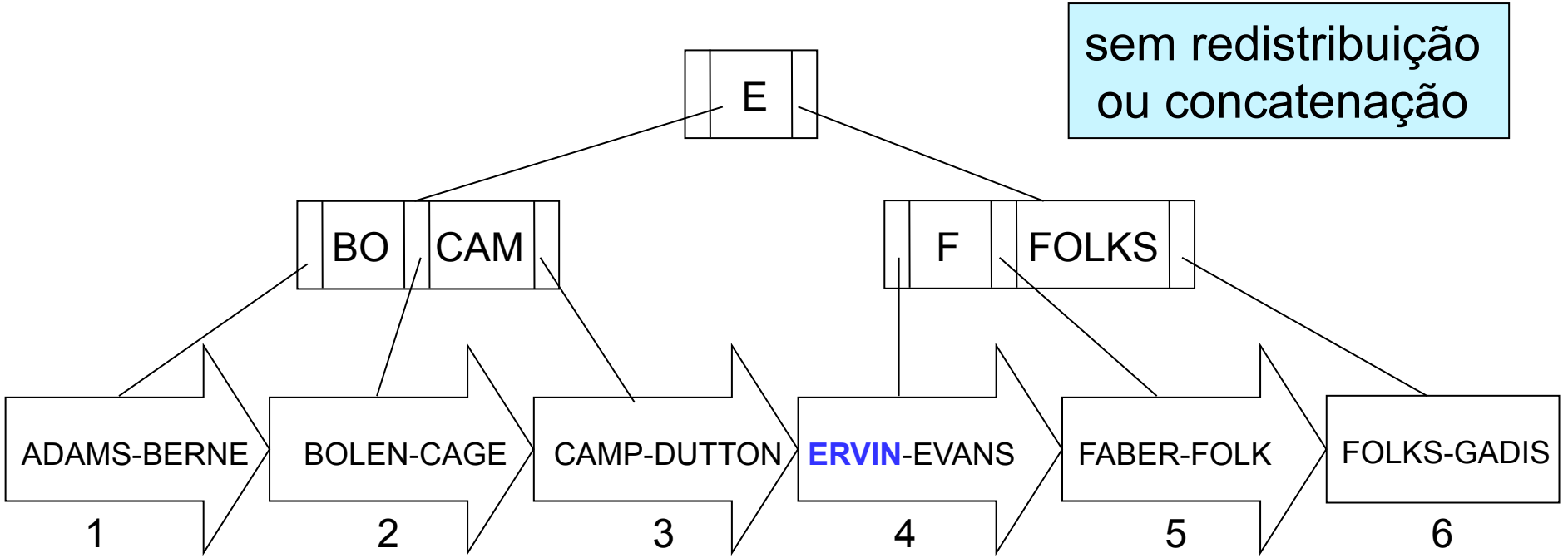


Remoção de EMBRY



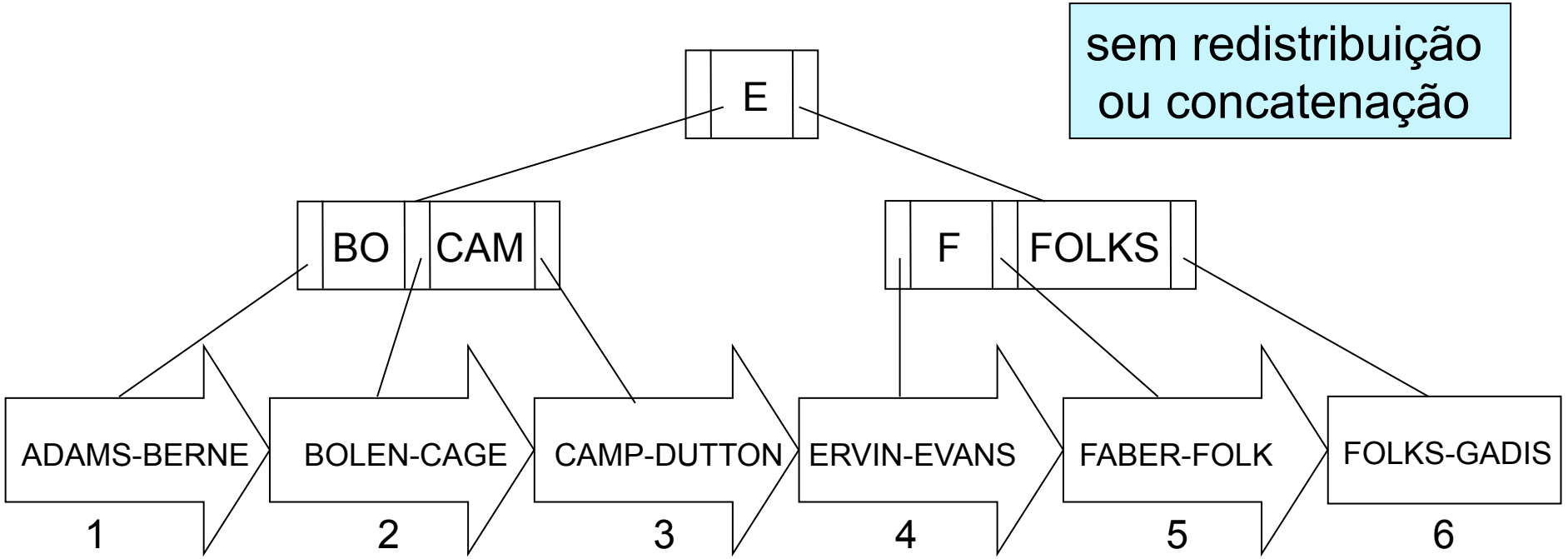
- Efeito no *sequence set*
  - limitado a alterações no bloco 4

Remoção de EMBRY

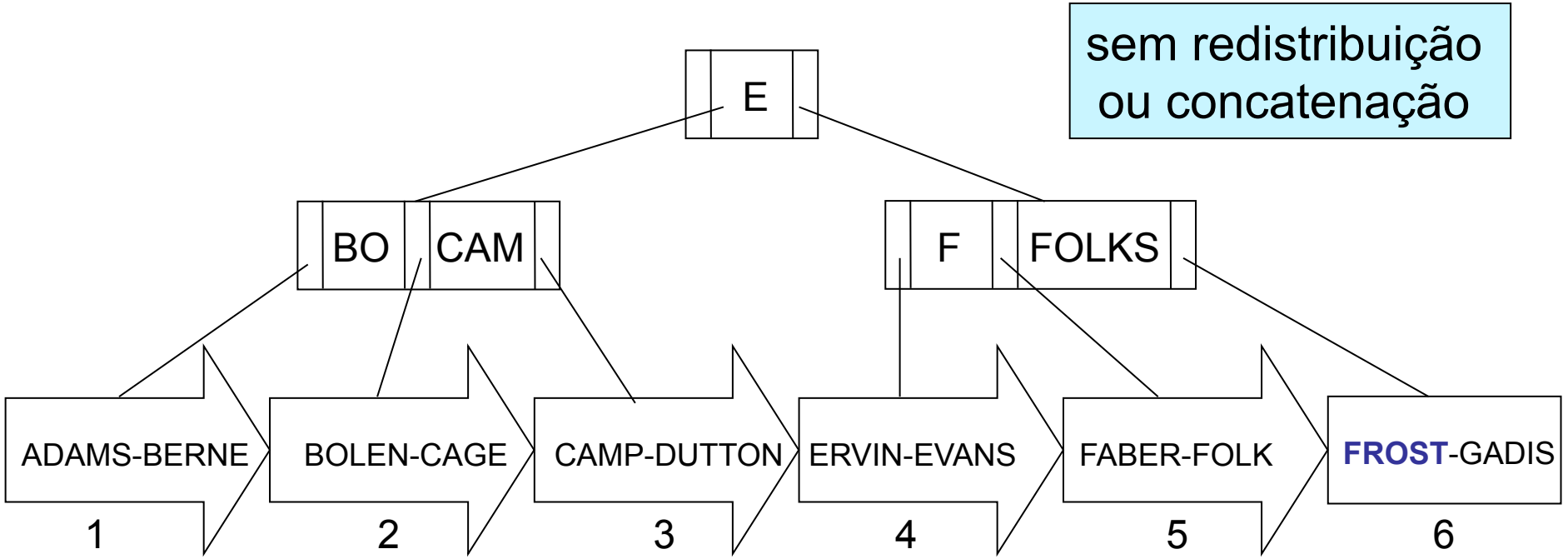


- Efeito na árvore-B+
  - nenhum: E é uma boa chave separadora

Remoção de FOLKS

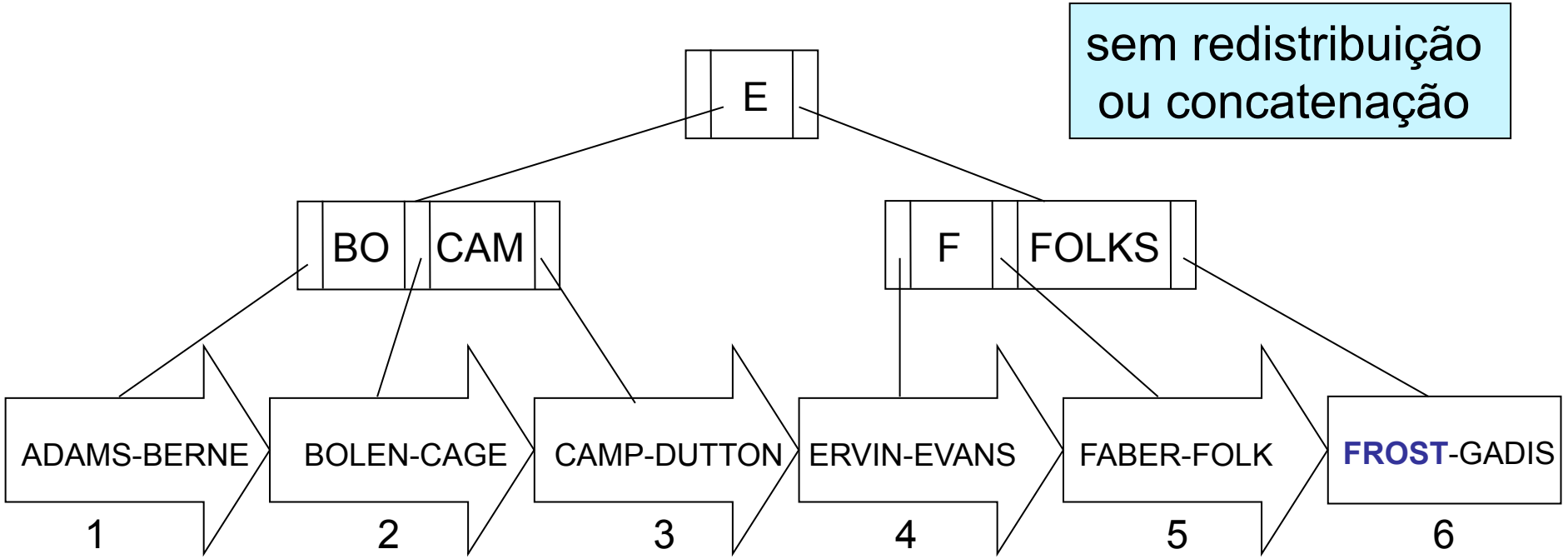


Remoção de FOLKS



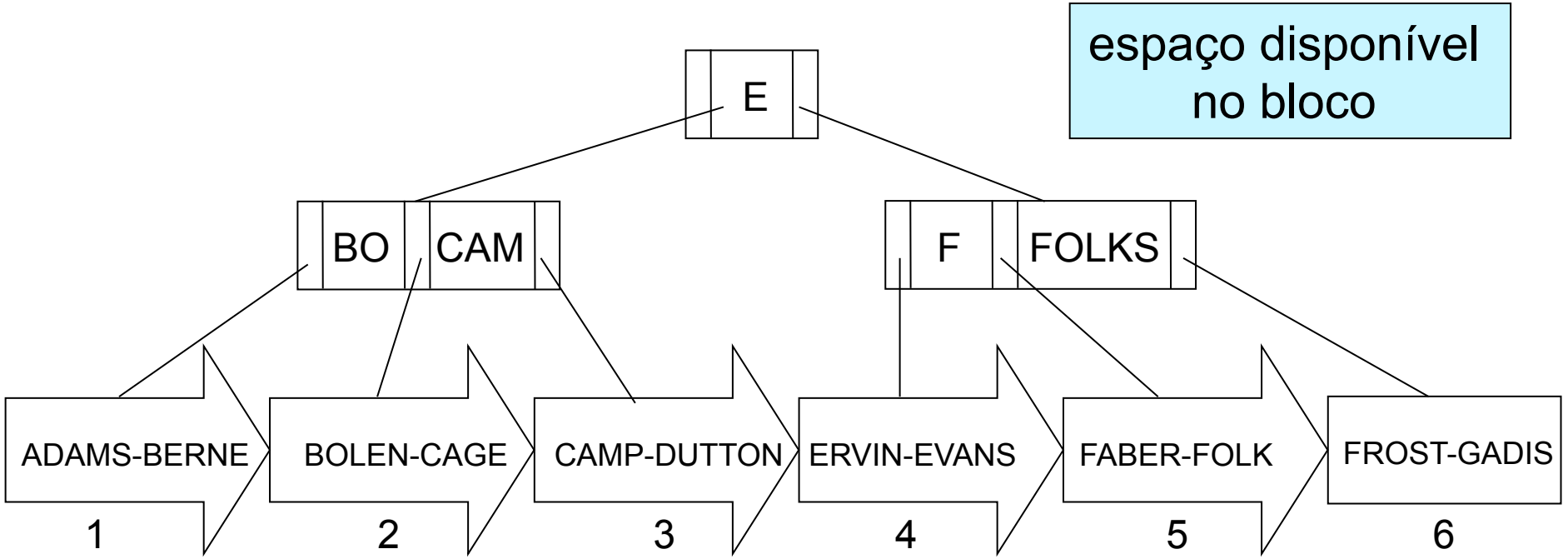
- Efeito no *sequence set*
  - limitado a alterações no bloco 6

Remoção de FOLKS

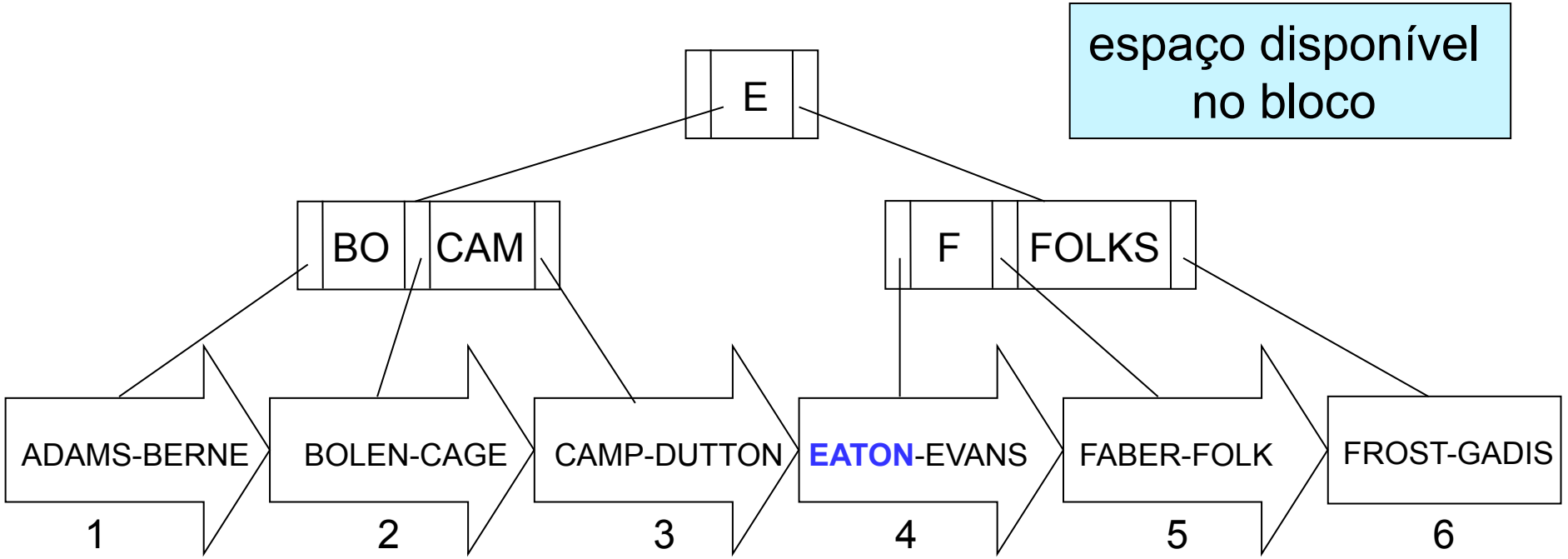


- Efeito na árvore-B+
  - nenhum: custos elevados

Inserção de EATON

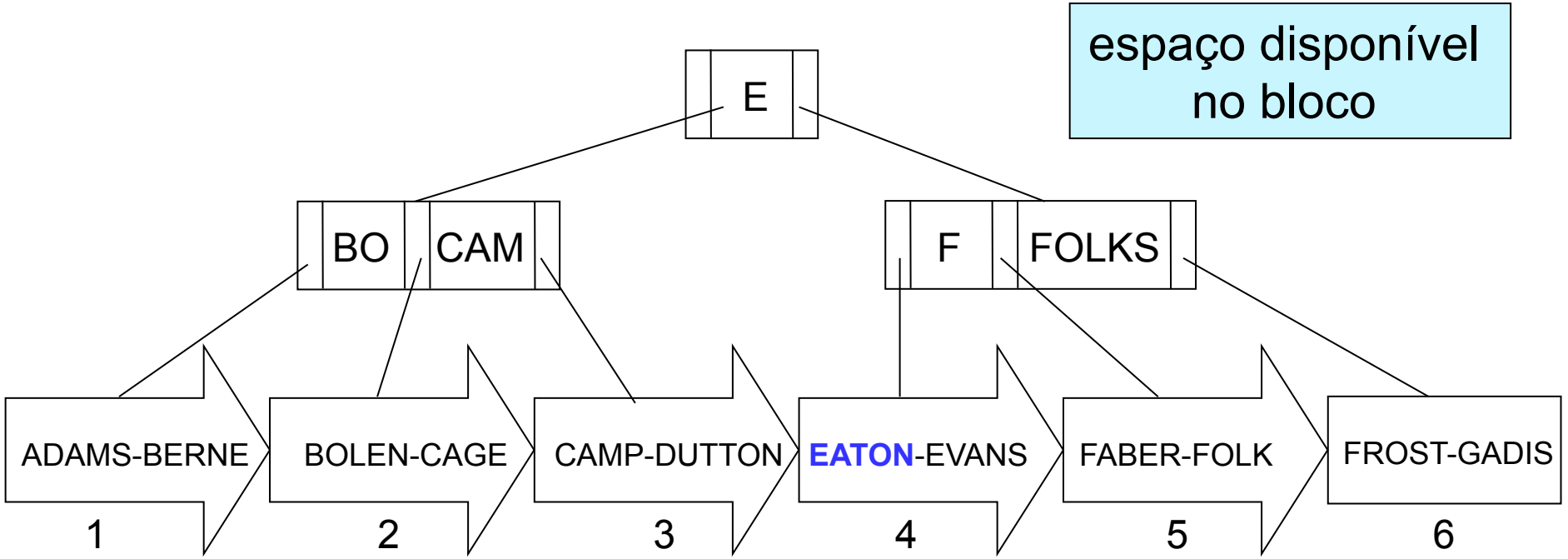


Inserção de EATON



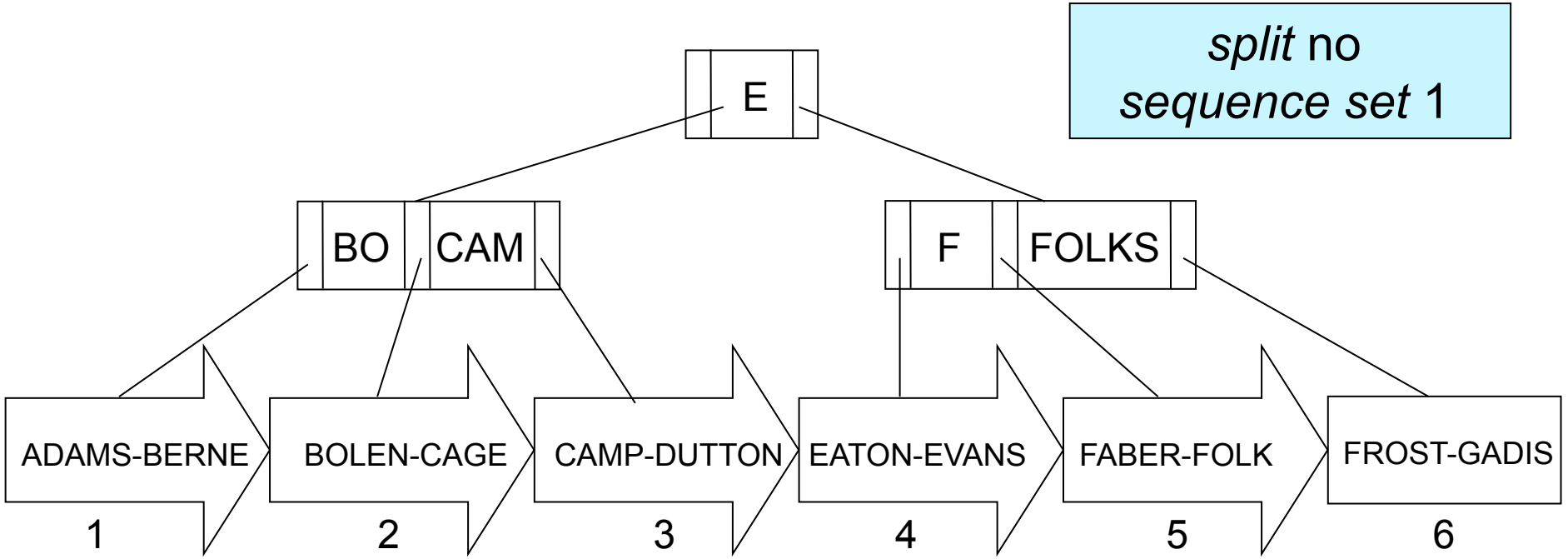
- Efeito no *sequence set*
  - limitado a alterações no bloco 4

Inserção de EATON

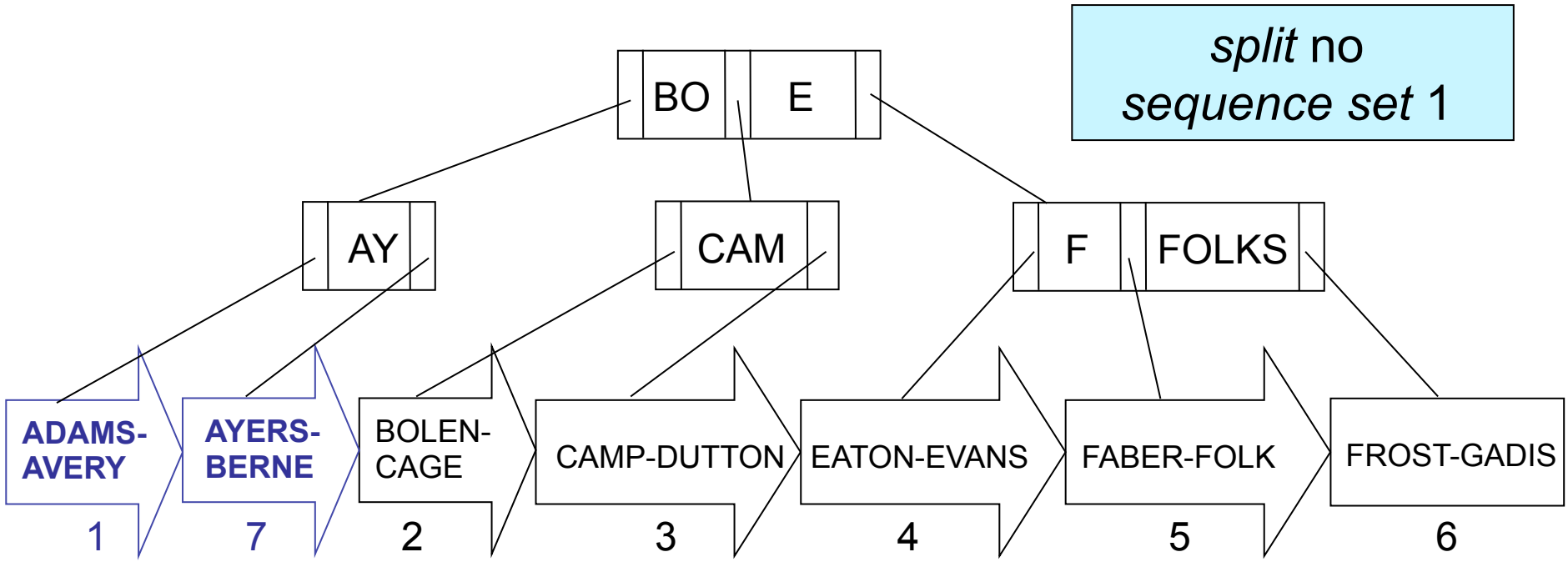


- Efeito na árvore-B+
  - nenhum: E é uma boa chave separadora

Inserção de AVERY

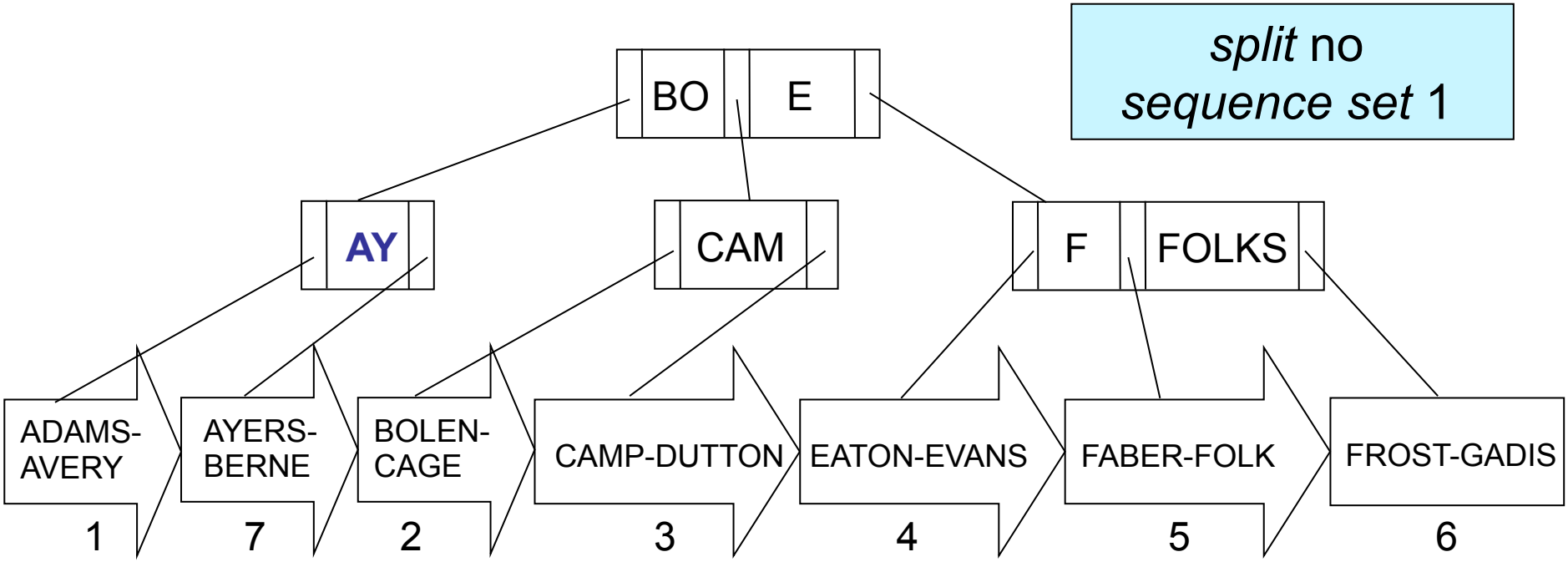


Inserção de AVERY



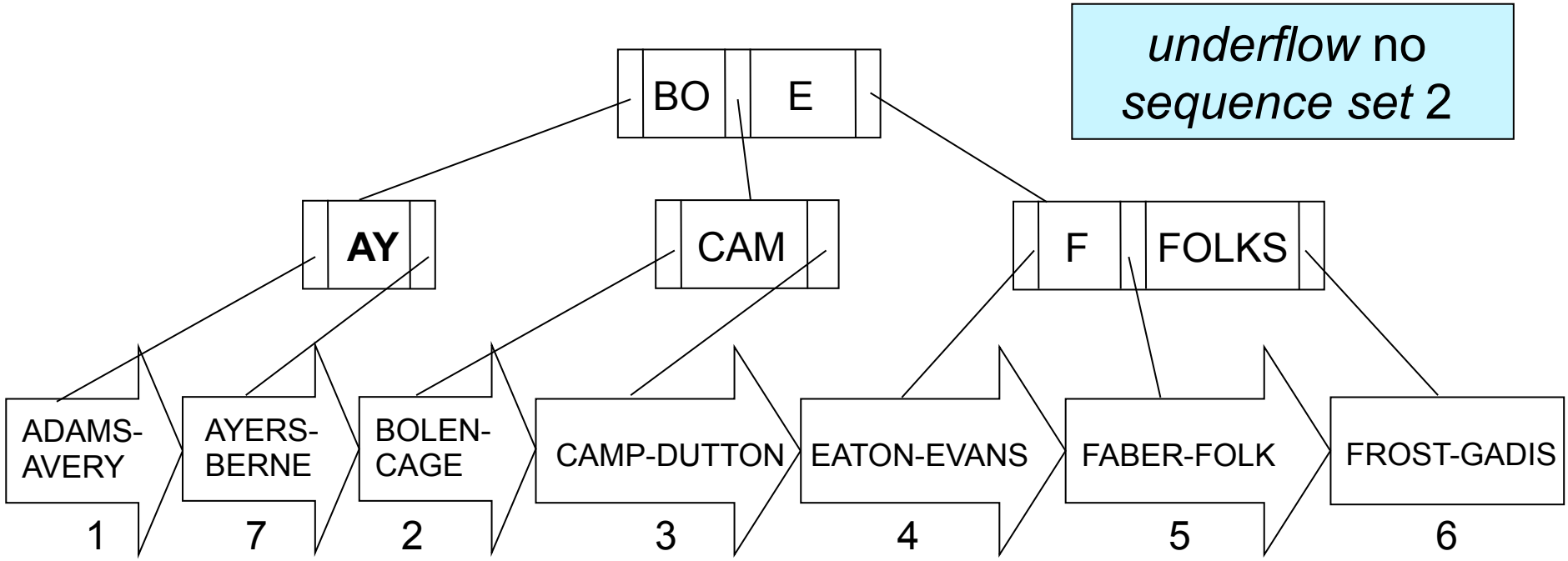
- Efeito no *sequence set*
  - dados do bloco 1 + AVERY distribuídos entre os blocos 1 e 7

Inserção de AVERY

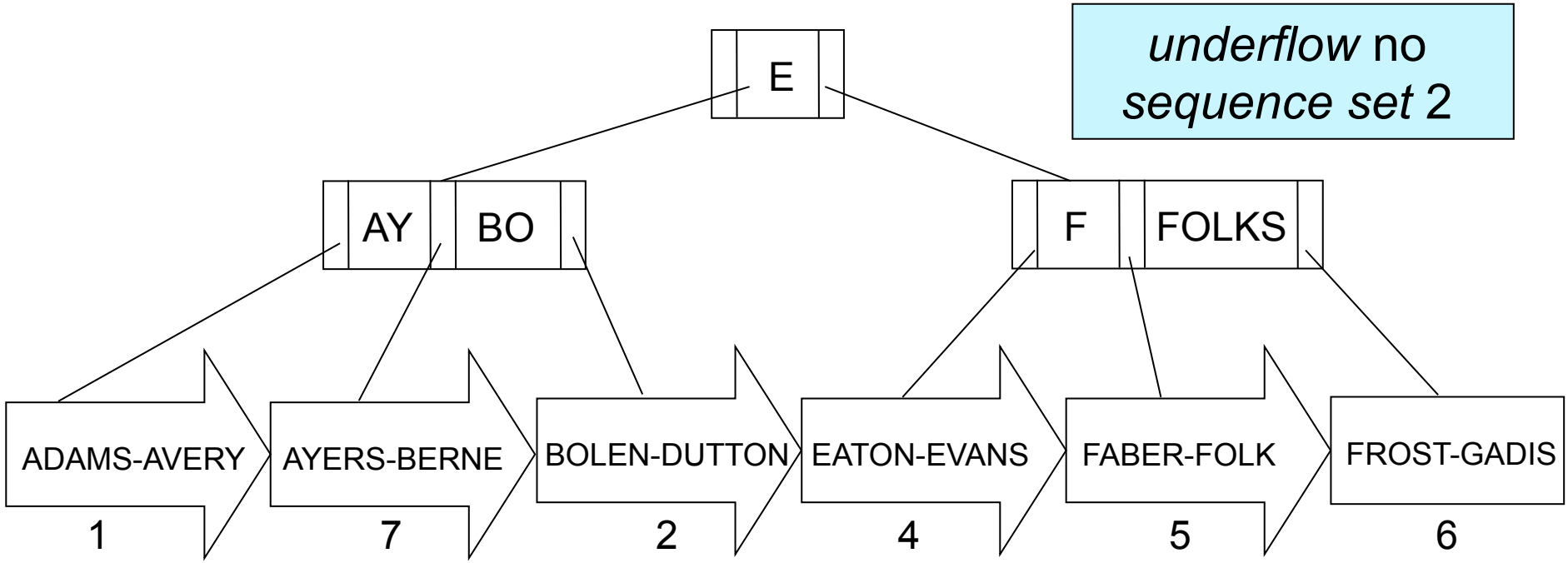


- Efeito na árvore-B+
  - separador adicional AY + *split* + promoção de chave

Remoção de CAEL

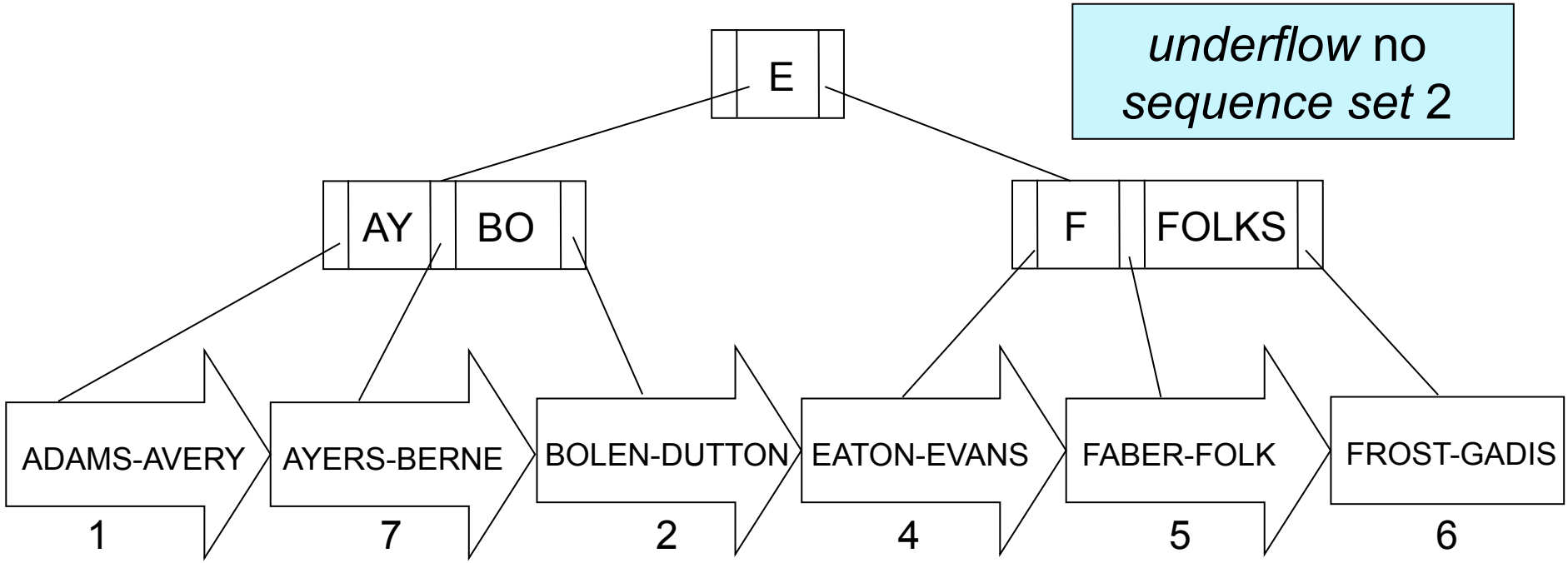


Remoção de CAEL



- Efeito no *sequence set*
  - concatenação dos blocos 2 e 3

Remoção de CAEL



- Efeito na árvore-B+
  - remoção de CAM e concatenação de nós

Inserção e Remoção

- Primeiro passo: *Sequence Set*
  - inserir ou remover o dado
  - tratar, caso necessário
    - *split*
    - contatenação
    - redistribuição

alterações são sempre realizadas a partir do arquivo de dados

Inserção e Remoção

- Segundo passo: *Árvore-B*$^+$
  - se *split* no *sequence set*

    inserir um novo separador no índice
  - se concatenação no *sequence set*

    remover um separador do índice
  - se distribuição no *sequence set*

    alterar o valor do separador no índice

Observações Adicionais

- Tamanho físico de um nó no índice (i.e., árvore-B^+) = Tamanho físico de um bloco no *sequence set*

- Escolha direcionada pelos mesmos quesitos
  - tamanho do bloco
  - características do disco
  - quantidade de memória disponível

Observações Adicionais

- Tamanho físico de um nó no índice (i.e., árvore-B$^+$) = Tamanho físico de um bloco no *sequence set*
- Facilidade para a implementação da árvore-B$^+$ virtual

Observações Adicionais

- Tamanho físico de um nó no índice (i.e., árvore-B^+) = Tamanho físico de um bloco no *sequence set*

- Uso de um mesmo arquivo para armazenar os blocos do índice e os blocos do *sequence set*
  - evita *seeks* entre dois arquivos separados