

Capítulo 9

Árvores-B Virtual, B+tree e Acesso
Seqüencial Indexado



Programa

- Introdução
- Operações básicas sobre arquivos
- Armazenagem secundária
- Conceitos básicos sobre estrutura de arquivo
- Organizando arquivos para desempenho
- Indexação
- Processamento co-seqüencial e ordenação
- B-Tree e outras organizações em árvores
- B+Tree e acesso seqüencial indexado
- Hashing
- Hashing estendido



Árvore-B Virtual

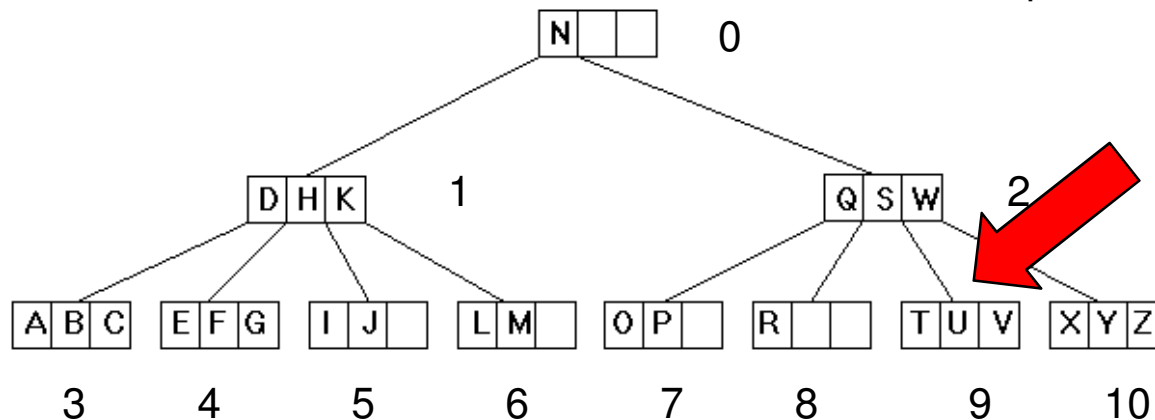
- Proposta
 - Diminuir o número de seeks mantendo a árvore, ou parte da árvore, na memória.
- Mantém a raiz na memória
 - Toda busca através da árvore requer acesso à raiz.
 - Utilizar o restante da memória disponível para outras páginas.

Árvore-B Virtual

- Supondo
 - Página raiz sempre na memória.
 - Páginas de 23 bytes.
 - 92 bytes de memória.

Páginas na memória			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2

Tabela de Mapeamento de Páginas

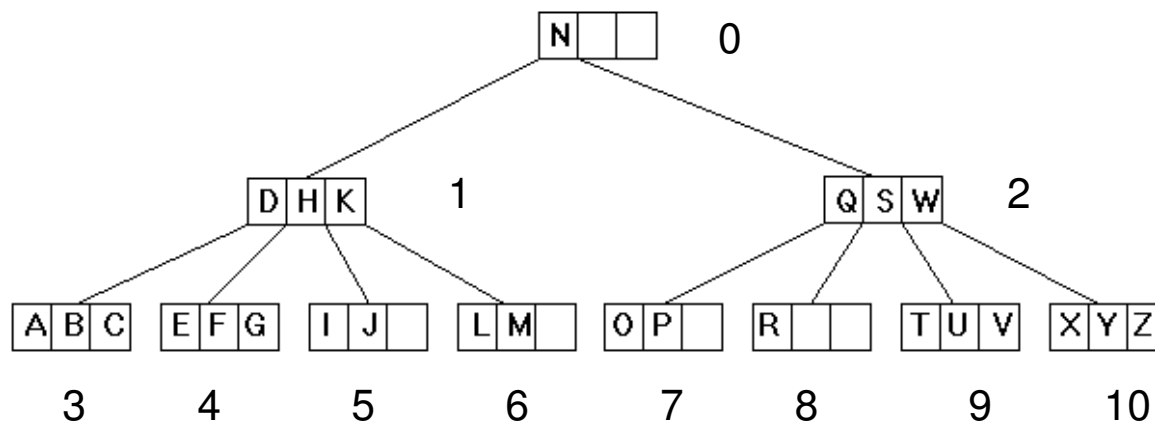


Árvore-B Virtual

- Configuração de páginas necessárias para acessar as chaves T, U ou V:
- Buscar qualquer dessas chaves força trazer para a memória as páginas 2 e 9.

Páginas na memória			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
9	3	T U V	- - - -

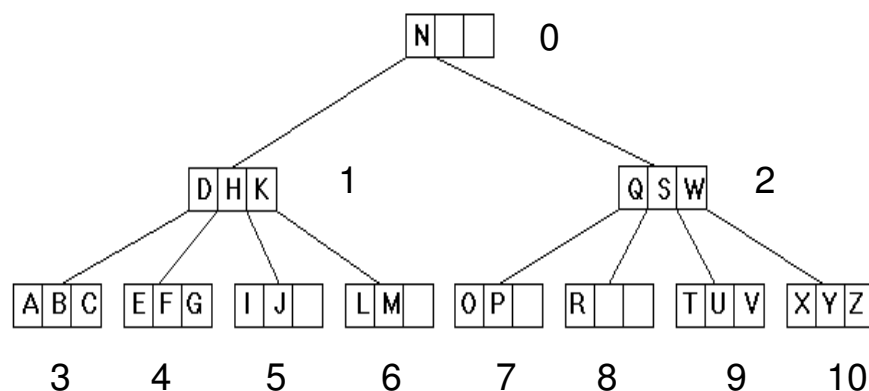
Tabela de Mapeamento de Páginas



A referência de uma página não carregada na memória gera uma interrupção do tipo *page fault*.

Árvore-B Virtual

- Buscar a chave A leva:
 - Incluir a página 1 na quarta posição disponível.
 - Substituir a página 9 pela página 3.



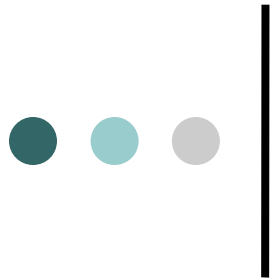
Páginas na memória (antes)			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
9	3	T U V	- - - -

Tabela de Mapeamento de Páginas

Páginas na memória (depois)			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
3	3	A B C	- - - -
1	3	D H K	3, 4, 5, 6

Tabela de Mapeamento de Páginas

Qual página deve ser retirada para liberar espaço para a carga de outra página?



Árvore-B Virtual

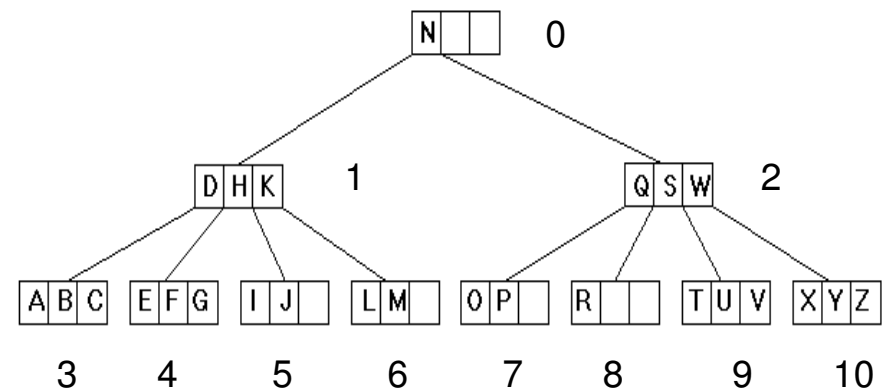
- A próxima busca fora das páginas 1, 2 ou 3
 - Causa um outro “page fault” que exigirá a liberação da memória (pois está cheia).
 - Por exemplo,
 - A busca da chave M utiliza as páginas 0 e 1, que já estão na memória.
 - No momento em que a busca solicita a página 6, ocorre um “page fault”.
- Qual página (2 ou 3) na memória deve ser eliminada para liberar memória para dar lugar à página 6?

Árvore-B Virtual

- O critério pode ser escolher:
 - Aleatoriamente
 - O desempenho será melhor se a página escolhida não for muito referenciada.
 - A página menos usada recentemente (**Least Recently Used**).
 - A página mais distante da raiz.
 - LRU e distância da raiz.

Páginas na memória			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
3	3	A B C	- - - -
1	3	D H K	3, 4, 5, 6

Tabela de Mapeamento de Páginas

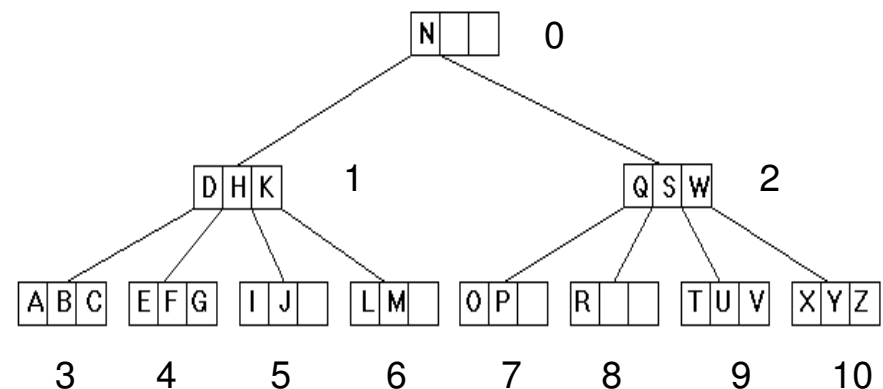


Árvore-B Virtual

- Menos recente usada (LRU)
 - Acredita que páginas utilizadas mais recentemente, continuarão sendo utilizadas
 - As páginas utilizadas há muito tempo, dificilmente serão referenciadas novamente.
 - No exemplo anterior, a página 2 cede espaço para a página 6, quando se deseja acesso a chave M.

Páginas na memória			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
6	2	LM	- - - -
3	3	ABC	- - - -
1	3	DHK	3, 4, 5, 6

Tabela de Mapeamento de Páginas



Árvore-B Virtual

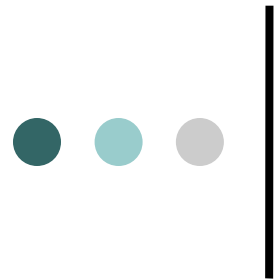
- A página mais distante da raiz.
 - As páginas mais próximas à raiz são as mais referenciadas.
 - Retira a página mais distante.
 - Ocorrendo um “page fault”, retira-se a página que está mais longe da raiz.
 - No exemplo anterior, a página 3 é que será substituída pela 6.

Páginas na memória (antes)			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
3	3	A B C	- - - -
1	3	D H K	3, 4, 5, 6

Tabela de Mapeamento de Páginas

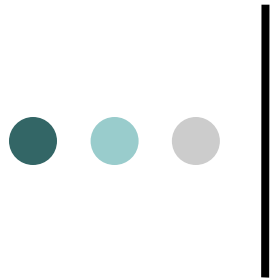
Páginas na memória (depois)			
Pág	Contador	Chaves	Ponteiros
0	1	N	1, 2
2	3	Q S W	7, 8, 9, 10
6	2	L M	- - - -
1	3	D H K	3, 4, 5, 6

Tabela de Mapeamento de Páginas



Árvore-B Virtual

- LRU + Altura (distância da raiz)
 - Pondera as páginas na memória
 - Considera pesos que leva em conta a altura da página e o tempo da sua última utilização.
 - Substitui-se aquela de maior peso
 - Quando duas páginas estão no mesmo nível, substitui-se aquela de maior tempo sem uso.



Árvore-B Virtual

- Referências

- Bayer, R. and McCreight, E. Organization and maintenance of large ordered index. Acta Informatica 1, No. 3 (1972). pp 173-189.
- Knuth, D. E. The art of computer programming: sort and searching. Reading, Massachusetts: Addison-Wesley, 1973.
- Folk, M. J. And Zoellick, B. File Structures. 2th edition. Reading Massachusetts: Addison-Wesley, 1992.
- Sedgewick, Robert. Algorithms. 2th edition. Reading, Massachusetts: Addison-Wesley, 1988.
- Yao, A. C-C, On random 2-3 trees. Acta Informatica 9, No. 2 (1978), pp 159-170.



B+Tree - Objetivos

- Introduzir arquivos sequenciais indexados.
- Descrever as operações num “*sequence set*” de blocos que mantém registros em ordem de chave.
- Mostrar como um conjunto de índices pode ser construído a partir de um *sequence set* para produzir uma estrutura sequencial indexada.

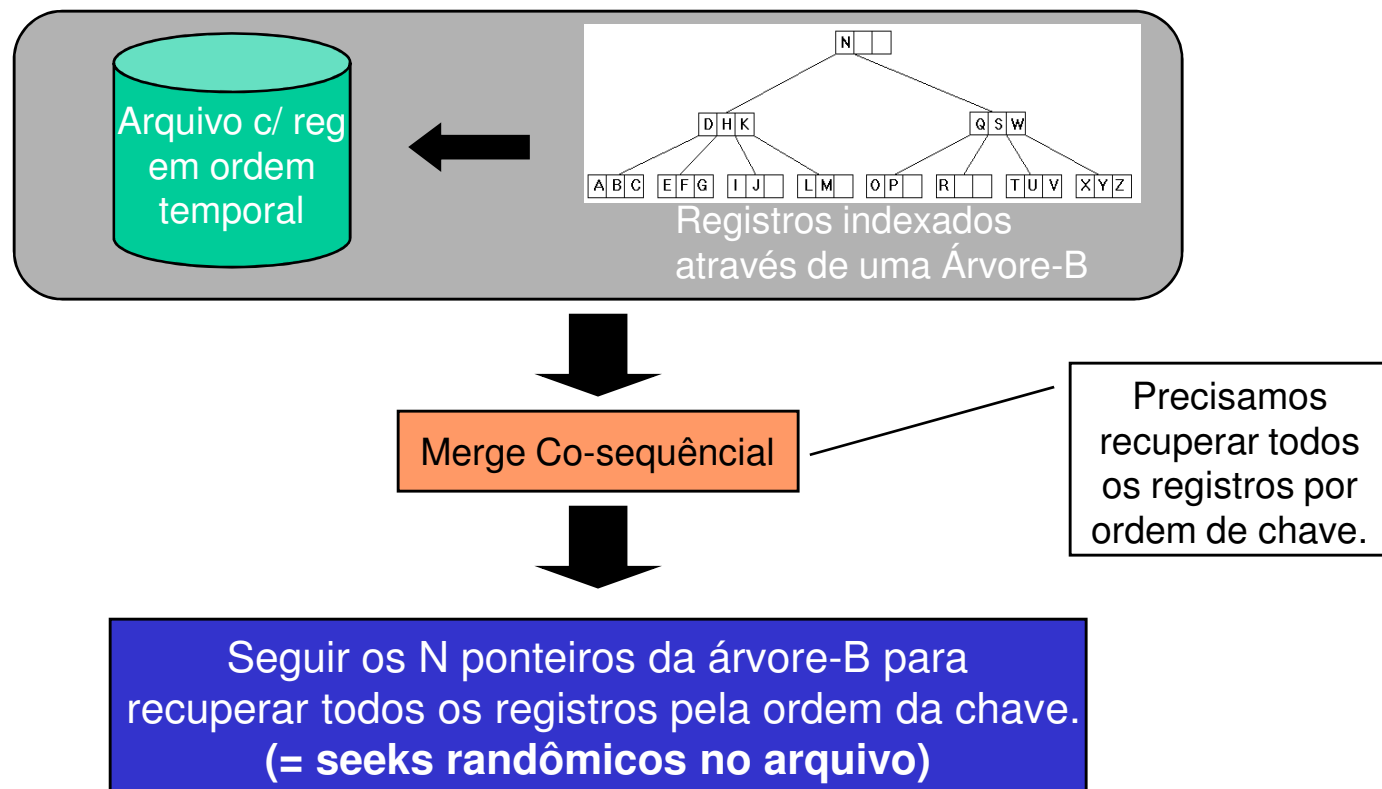
● ● ● | Seqüencial Indexado

- Arquivo seqüencial indexado contempla as duas visões de arquivos:
 - Arquivo indexado
 - Conjunto de registros indexados por uma chave.
 - O registro com uma dada chave pode ser acessado com um único seek, seu NRR (PRR) é conhecido(a).
 - Arquivo seqüencial
 - Arquivo ordenado por uma chave.
 - Acessado seqüencialmente, retorna os registros por ordem de chave, sem seeks.

Ideal: Ter um único método organizacional que proporcione ambas visões de um arquivo, indexado e sequencial.

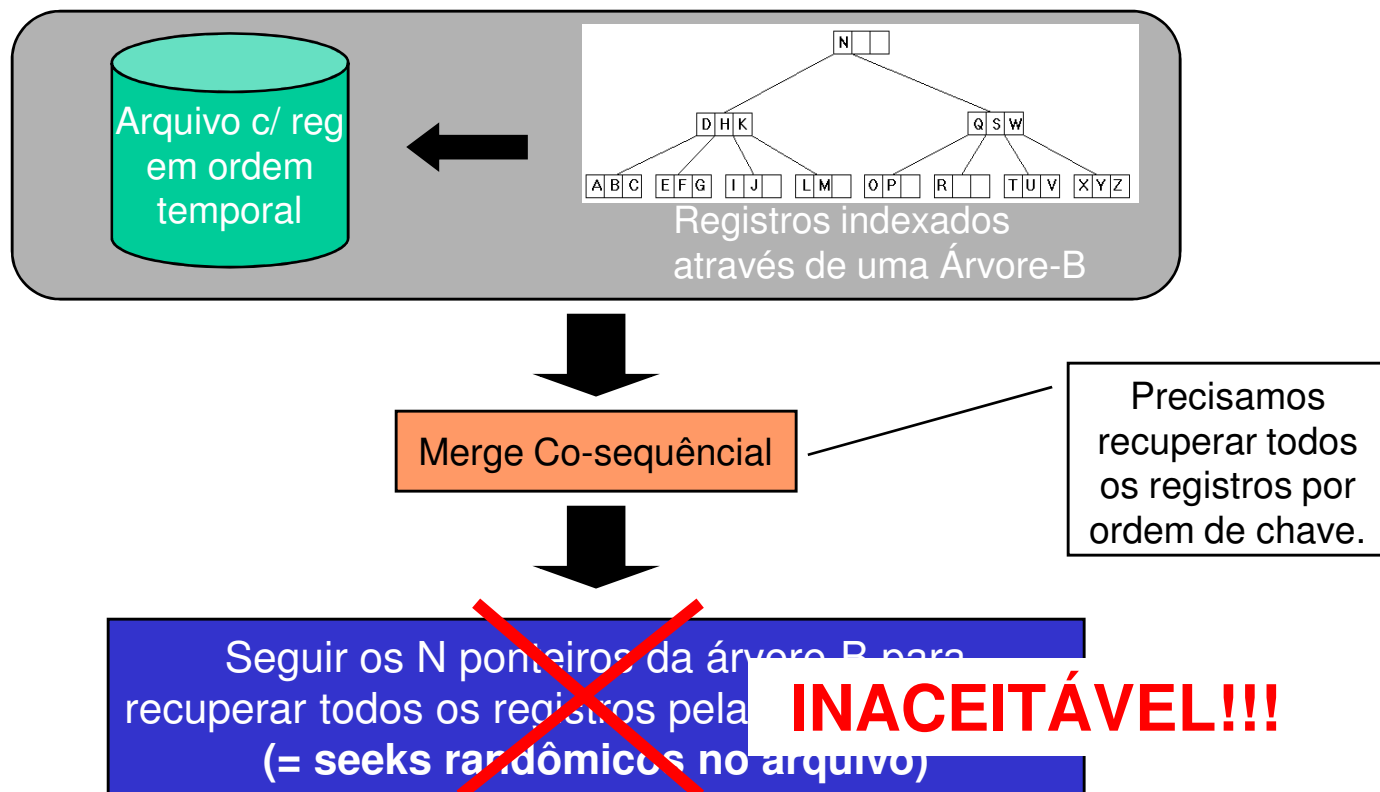
● ● ● | Motivação

- Por exemplo:



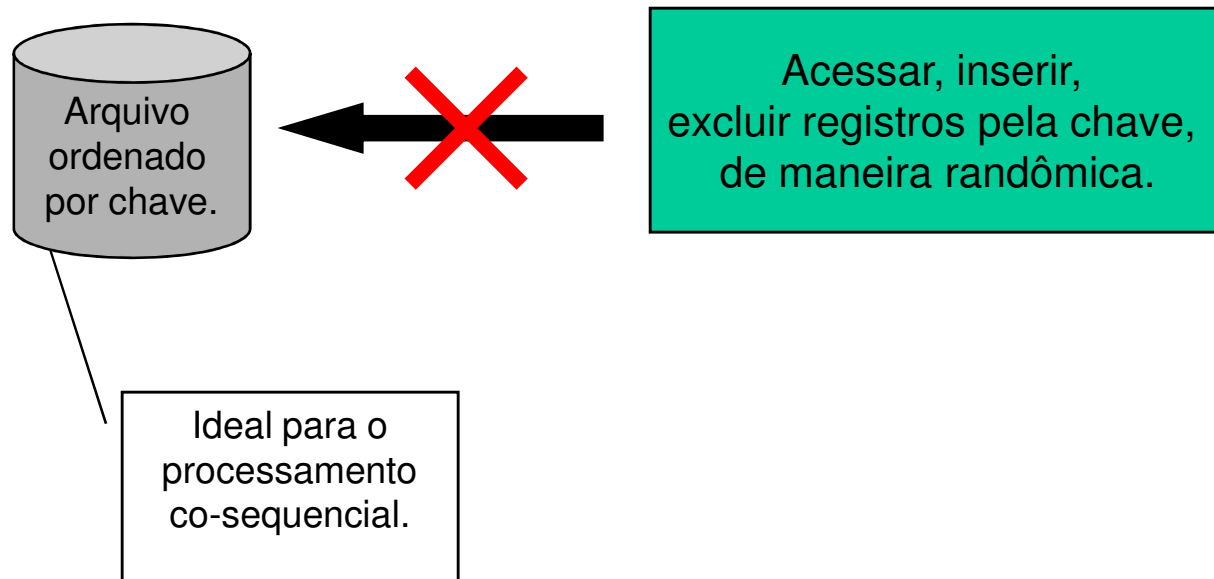
Motivação

- Por exemplo:



● ● ● | Motivação

- Por outro lado,





Motivação

- Aplicações podem envolver ambos, acesso randômico interativo, e processamento co-sequencial em lote.
- Por exemplo:
 - Sistema de controle de estudantes numa universidade:
 - Acesso via chave a registros individuais
 - Grande quantidade de processamento em lote, quando as menções são publicadas, por ex.
 - Sistema de cartão de crédito:
 - Acesso interativo para verificar a situação de uma conta.
 - Processamento em lote quando as faturas são geradas.

Acesso Sequencial Indexado foi criado para atender a essas necessidades.

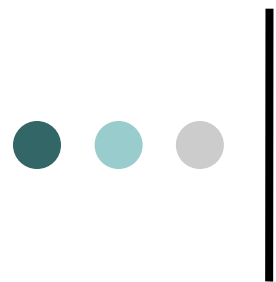
• • • | Seqüencial Indexado

- Primeiro vamos encontrar uma maneira de:
 - Manter os registros fisicamente por ordem de chave, conforme vão sendo inseridos e eliminados.



Sequence Set

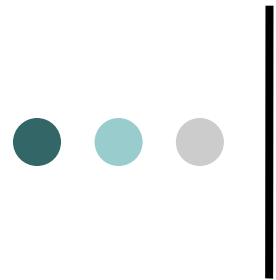
- Depois veremos como indexar o *sequence set*.



Seqüencial Indexado

- Já vimos que reordenar o *sequence set* a cada modificação é muito caro e inviável.
- Precisamos de uma maneira em que as mudanças no arquivo tenham efeito local.
- Vimos que uma das melhores maneiras de restringir os efeitos da inserção e exclusão à apenas parte do *sequence set*, é reunir os registros em blocos.

O bloco se torna uma unidade básica de input e output

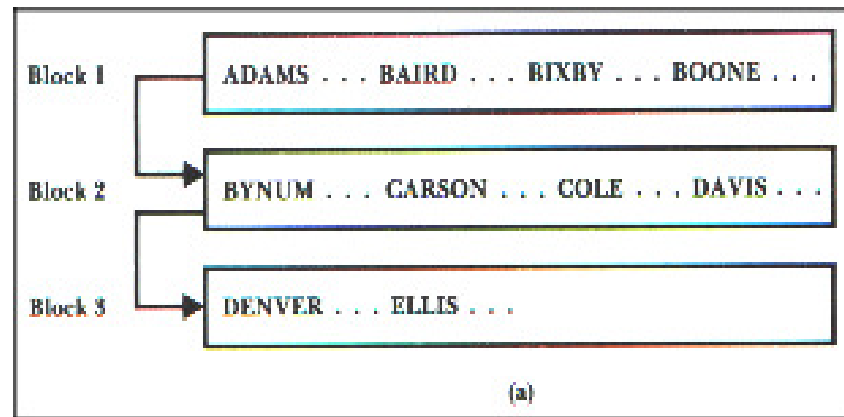


Seqüencial Indexado

- **BLOCO:** Se torna uma unidade básica de input e output. Lemos e escrevemos blocos inteiros de uma vez.
- Conseqüentemente o tamanho dos buffers que usamos num programa deve ser tal que possa conter um bloco inteiro.
- Depois de ler um bloco inteiro, todos os registros daquele bloco estão na RAM, onde nos podemos trabalhar com eles e rearranjá-los mais rapidamente.

Seqüencial por Blocos

- Idéia inicial
 - Arquivo seqüencial em blocos:
 - Blocos ordenados e encadeados, mantendo o arquivo como um todo ordenado.



Ponteiros são necessários pois blocos consecutivos não são, necessariamente, fisicamente adjacentes.



Seqüencial por Blocos

- Propriedades
 - Cada bloco tem um fator de lotação mínimo.
 - Caso essa restrição seja violada, faz-se
 - **Redistribuição** entre os vizinhos ou
 - **Concatenação** com liberação de blocos.
 - Cada bloco tem um máximo de registros.
 - Caso haja estouro pela inclusão de novo registro, o bloco é **dividido**.
 - Esta divisão não envolve promoção.
 - Faz-se encadeamento dos blocos.

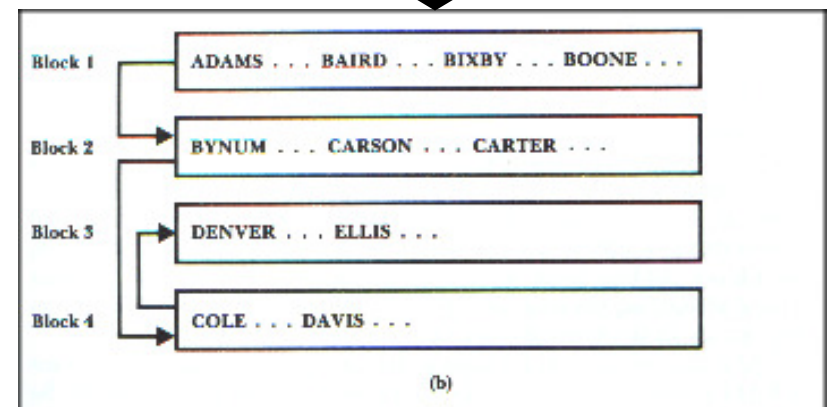
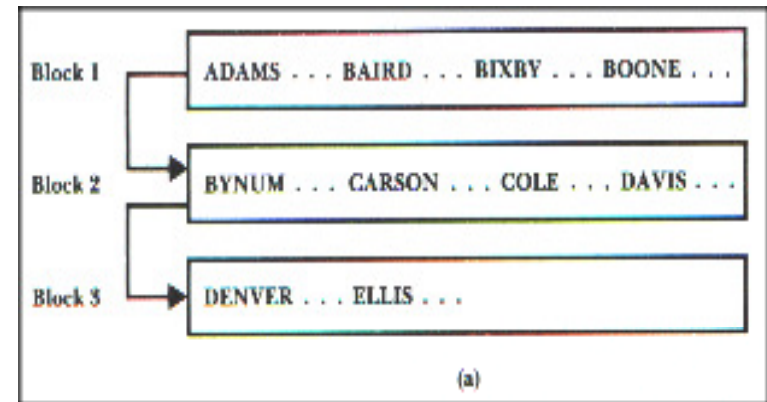


Seqüencial por Blocos

- Operações
 - Inclusão de novos registros
 - Se bloco tem espaço, inclui na ordem correta dentro do bloco.
 - Se o bloco está cheio, faz-se a divisão do bloco e a redistribuição com a inclusão no lugar pertinente.
 - Retirada de registros
 - Se o bloco não ficou com sublotação, então fim.
 - Se o bloco ficou com sublotação, então
 - Se um dos vizinhos tem lotação superior a mínima, faça uma redistribuição com ele.
 - Senão concatene com o bloco sucessor, liberando-o.

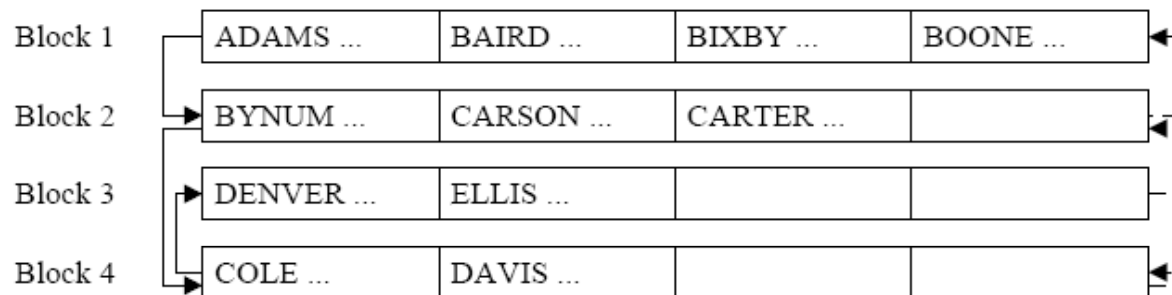
Seqüencial por Blocos

- Inclusão de CARTER provoca estouro de bloco:
- O procedimento é
 - a) Ler o bloco 2 levar para a memória.
 - b) Transferir o conteúdo do bloco 2 para uma área de trabalho com máximo + 1 registros.
 - Inserir o registro novo na ordem correta.
 - Divisão: transferir a metade para o bloco 2 e a outra metade para o bloco novo.
 - c) Atualizar os ponteiros e grava bloco velho e novo.

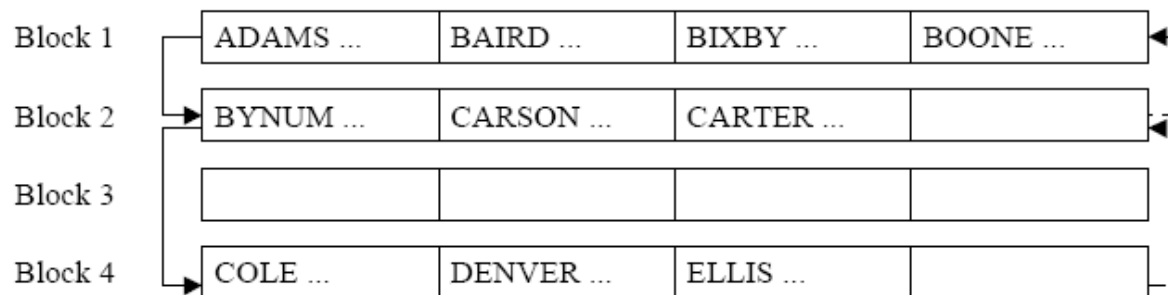


Seqüencial por Blocos

- Retirada de DAVIS

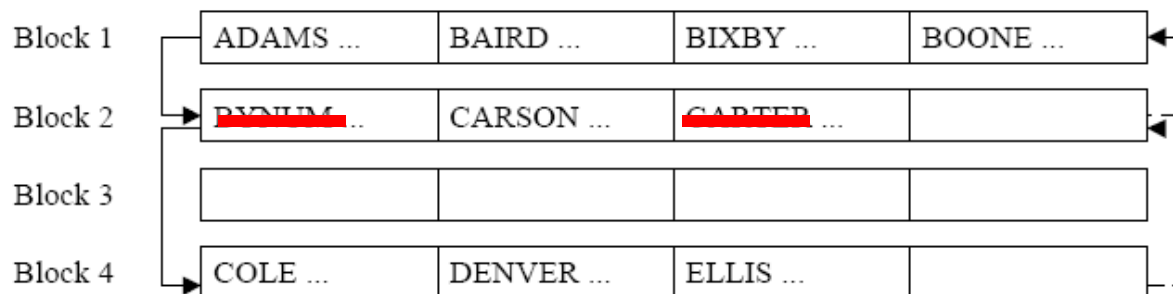


- Há concatenação lógica do bloco 4 com o seu sucessor, bloco 3. Como resultado o bloco 3 fica livre para reuso.

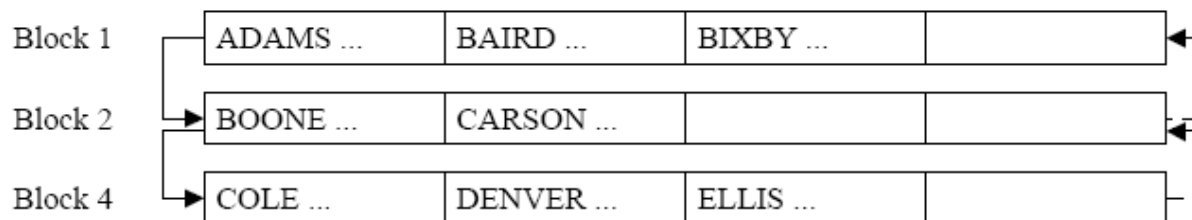


Seqüencial por Blocos

- Retirada de BYNUM (Apenas remove o mesmo do bloco 2)
- Retirada de CARTER



- Podemos fazer a concatenação dos blocos 2 e 4 ou redistribuir os registros entre os blocos 1 e 2.



Quando os dois blocos vizinhos estão cheios, redistribuição é a única alternativa.

● ● ● | Seqüencial por Blocos

- A separação dos registros em blocos
 - Com as operações de divisão, concatenação e redistribuição,
 - Mantém os registros ordenados por chave.
 - Mas há um desperdício de armazenagem pela fragmentação dos blocos.
 - Para minimizar o problema, pode-se aplicar as mesmas estratégias da árvore-B:
 - » Redistribuição em lugar da divisão durante a inserção.
 - » Divisão 2-para-3 (Árvore-B*)

Em qualquer uma das implementações citadas a cima, deve-se levar em consideração que estamos tratando de bloco e não de árvore, portanto não há “promoção”.



Seqüencial por Blocos

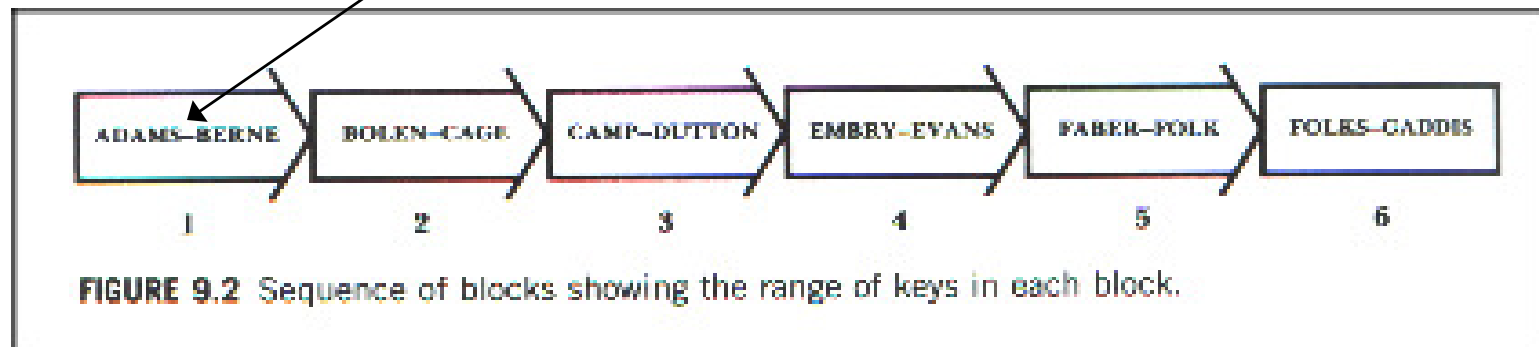
- Qual o tamanho ideal de um bloco?
 - Requer mais de um bloco na memória (divisão, concatenação, redistribuição).
 - O tamanho do bloco deve nos permitir manter vários blocos na memória.
 - Considerando o acesso randômico de um registro do *sequence set*:
 - Temos que ler o bloco todo para acessarmos um registro dentro daquele bloco.
 - O tamanho do bloco não deve ser tão grande que a leitura seqüencial leve o tempo de um seek!

Seqüencial Indexado

- Vamos ver uma maneira eficiente de localizar um bloco específico contendo um determinado registro, dada a chave do registro.

Existem chaves intermediárias

Uma visão externa dos blocos.



As chaves indicadas em cada bloco são a maior e a menor no bloco.

Seqüencial Indexado

- Adicionando índices simples:

FIGURE 9.3 Simple index for the sequence set illustrated in Fig. 9.2.

Key	Block number
BERNE	1
CAGE	2
DUTTON	3
EVANS	4
FOLK	5
GADDIS	6

As chaves representam os limites em cada bloco, ou seja, o último registro.

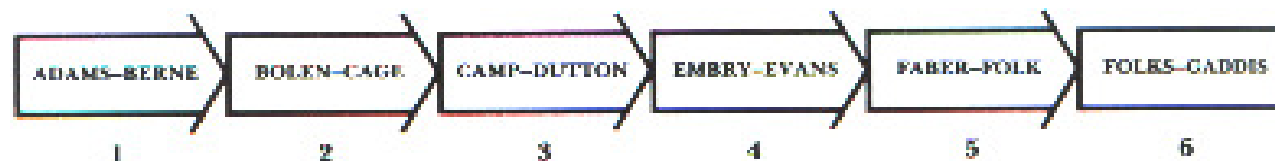
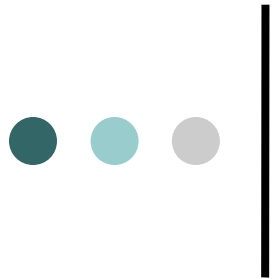


FIGURE 9.2 Sequence of blocks showing the range of keys in each block.



Seqüencial Indexado

- Acesso sequencial indexado
 - A combinação do índice com o arquivo sequencial em blocos dá o **acesso sequencial indexado**.
 - Pode-se acessar um registro específico via índice (recuperando o bloco correto).
 - O **acesso sequencial** é obtido pela leitura sequencial dos blocos, a partir do primeiro.
- O índice cabe em memória (c/ visto cap.06)
 - Cada entrada no índice está associada a um bloco.
 - Faz-se pesquisa binária no índice para descobrir o bloco de um determinado registro.
 - Dentro do bloco faz-se pesquisa binária ou sequencial para achar o registro.

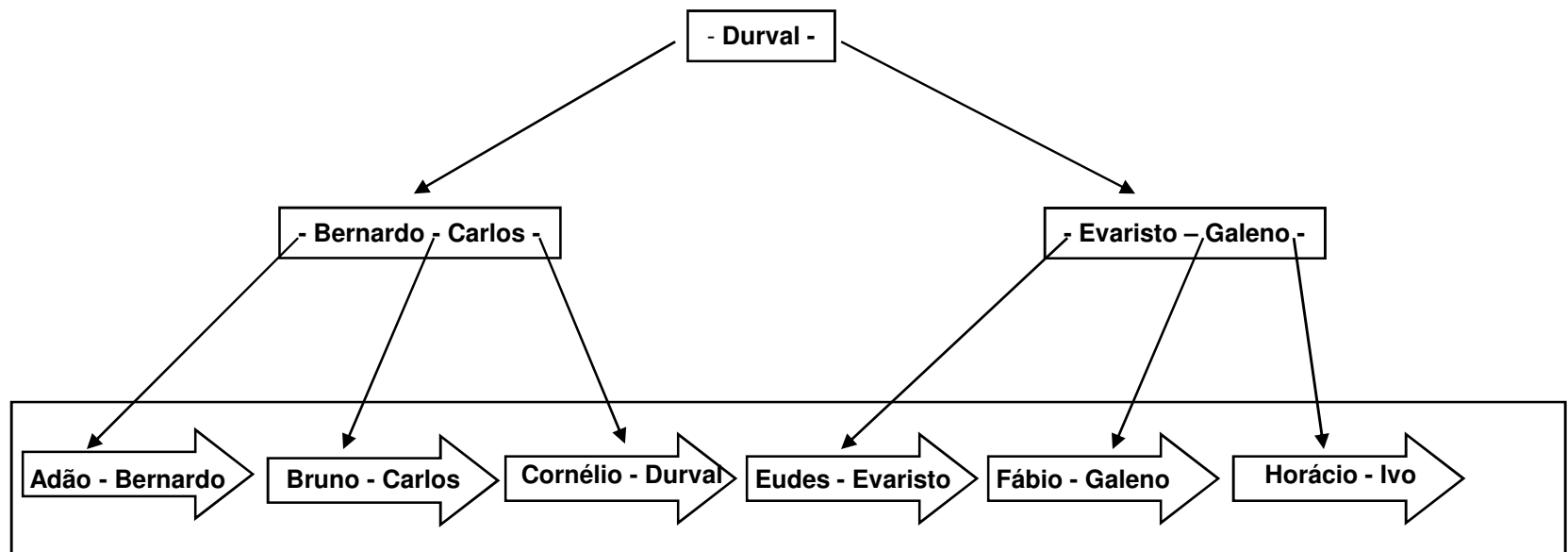
● ● ● | Seqüencial Indexado

- Índice em disco
 - Caso o índice não caiba na memória
 - Ele pode ser mantido em disco na forma de uma árvore-B.
 - O uso de uma árvore-B para indexar um arquivo sequencial em bloco é chamado de **árvore-B⁺**.
 - As chaves nesta árvore estão associadas aos blocos de registros e não a registros.

Árvore-B⁺ = árvore-B com os índices + *sequence set*

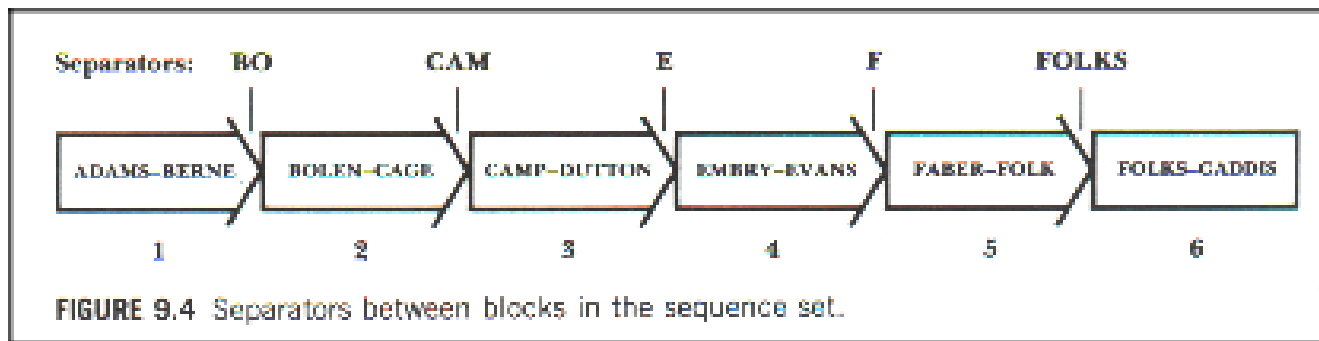
Seqüencial Indexado

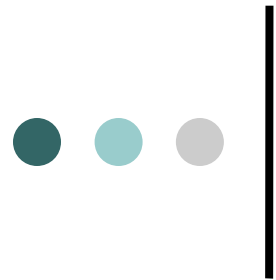
- Árvore-B⁺



Seqüencial Indexado

- O conjunto de índices é apenas um mapa para nos auxiliar a obter o bloco correto.
 - Não contém as respostas, mas sim onde devemos ir para obtermos as respostas.
- Separadores ao invés de chaves
 - Índices usados como separadores não precisam ser chaves completas.

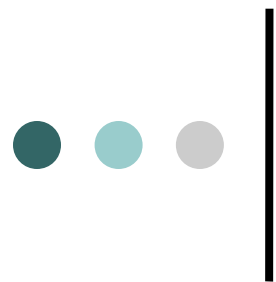




Seqüencial Indexado

- Tabela de decisões:

Relation of Search Key and Separator	Decision
Key < separator	Go left
Key = separator	Go right
Key > separator	Go right

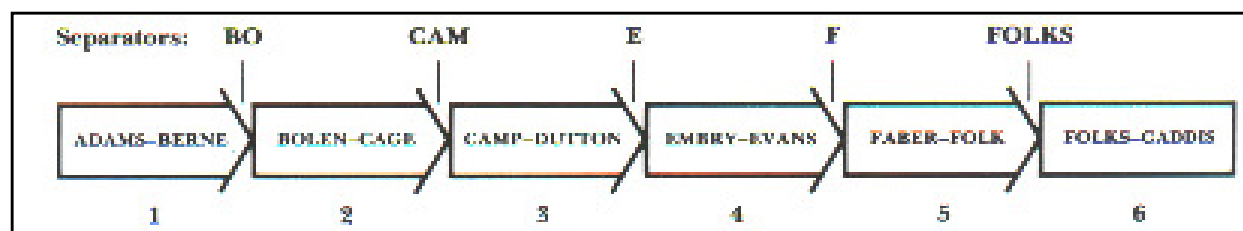
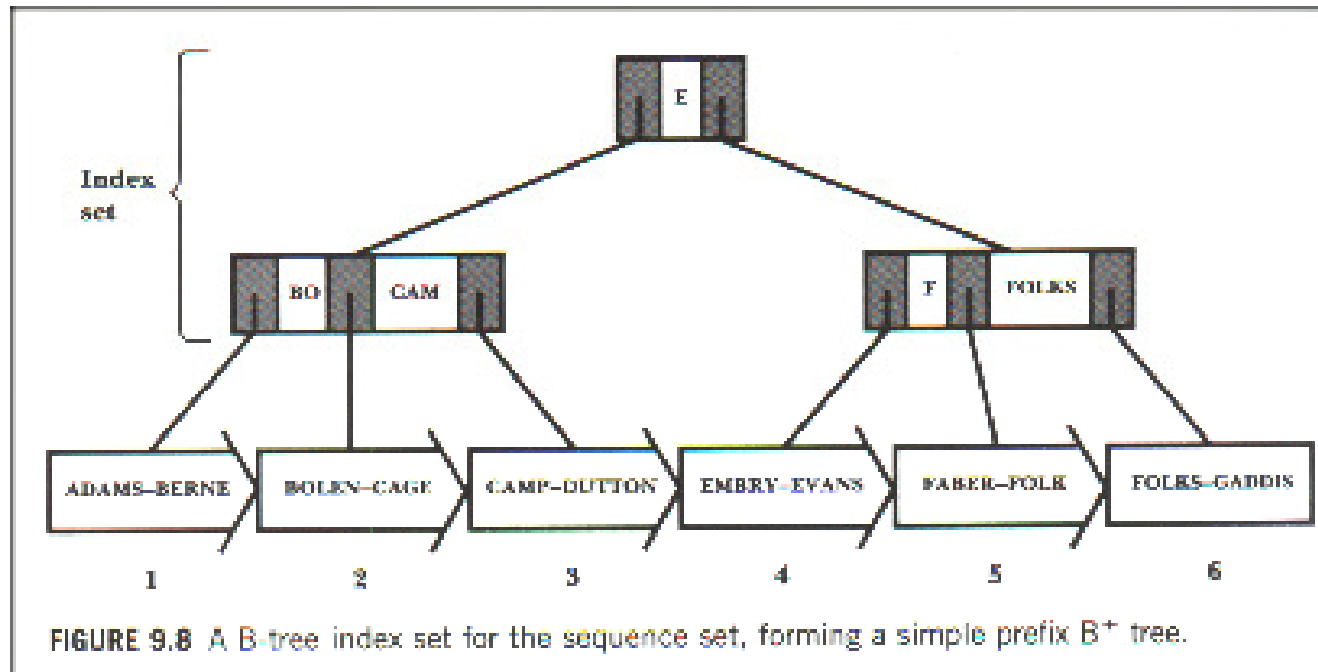


Árvore-B⁺ Prefixada Simples

- Ao invés de chaves, a árvore de acesso tem somente separadores, que são menores que as chaves ou cópias curtas das chaves.
- Na árvore de acesso não há ponteiros para blocos nos nós internos, eles ficam todos nas folhas.
- Se há N separadores, então haverá N+1 nós folhas na árvore de acesso (pois é uma árvore-B).
- Tudo que se falou sobre árvore-B se aplica.

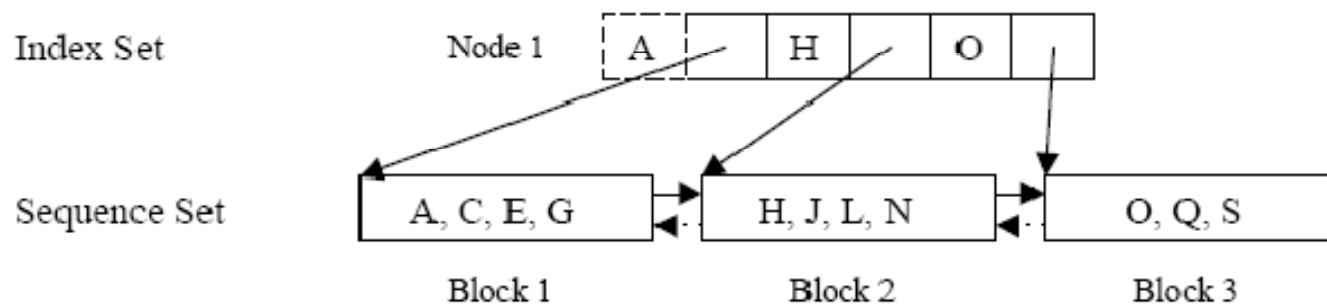
A árvore contém os prefixos das chaves, ao invés das chaves reais.

Árvore-B⁺ Prefixada Simples



Árvore-B⁺ Prefixada Simples (Manutenção)

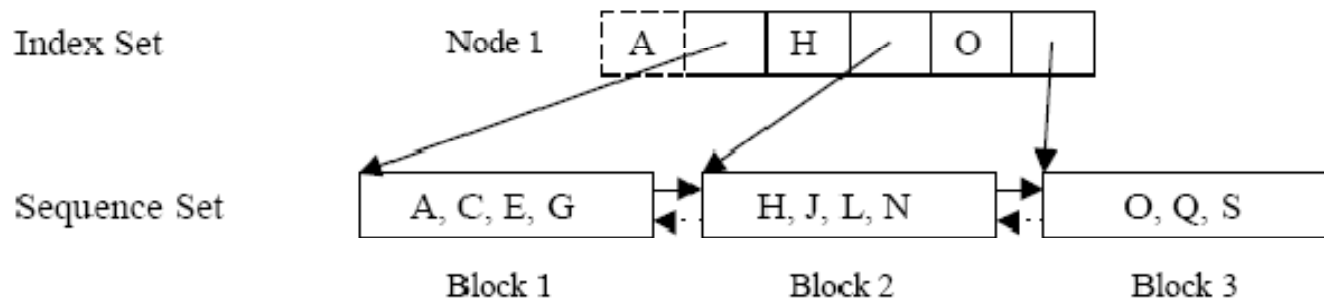
- Fator de blocagem: 4
- Conjunto de índices: árvore-B⁺ de ordem 3 (ou seja, 3 ponteiros e 2 chaves por nó da árvore).



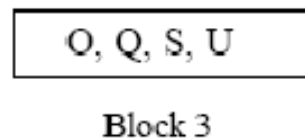
Na verdade, "A" não está no nó 1.

Árvore-B⁺ Prefixada Simples (Manutenção)

- 1. Mudanças que são locais a blocos únicos no *sequence set*:
- Inserir “U”:



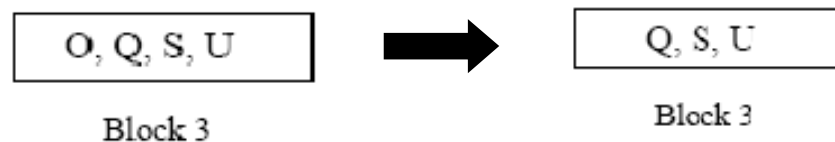
- Vá para a raiz, depois para a direita de “O”, insira “U” no bloco 3.
- A única modificação é:



Não há modificação no conjunto de índices.

Árvore-B⁺ Prefixada Simples (Manutenção)

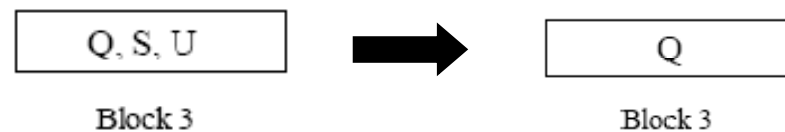
- Excluir “O”:
 - Vá para a raiz, depois para a direita de “O”, exclua “O” do bloco 3. A única modificação é:



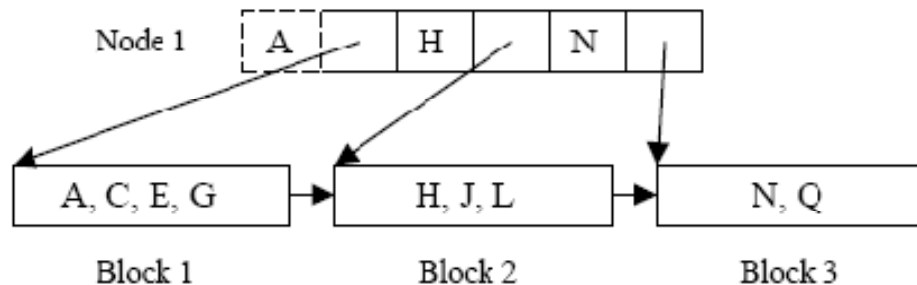
Não há modificação no conjunto de índices.
“O” continua sendo um separador perfeito para o blocos 2 e 3.

Árvore-B+ Prefixada Simples (Manutenção)

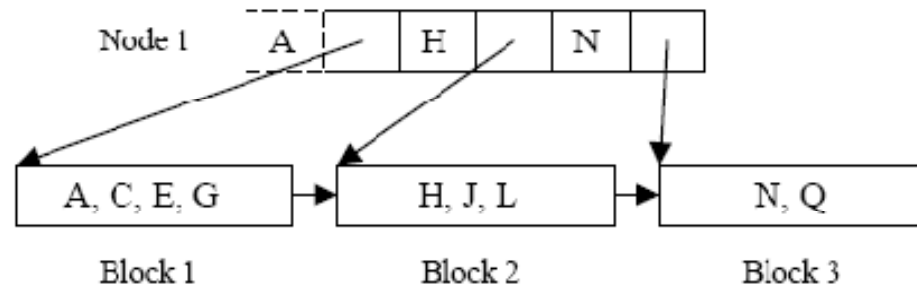
- 2. Mudanças envolvendo blocos múltiplos no *sequence set*:
- Excluir “S” e “U”:
 - Ocorre um “underflow” no bloco 3.



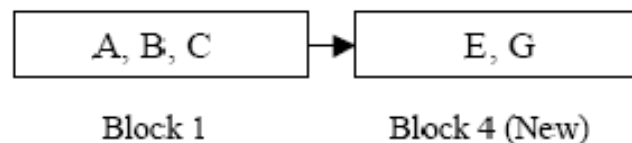
- Uma vez que o bloco 2 está cheio, a única opção é a redistribuição trazendo uma chave do bloco 2 para o bloco 3.
- Temos que atualizar o separador “O” para “N”.



Árvore-B⁺ Prefixada Simples (Manutenção)

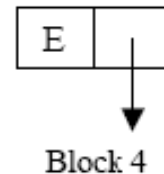


- Inserir “B”:
 - Vá para a raiz, depois para a esquerda de “H” e para o bloco 1.
 - Bloco 1 conterá A,B,C,E,G
 - O Bloco 1 é dividido:

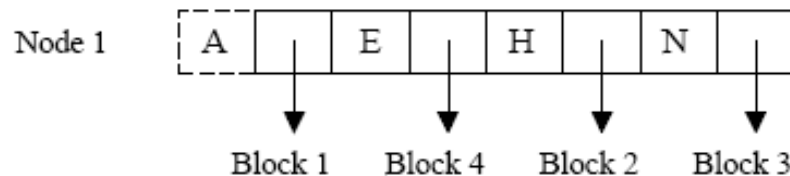


Árvore-B⁺ Prefixada Simples (Manutenção)

- Promova um novo separador “E” juntamente com o ponteiro para o novo bloco 4.

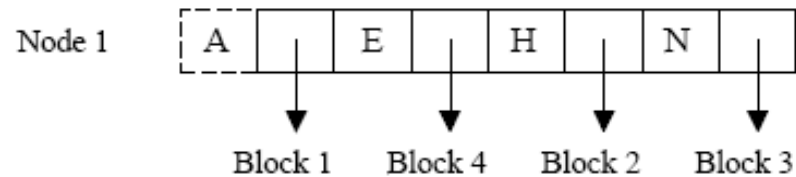


- Desejaríamos ter:

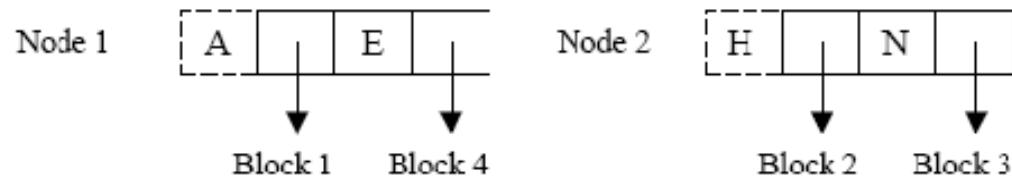


- Mas a ordem do conjunto de índices é 3 (3 ponteiros, 2 chaves).

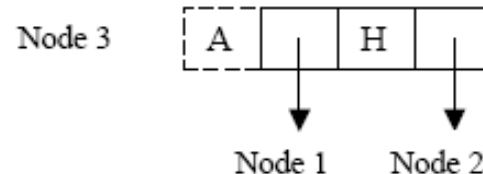
Árvore-B+ Prefixada Simples (Manutenção)



- Divisão do nó:

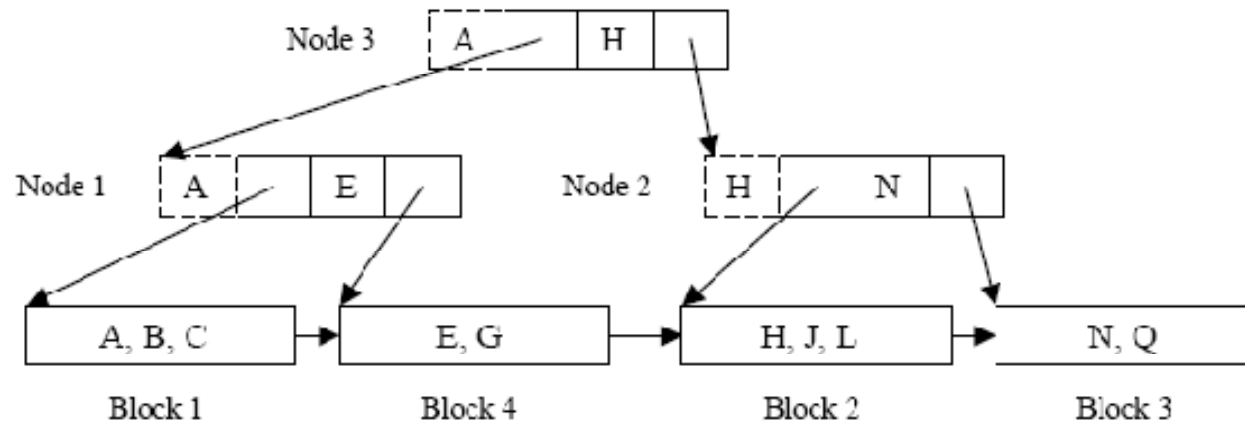


- Crie uma nova raiz para apontar para ambos os nós:



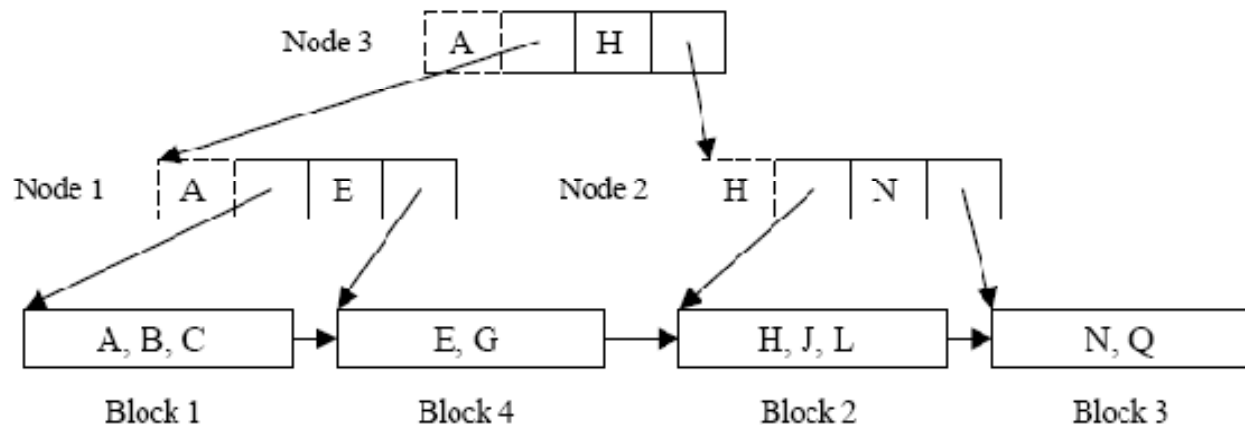
Árvore-B⁺ Prefixada Simples (Manutenção)

- A nova árvore:



Lembre-se, na verdade “A” não está presente nos nós 1 e 3, e “H” não está presente no nó 2.

Árvore-B⁺ Prefixada Simples (Manutenção)



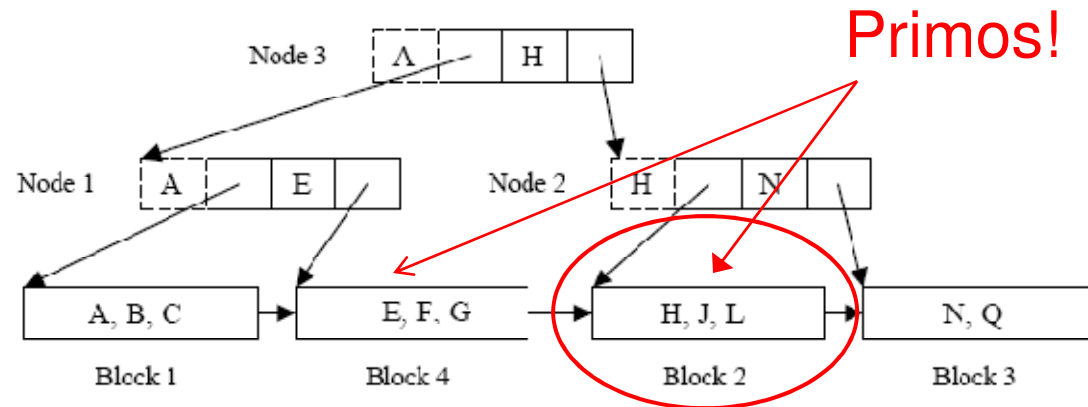
- Inserir “F”:
 - Vá para a raiz, depois para a esquerda de “H”, depois para a direita de “E” no nó 1.
 - Insira “F” no bloco 4.



Não há modificação
no conjunto de índices.

Árvore-B⁺ Prefixada Simples (Manutenção)

- Excluir “J” e “L”:
 - Bloco 2 sofre um “underflow”:

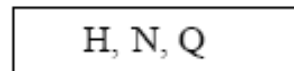


- Pensaríamos numa redistribuição entre os blocos 4 e 2, ou seja, E, F, G e H ficariam E, F e G, H.
- Porque isso não é possível?

Os blocos 4 e 2 não são irmãos!!!
São primos!

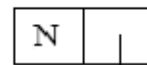
Árvore-B⁺ Prefixada Simples (Manutenção)

- O único irmão do bloco 2 é o bloco 3.
- Redistribuição não é possível entre H e N,Q.
- Portanto, a única possibilidade é fazer o merge dos blocos 2 e 3.



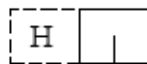
Block 2

- Enviar o bloco 3 para uma LISTA DE BLOCOS DISPONÍVEIS.
- Remover o separador N e o ponteiro do nó 2.



Block 3

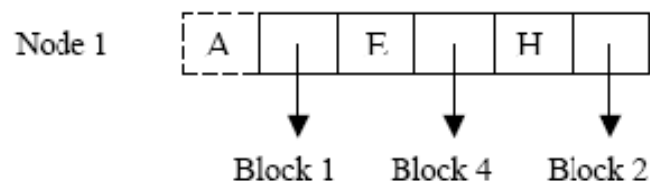
- Causa um “underflow” no nó 2.



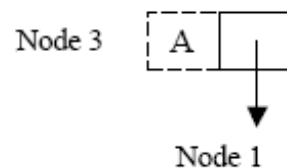
Block 2

Árvore-B⁺ Prefixada Simples (Manutenção)

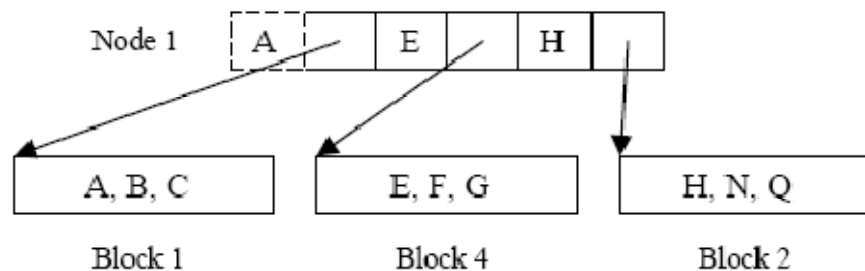
- A única possibilidade é fazer um merge com o nó irmão (nó 1):



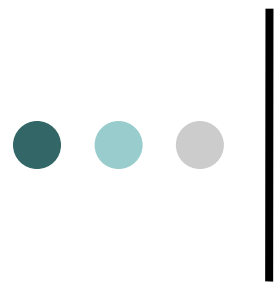
- “H” desce da raiz, o que causa um “underflow” da raiz.



- A raiz é removida e o nó 1 se torna a nova raiz.



Árvore final após as modificações



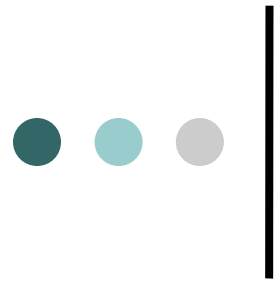
Árvore-B⁺ Prefixada Simples

- Tópicos Avançados para Reflexão:
 - Geralmente os blocos do *sequence set* e os nós do conjuntos de índices estão armazenados num mesmo arquivo.
 - Os nós do conjunto de índices podem ter um estrutura interna complexa para armazenar separadores de tamanho variável, e permitir pesquisa binária neles. (ver livro texto p.423 Fig.9.13)
 - Árvores-B⁺ Prefixadas Simples e Árvores-B⁺ normais são muito similares. A diferença é que a última armazena chaves reais ao invés de separadores/prefixos. (ver seção 9.10 do livro texto, pag. 429)



Observações finais

- Atenção: Árvores-B e árvores-B⁺ são estruturas de arquivos muito poderosas e flexíveis, mas é um grande erro achar que elas servem de solução para todos os problemas.
 - Por exemplo: estruturas de índices simples como visto no Cap.06, que podem ser mantidos em RAM, são mais simples quando suficientes para resolver o problema em questão.
- Ler págs 431 a 434 para um comparativo entre árvores-B e árvores-B⁺.
- Ler o SUMMARY dos capítulos.



Próxima aula...

- Hashing