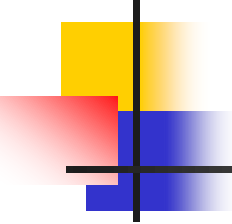




Árvores-B⁺ (B⁺-Trees)

Estrutura de Dados II



Acesso Sequencial Indexado (Indexed Sequential Access)

- Problema: arquivos devem suportar acesso indexado eficiente **e também acesso sequencial.**
- Não é possível fazer acesso sequencial eficiente em um arquivo *entry-sequenced*, cujo índice é mantido como árvore-B.



Acesso Sequencial Indexado

- **Duas visões alternativas:**
 - **Indexado:** o arquivo pode ser visto como um conjunto de registros que é indexado por chave.
 - **Sequencial:** o arquivo pode ser acessado sequencialmente, retornando os registros em ordem de chave.

Organização de Arquivos
Sequencial
Sequencial Indexado
Indexado
Direto



Acesso Sequencial Indexado

- A idéia de ter uma forma de organização que fornece os dois tipos de acesso não foi considerada até o momento.



Acesso Sequencial Indexado

- Suponha um arquivo que consiste de um conjunto de registros *entry-sequenced*, que é acessado através de índices estruturados como árvores-B.
- Isso garante um acesso aleatório eficiente a qualquer registro, mas se este mesmo arquivo precisar ser utilizado por um merge consecutivo, será necessário recuperar os registros em ordem crescente de chaves.



Acesso Sequencial Indexado

- O problema é que a única forma de recuperar os registros em ordem crescente de chaves é através do índice (já que os registros no arquivo aparecem na ordem em que foram entrados).
- Para um arquivo com N registros, isso significa fazer N acessos aleatórios ao disco (N seeks), um processo bem menos eficiente que a leitura sequencial de registros fisicamente adjacentes.



Acesso Sequencial Indexado

- Por outro lado, manter um conjunto de registros ordenados por chave é inaceitável quando queremos acessar, inserir e deletar por chave em ordem aleatória.



Acesso Sequencial Indexado

- Exemplo:
 - Cartão de Crédito
 - Emissão de contas: sequencial
 - Consultas/transações interativas: indexado
- Nestes casos, precisamos de uma forma de organizar os arquivos que suporte ambos os tipos de acesso eficientemente.



Mantendo um Sequence Set

- Consideremos, primeiramente, o problema de manter um conjunto de registros ordenado segundo alguma chave assim que os registros são adicionados e removidos.
- Esse conjunto ordenado de registros é referenciado como Sequence Set.

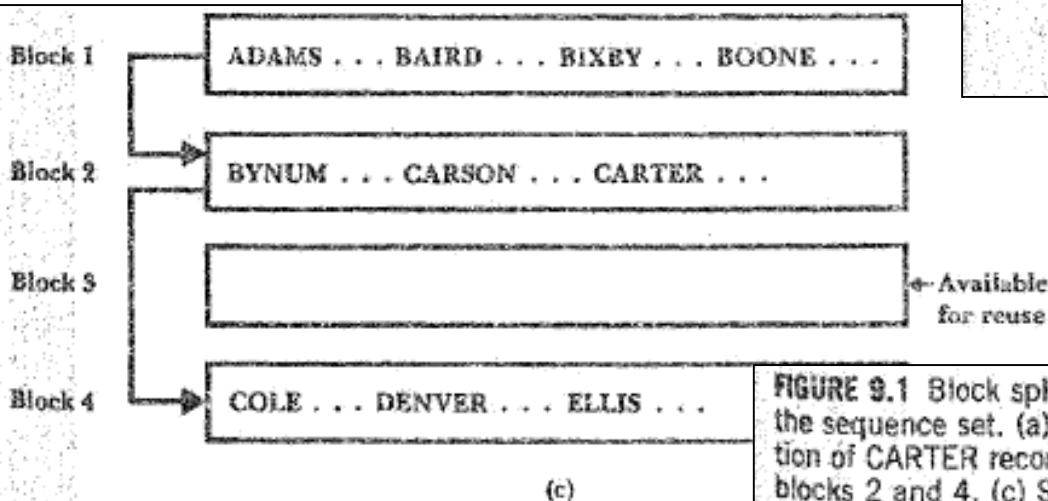
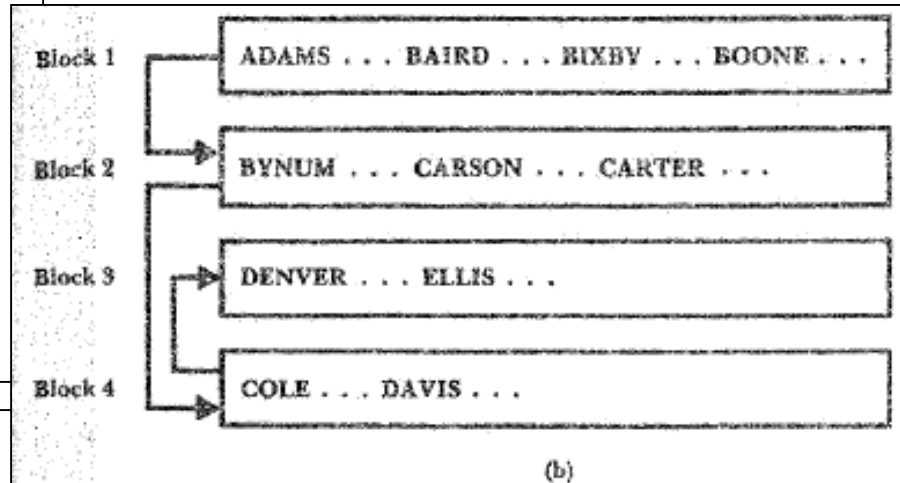
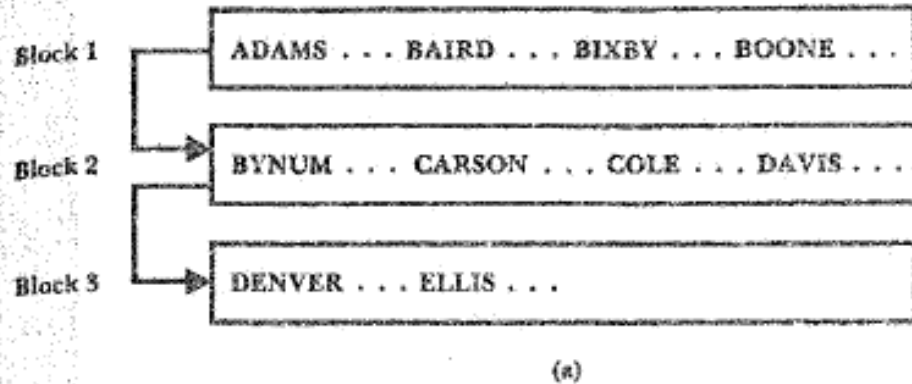


Mantendo um Sequence Set

- **Opção:** reordenação de todo o arquivo, o tempo todo? **Fora de cogitação!** Na verdade, precisamos, de alguma forma, deixar as modificações localizadas.
- **Solução:** uso de blocos.
- **Blocos de registros:** o bloco é a unidade básica de Entrada/Saída, e o tamanho dos buffers de E/S utilizados é igual ao tamanho de bloco escolhido.
- Este conceito de **blocos de registros** pode ser utilizado para manter a ordem em um Sequence Set. Como?

Mantendo um Sequence Set

Carter/Inserção



Overflow: Divisão

Underflow: Concatenação/Redistribuição

FIGURE 9.1 Block splitting and concatenation due to insertions and deletions in the sequence set. (a) Initial blocked sequence set. (b) Sequence set after insertion of CARTER record—block 2 splits, and the contents are divided between blocks 2 and 4. (c) Sequence set after deletion of DAVIS record—block 4 is less than half full, so it is concatenated with block 3.

Davis/Remoção



Mantendo um Sequence Set

- O arquivo é visto como um conjunto de blocos de registros, sendo que os blocos são encadeados em sequência: cada bloco possui um campo de ligação, que mantém um "ponteiro" para o bloco anterior e posterior (na fig. só o posterior).
- Esse ponteiro é necessário porque os blocos não estão necessariamente fisicamente adjacentes.



Mantendo um Sequence Set

- Como em árvores-B, a inserção de novos registros pode provocar o overflow de um bloco, e a remoção de registros pode provocar underflow do bloco.
- Assim, podemos utilizar: concatenação e divisão de blocos, bem como redistribuição de chaves em blocos.
 - Diferença: não há promoção na divisão (uma vez que todos os blocos estão no mesmo nível).



Mantendo um Sequence Set

- Assim, mantendo a separação dos registros em blocos, junto com as operações de divisão, concatenação e redistribuição, podemos manter um Sequence Set em ordem de chave sem ter que ordenar o conjunto inteiro de registros.
- Custos
 - Fragmentação interna dentro dos blocos.
 - A ordem dos registros não é fisicamente sequencial em todo o arquivo. Só é garantida a sequência dentro do bloco.



Escolha do tamanho do bloco

- Quando trabalhamos com nossa sequência de registros, um bloco é a unidade básica para as operações de E/S, e o conteúdo de um bloco é sempre um conjunto de registros fisicamente contíguos.
- Isso sugere que quanto maior o tamanho dos blocos, melhor, pois isso evitaria seeks.
 - Nesse caso, porque não o arquivo todo em um único bloco?



Escolha do tamanho do bloco

- Existem várias considerações a serem feitas:
 - Blocos grandes acessam grandes quantidades de informação sequencialmente, mas precisamos ser capazes de manter vários blocos na memória (porquê?), e talvez não tenhamos RAM suficiente.
 - Além disso, o tempo de leitura/escrita de um bloco não deve ser grande demais:
 - Não queremos ter de ler o arquivo inteiro para acessar um único registro, por exemplo.
 - Se o tempo de leitura de um bloco grande for maior que o tempo que gastaríamos para fazer vários seeks no bloco (para localizar um registro, por exemplo), então não interessa.



Escolha do tamanho do bloco

- Conceitos conhecidos:
 - Tamanho de bloco = tamanho de cluster. O cluster garante contiguidade física: todos os registros do cluster são acessados com um único seek. Mas e se você estiver configurando um disco para uma aplicação em particular e puder escolher o tamanho do cluster?
 - Se você está trabalhando com um disco que não é organizado por setores, mas por blocos cujo tamanho você pode escolher, uma sugestão seria fazer o tamanho de um bloco = tamanho de uma trilha (considerando memória, etc.).
- Assim, é importante tomar uma decisão informada, e não "no chute". Isso significa levar em consideração questões relevantes relativas ao tamanho do cluster, tamanho da RAM, etc.



Adição de um Índice Simples ao Sequence Set

- Mas, e o acesso indexado a um registro, dada a sua chave?
- Criamos um mecanismo para manter um conjunto de registros de forma que eles possam ser acessados sequencialmente, por ordem de chave.
- A ideia é agrupar os registros em blocos, e manter os blocos, a medida que registros são inseridos e removidos, através de divisão e concatenação de blocos, e redistribuição de registros.



Adição de um Índice Simples ao Sequence Set

- Podemos enxergar cada bloco logicamente como contendo um intervalo de registros.
- Esta é uma visão externa dos blocos: ainda não lemos os blocos, e não sabemos o que eles contém, mas é suficiente para sabermos, por exemplo, em que bloco pode estar um determinado registro que estamos procurando (BURNS, por exemplo, só pode estar no segundo bloco).
 - Ver Figura 9.2 a seguir.



Adição de um Índice Simples ao Sequence Set

- É fácil ver como podemos manter um índice simples para esses blocos:
 - Podemos, por exemplo, manter um índice com entradas de tamanho fixo, onde cada entrada corresponde a um bloco e informa o valor da última chave contida no bloco.

Adição de um Índice Simples ao Sequence Set



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

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



Adição de um Índice Simples ao Sequence Set

- A combinação deste índice com os blocos permite acesso sequencial indexado:
 - Se desejamos recuperar um registro em particular, consultamos o índice e, a partir dele, recuperamos o bloco que o contém, se ele existir.
 - Se desejamos acesso sequencial, começamos no primeiro bloco e vamos lendo a lista encadeada de blocos até percorrer todos os blocos.



Adição de um Índice Simples ao Sequence Set

- Este índice admite inclusive Busca Binária, o que é eficiente se o índice estiver em RAM (caso contrário, muitos acessos deverão ser realizados).
- O índice precisa ser atualizado a medida que os blocos mudam com divisões, concatenações e redistribuições. Isso é relativamente fácil se o índice estiver em memória.

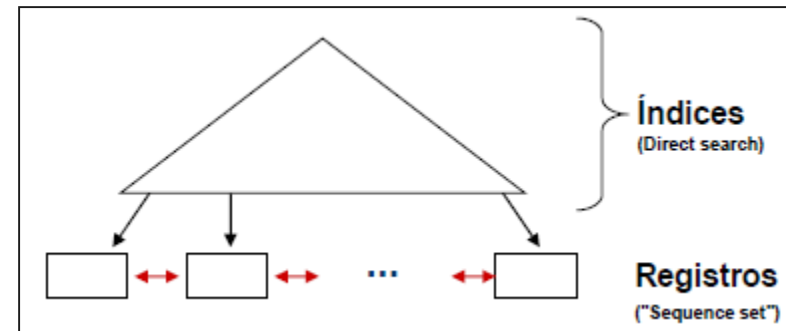
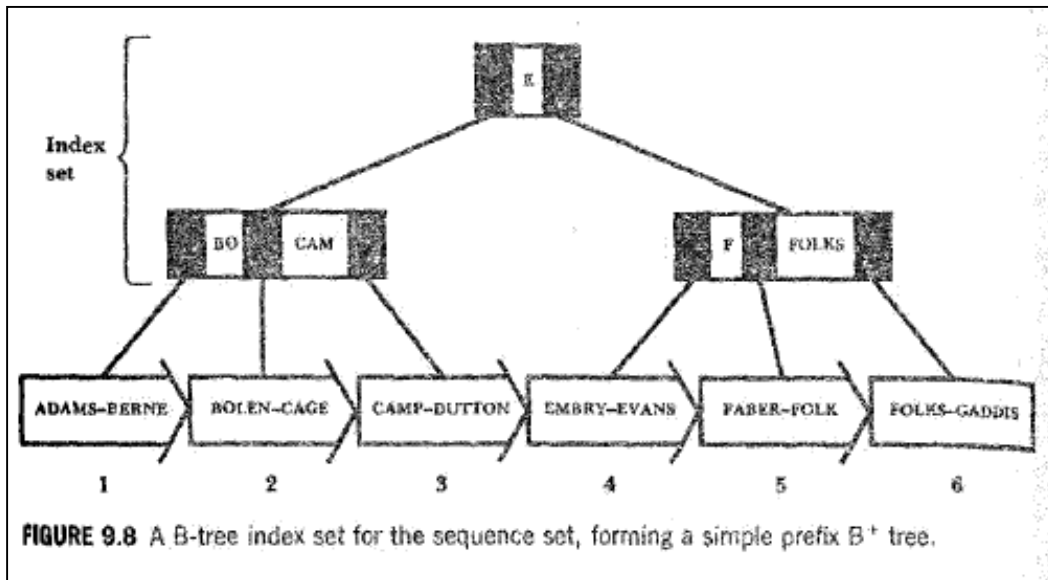


Adição de um Índice Simples ao Sequence Set

- Mas e se o índice não cabe inteiro na memória principal?
 - Árvores-B⁺

Árvores-B⁺

- As árvores-B⁺ são uma estrutura híbrida que combinam o uso de Sequence Set e árvores-B.
 - As árvores-B armazenam os índices que nos guiam até o Sequence Set.
 - O Sequence Set armazena os registros.

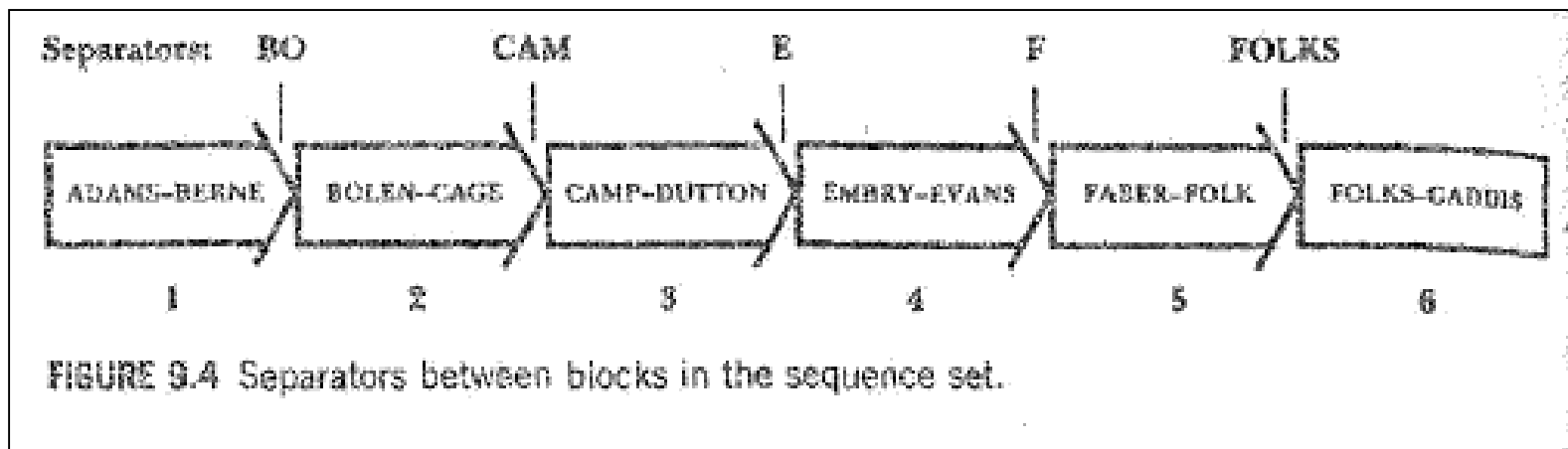




Conteúdo do Índice: Separadores, ao invés de Chaves

- **Precisamos manter as chaves no índice?**
- O índice por si só não oferece a informação que estamos procurando, mas atua como um mapa através do qual podemos localizá-la.
- Diante disso, não há necessidade das chaves propriamente ditas aparecerem no índice: nossa necessidade real é de **separadores**.
- Uso de separadores ao invés de chaves tem a vantagem de permitir o uso de separadores curtos e de tamanho variável.

Conteúdo do Índice: Separadores, ao invés de Chaves



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

Conteúdo do Índice: Separadores, ao invés de Chaves

Quem pode ser separador?

FIGURE 9.5 A list of potential separators.

CAMP-DUTTON

3

DUTU
DVXGHESJF
DZ
E
EBQX
ELEEMOSYNARY

EMBRY-EVANS

4

Conteúdo do Índice: Separadores, ao invés de Chaves

Quem pode ser separador?

FIGURE 9.5 A list of potential separators.

CAMP-D

```
/* find_sep(key1,key2,sep) ...  
    finds shortest string that serves as a separator between key1 and  
    key2. Returns this separator through the address provided by  
    the "sep" parameter  
    the function assumes that key2 follows key1 in collating sequence  
*/  
find_sep(key1,key2,sep)  
    char key1[], key2[], sep[];  
{  
    while ( (*sep++ = *key2++) == *key1++)  
        ;  
    *sep='\0';    /* ensure that separator string is null terminated */  
}
```

FIGURE 9.6 C function to find a shortest separator.

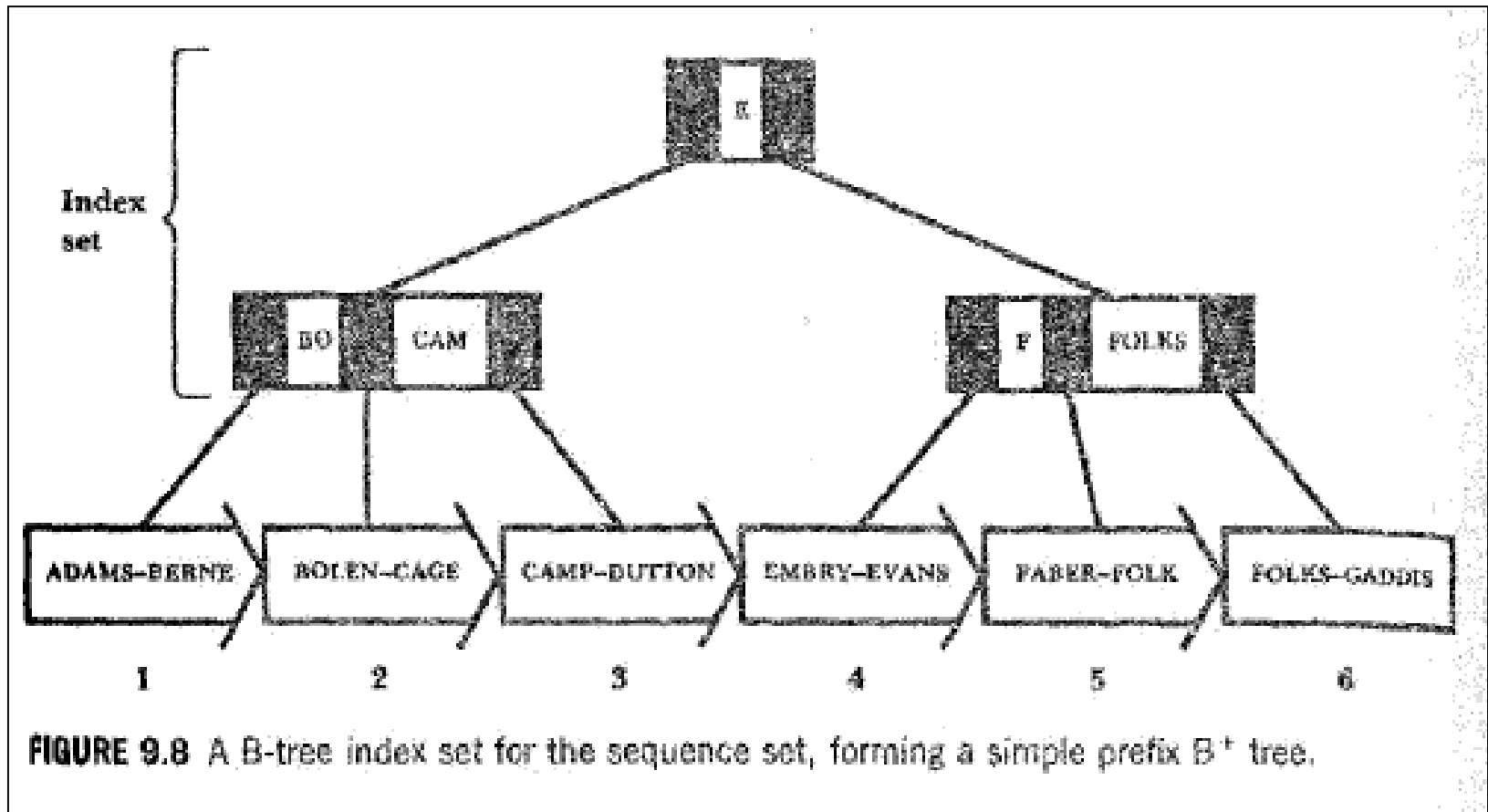


Árvore-B⁺ Prefixada Simples (Simple Prefix B⁺-Tree)

- Quando utilizamos separadores ao invés de chaves inteiras, como nos índices mantidos como árvores-B, chamamos a estrutura de Árvore-B⁺ Prefixada Simples.
- O termo prefixada simples indica que o índice contém os separadores mais curtos (mais simples), ou os prefixos das chaves ao invés de cópias das chaves.

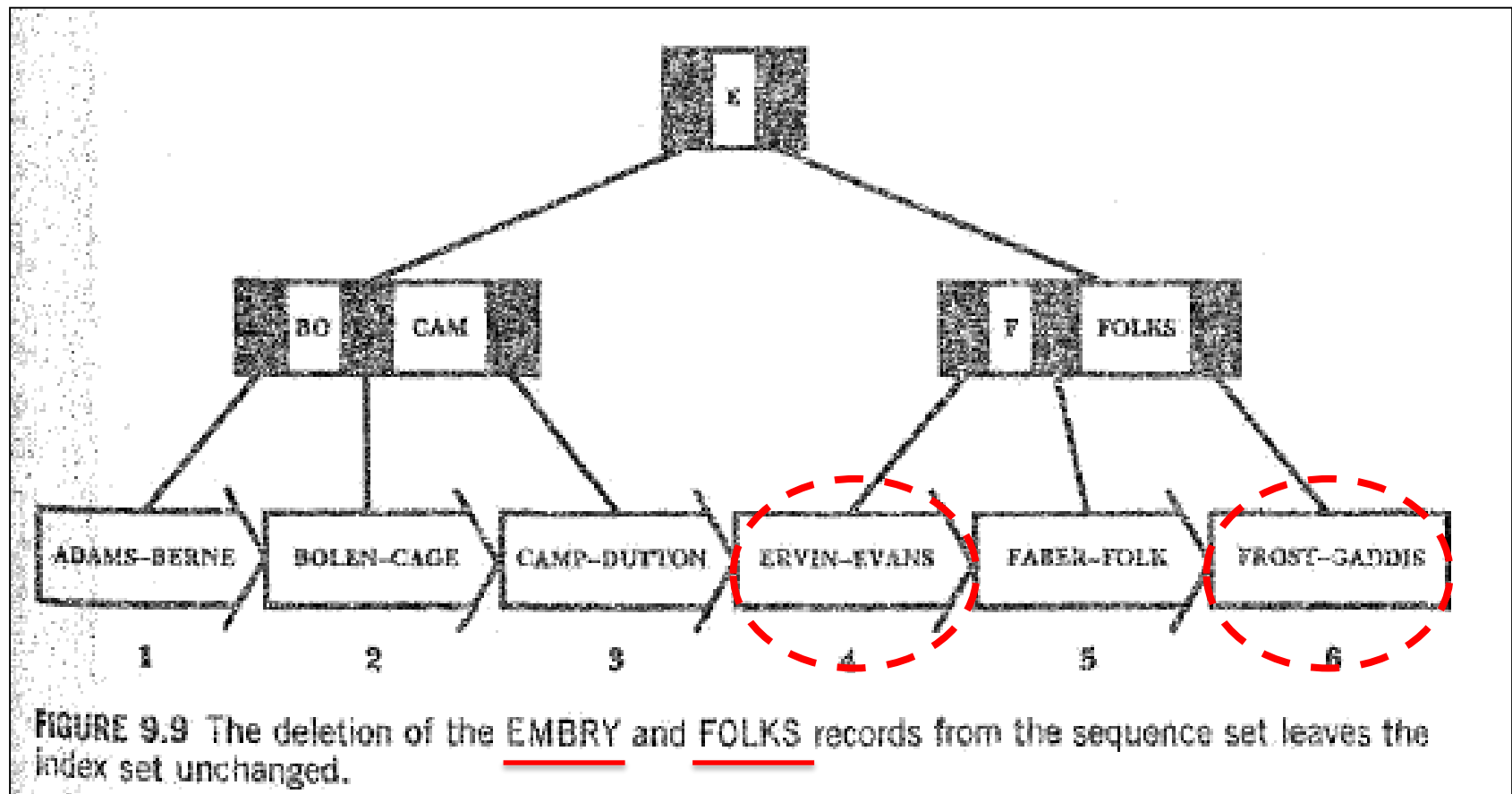
Árvore-B⁺ Prefixada Simples

Pesquisar a chave EMBRY



Manutenção da Árvore-B⁺ Prefixada

Deleção de chaves no Sequence Set





Manutenção da Árvore-B⁺ Prefixada

- Que efeitos essas deleções têm no Index Set?
 - Uma vez que o número de blocos no Sequence Set não é alterado, e uma vez que nenhum registro é movido entre blocos, o Index Set permanece inalterável.



Manutenção da Árvore-B⁺ Prefixada

- O efeito de inserções de novos registros no Sequence Set que não causam a divisão de blocos produz o mesmo efeito das deleções que não causam concatenação: o Index Set permanece inalterado.
- O que acontece quando a inserção e a deleção de registros no Sequence Set altera o número de blocos no Sequence Set?



Manutenção da Árvore-B⁺ Prefixada

- Se tivermos mais blocos, precisaremos de mais separadores; se tivermos menos blocos, precisaremos de menos separadores.
 - É necessário alterar o Index Set.
- Uma vez que o Index Set é uma árvore-B, as mudanças no Index Set são manipuladas de acordo com as regras de inserção e deleção das árvores-B.

Manutenção da Árvore

Inserção

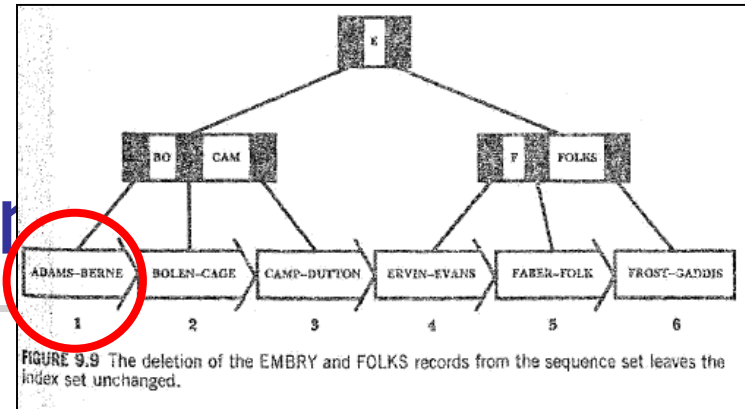
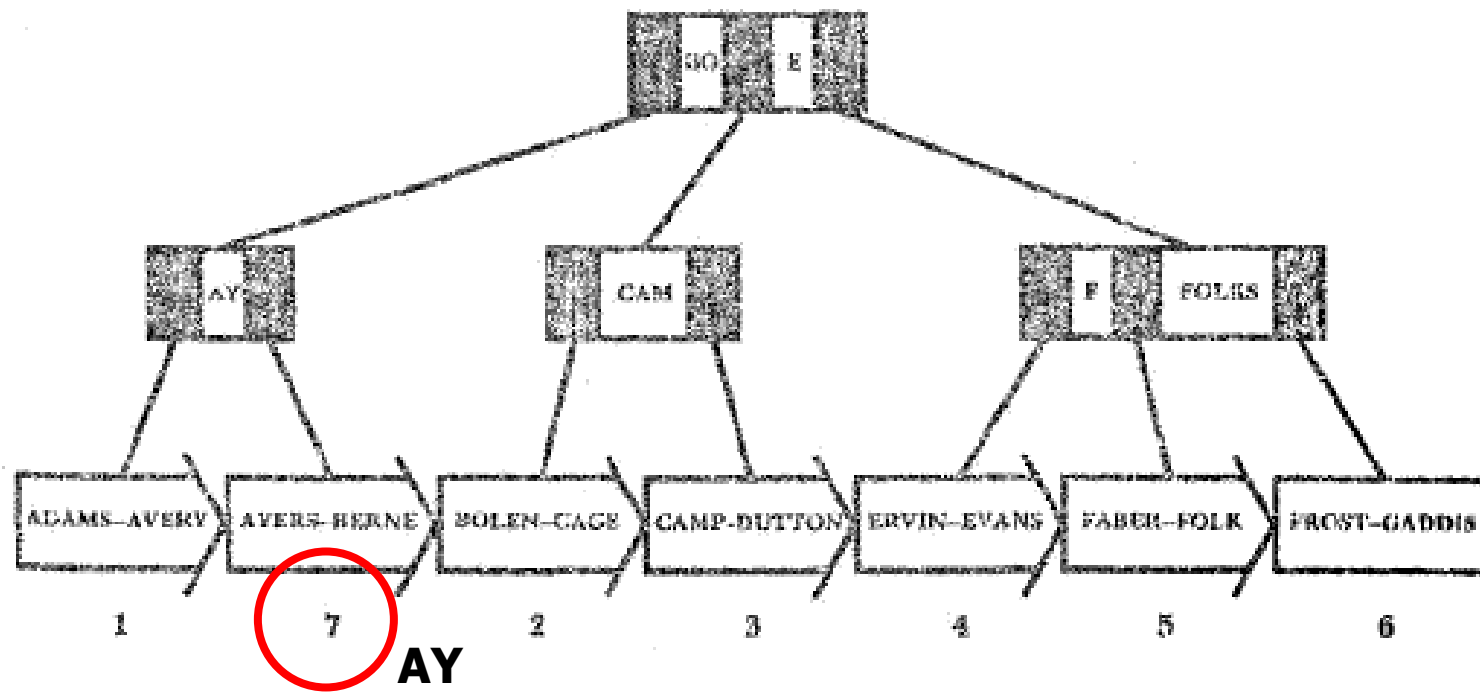
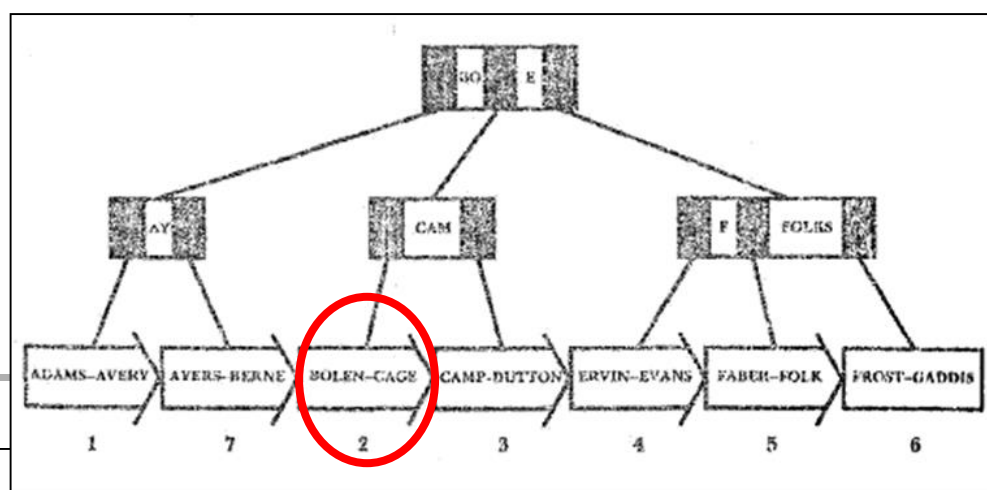


FIGURE 9.10 An insertion into block 1 causes a split and the consequent addition of block 7. The addition of a block in the sequence set requires a new separator in the index set. Insertion of the AY separator into the node containing BO and CAM causes a node split in the index set B-tree and consequent promotion of BO to the root.



Manutenção da



Remoção

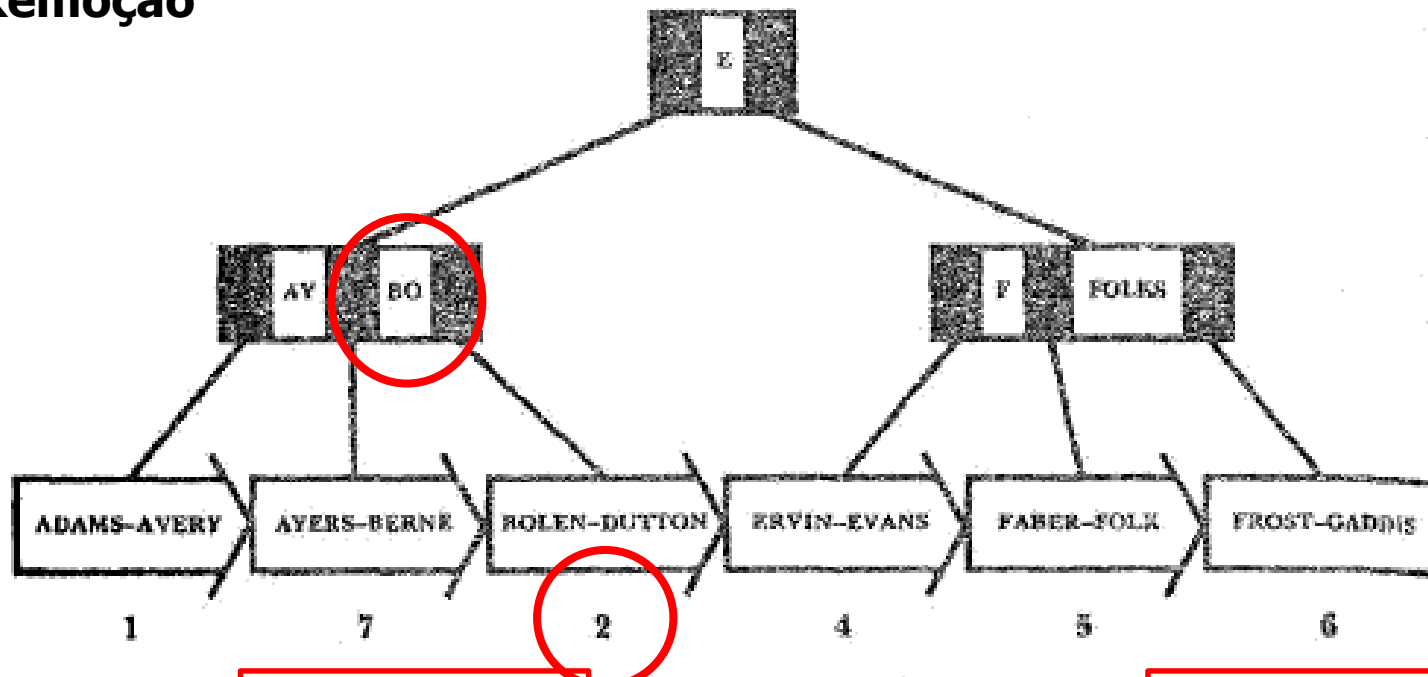


FIGURE 9.11 A deletion from block 2 causes underflow and the consequent concatenation of blocks 2 and 3. After the concatenation, block 3 is no longer needed and can be placed on an avail list. Consequently, the separator CAM is no longer needed. Removing CAM from its node in the index set forces a concatenation of index set nodes, bringing BO back down from the root.



Manutenção da Árvore-B⁺ Prefixada

- É importante mencionar que a ocorrência de overflow/underflow dos nós do índice não acompanha a ocorrência de overflow/underflow no Sequence Set.
- Assim, uma divisão/concatenação de blocos no Sequence Set não resulta necessariamente em uma divisão/concatenação de nós do índice.
- O índice é gerenciado como uma árvore-B: inserções e remoções no índice funcionam como as operações correspondentes em árvores-B.



Manutenção da Árvore-B⁺ Prefixada

- Resumindo:

- Inserção e deleção de registros sempre ocorrem no Sequence Set, uma vez que é onde os registros se encontram.
- Se operações de divisão, concatenação, ou redistribuição forem necessárias, realize-as não se preocupando com o Index Set.



Manutenção da Árvore-B⁺ Prefixada

- Resumindo:
 - Uma vez que as operações no Sequence Set estiverem concluídas, faça as alterações necessárias no Index Set:
 - Se blocos foram separados, um novo separador deve ser inserido no Index Set.
 - Se blocos foram concatenados, um separador deve ser removido do Index Set.
 - Se registros foram distribuídos entre blocos, um separador do Index Set precisa ser modificado.



Tamanho do Bloco do Index Set

- O tamanho físico de um nó no Index Set é normalmente o mesmo que o tamanho físico escolhido para os blocos no Sequence Set.
- Razões:
 - O tamanho do bloco para o Sequence Set é usualmente escolhido em função das características do disco e da quantidade de memória disponível. A escolha para o tamanho do nó leva em consideração os mesmos fatores; assim, o melhor tamanho para um é o melhor tamanho para o outro.



Tamanho do Bloco do Index Set

- Razões:
 - Escolhendo o mesmo tamanho fica mais fácil implementar uma árvore-B⁺ prefixada virtual.
 - Escolhendo o mesmo tamanho podemos misturar em um mesmo arquivo Sequence Set e Index Set, evitando trabalhar com arquivos separados e facilitando a implementação.



Index Set:

Estrutura Interna de um Nó

- Até o momento consideramos que um nó contém um número fixo de separadores
 - Não faz sentido então utilizar separadores de tamanho variável!!!
- É necessário que cada nó armazene um número variável de separadores de tamanho variável.
 - Como seria a busca através desses separadores variáveis?
 - É interessante podermos realizar busca binária!!!
 - É necessário criar um “índice” dentro do próprio índice!!!

Index Set:

Estrutura Interna de um Nó

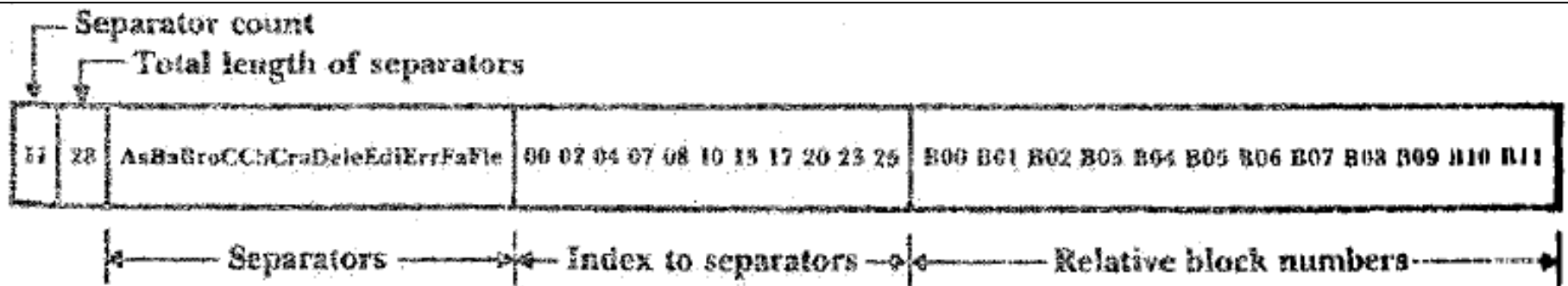


FIGURE 9.13 Structure of an index set block.

Index Set:

Estrutura Interna de um Nó

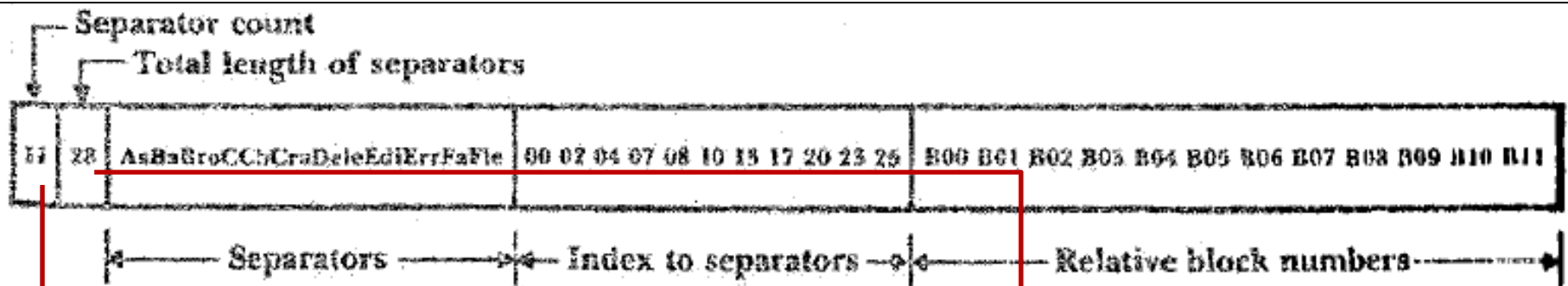


FIGURE 9.13 Structure of an index set block.

Utilizado para encontrar o elemento do meio no índice dos separadores para iniciar a busca binária

Utilizado também para encontrar o fim do índice

Utilizado para encontrar onde o índice se inicia, uma vez que os separadores possuem tamanho variável

Index Set:

Estrutura Interna de um Nó

- Se existem N separadores dentro de um bloco, o bloco possui $N+1$ filhos.
- Agora, um nó dentro de um Index Set é de ordem variável, uma vez que cada nó contém um número variável de separadores.

Index Set:

Estrutura Interna de um Nó

- Essa variabilidade possui as seguintes implicações:
 - O número de separadores em um bloco é limitado diretamente pelo tamanho do bloco do que por uma ordem pré-determinada (como na árvore-B).
 - Uma vez que a árvore possui ordem variável, operações para determinar quando um nó está cheio, ou meio cheio não é tão simples. Decisões sobre quando dividir, concatenar e redistribuir se tornam mais complicadas.



Árvores-B⁺

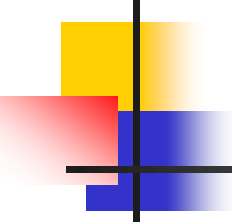
- As árvores-B⁺ prefixadas simples são uma variação das árvores-B⁺:
 - A diferença entre ambas é que a última não usa separadores no índice (prefixos), mas sim cópias das chaves.
- Vantagem:
 - Nas árvores-B⁺ prefixadas utiliza-se campos de tamanho variável para lidar com separadores de diversos tamanhos. Para algumas aplicações, o custo extra requerido para manter e usar essas estruturas de tamanho variável ultrapassa os benefícios de se usar separadores compactos. Nesses casos, compensa escolher a construção direta de árvores-B⁺ usando cópias de chaves de tamanho fixo como separadores do Sequence Set.



Árvores-B⁺

- Vantagem:

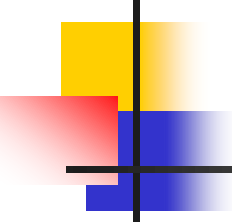
- Alguns conjuntos de chaves não apresentam muita compressão quando o método do prefixo simples é usado para produzir separadores.
Exemplo: 34C18K756, 34C18K757, 34C18K758. Se a altura da árvore (deve-se realizar o cálculo) se mantiver aceitável com o uso das cópias das chaves como separadores, a opção sem compressão deve ser escolhida.



Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Semelhanças

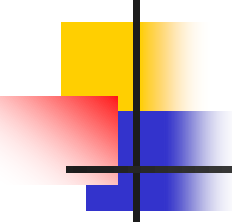
- Todas são estruturas de índice paginadas, o que significa que elas trazem blocos inteiros de informação para a RAM de uma única vez.
 - Poucos seeks são necessários para encontrar uma chave entre milhares de chaves.
- Todas as abordagens mantêm a árvore balanceada.
- Em todas as abordagens a árvore é construída de baixo para cima (bottom-up).



Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Semelhanças

- Em todas as abordagens é possível aumentar a taxa de utilização dos nós através da redistribuição e do two-to-three split.
- Todas as abordagens podem ser implementadas como estruturas de árvores virtuais.
- Qualquer uma das abordagens pode ser adaptada para o uso com registros de tamanho variável usando estruturas dentro dos nós similares á vista anteriormente.

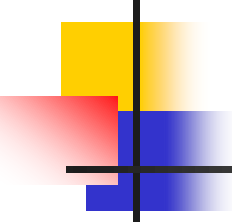


Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Diferenças

- Árvores-B

- Contém informações que são agrupadas como um conjunto de pares. Um membro de cada par é uma chave; o outro membro é a informação associada.
 - Esses pares estão distribuídos por todos os nós da árvore.
 - Assim, uma informação desejada pode ser encontrada em qualquer nível da árvore.
 - Diferente das árvores-B⁺ e árvores-B⁺ prefixada que requerem que todas as buscas desçam até o nível mais baixo da árvore, onde se encontra o Sequence Set.

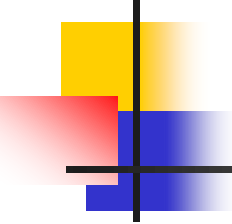


Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Diferenças

- Árvores-B

- Uma escrita ordenada das chaves deve ser feita por meio de um atravessamento em-ordem, o que requer um grande número de seeks.
 - Nas árvores-B⁺ e árvores-B⁺ prefixada a escrita é feita percorrendo-se sequencialmente o Sequence Set.
 - Entretanto, pode-se utilizar árvore-B virtual para melhorar o desempenho do atravessamento.

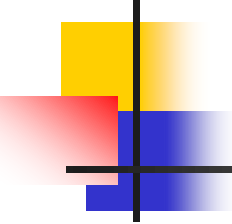


Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Diferenças

- Árvores-B⁺

- A principal diferença das árvores-B⁺ para as árvores-B é que nas árvores-B⁺ todas as chaves e informações dos registros estão contidas em um conjunto de blocos linkados chamado Sequence Set.
 - A chave e a informação do registro não estão em nenhuma das partes dos níveis superiores da árvore-B⁺.
 - O acesso indexado ao Sequence Set se dá através de uma estrutura chamada Index Set.
 - Em uma árvore-B⁺ o Index Set consiste de cópias das chaves que representam os limites entre os blocos do Sequence Set.
 - Essas cópias das chaves são chamadas de separadores, uma vez que elas separam um bloco do Sequence Set de seu predecessor.

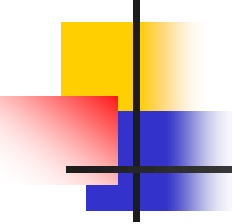


Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Diferenças

- Árvores-B⁺

- As duas vantagens que as árvores-B⁺ oferecem em relação as árvores-B são:
 - O Sequence Set pode ser processado de forma sequencial, provendo acesso eficiente aos registros por ordem de chave;
 - O uso de separadores, ao contrário de registros inteiros, no Index Set implica que o número de separadores que podem ser colocados em um nó em uma árvore-B⁺ excede o número de registros que podem ser colocados em um nó de tamanho equivalente em uma árvore-B.
 - Gera uma árvore mais larga e mais rasa.



Árvores-B, Árvores-B⁺, Árvores-B⁺ Prefixada

- Diferenças

- Árvores-B⁺ Prefixada

- Os mesmos que o da árvore-B⁺ em relação as árvores-B, com a vantagem de conseguir produzir árvores ainda mais largas e rasas em comparação com as árvores-B⁺, uma vez que mais separadores podem ser armazenados em um nó.
 - Entretanto, há um preço a se pagar, já que teremos que trabalhar com campos de tamanhos variáveis.
 - Atenção com os algoritmos de inserção/remoção.