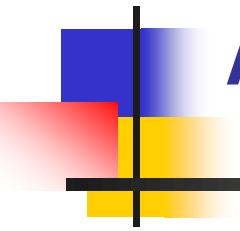


Árvores-B (Parte IV)

Leandro C. Cintra

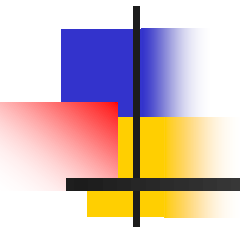
M.C.F. de Oliveira

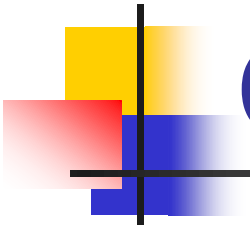
2004

Fonte: Folk & Zoelick, File Structures

http://www.icmc.sc.usp.br/~sce183

Definição e Propriedades de árvores-B





Ordem

- A ordem de uma árvore-B é dada pelo número máximo de descendentes que uma página, ou nó, pode possuir
- Em uma árvore-B de ordem **m**, o número máximo de chaves em uma página é **m-1**
- **Exemplo:**
  - Uma árvore-B de ordem 8 tem, no máximo, 7 chaves por página.

Número mínimo de chaves por página

- Quando uma página é sub-dividida na inserção, as chaves são divididas (quase) igualmente entre as páginas velha e nova. Deste modo, o número mínimo de chaves em um nó é dado por $\lceil m/2 \rceil - 1$ (exceto para a raiz)
- **Exemplo:** árvore B de ordem 8, armazena no máximo 7 chaves por página e tem, no mínimo, 3 chaves por página
- **Nós folha:** alocados no nível mais baixo da árvore

Definição formal das Propriedades de árvores-B

- Para uma árvore-B de ordem **m**
  - **1.** cada página tem, no máximo, **m** descendentes
  - **2.** cada página, exceto a raiz e as folhas, tem no mínimo $\lceil \mathbf{m/2} \rceil$ descendentes
  - **3.** a raiz tem, no mínimo, dois descendentes - a menos que seja uma folha
  - **4.** todas as folhas estão no mesmo nível
  - **5.** uma página não folha que possui **k** descendentes contém **k-1** chaves
  - **6.** uma página folha contém, no mínimo $\lceil \mathbf{m/2} \rceil$ **-1** e, no máximo, **m-1** chaves

Profundidade da Busca no Pior Caso

- Ex. armazena-se 1.000.000 chaves, utilizando uma árvore-B de ordem 512
  - **qual a altura máxima que a árvore pode atingir?**
- O pior caso ocorre quando cada página tem apenas o número mínimo de descendentes, e a árvore possui, portanto, altura máxima e largura mínima

Profundidade da Busca no Pior Caso

- o número mínimo de descendentes para a página raiz é 2,sendo que cada uma destas páginas, por sua vez, possui no mínimo ⌈**m/2**⌉ descendentes
- O segundo nível, por sua vez, possui no mínimo 2* ⌈**m/2**⌉ descendentes
- Em geral, para um nível **d** da árvore, o número mínimo de descendentes é dado por

  2*⌈m/2⌉^(d-1)

Profundidade da Busca no Pior Caso

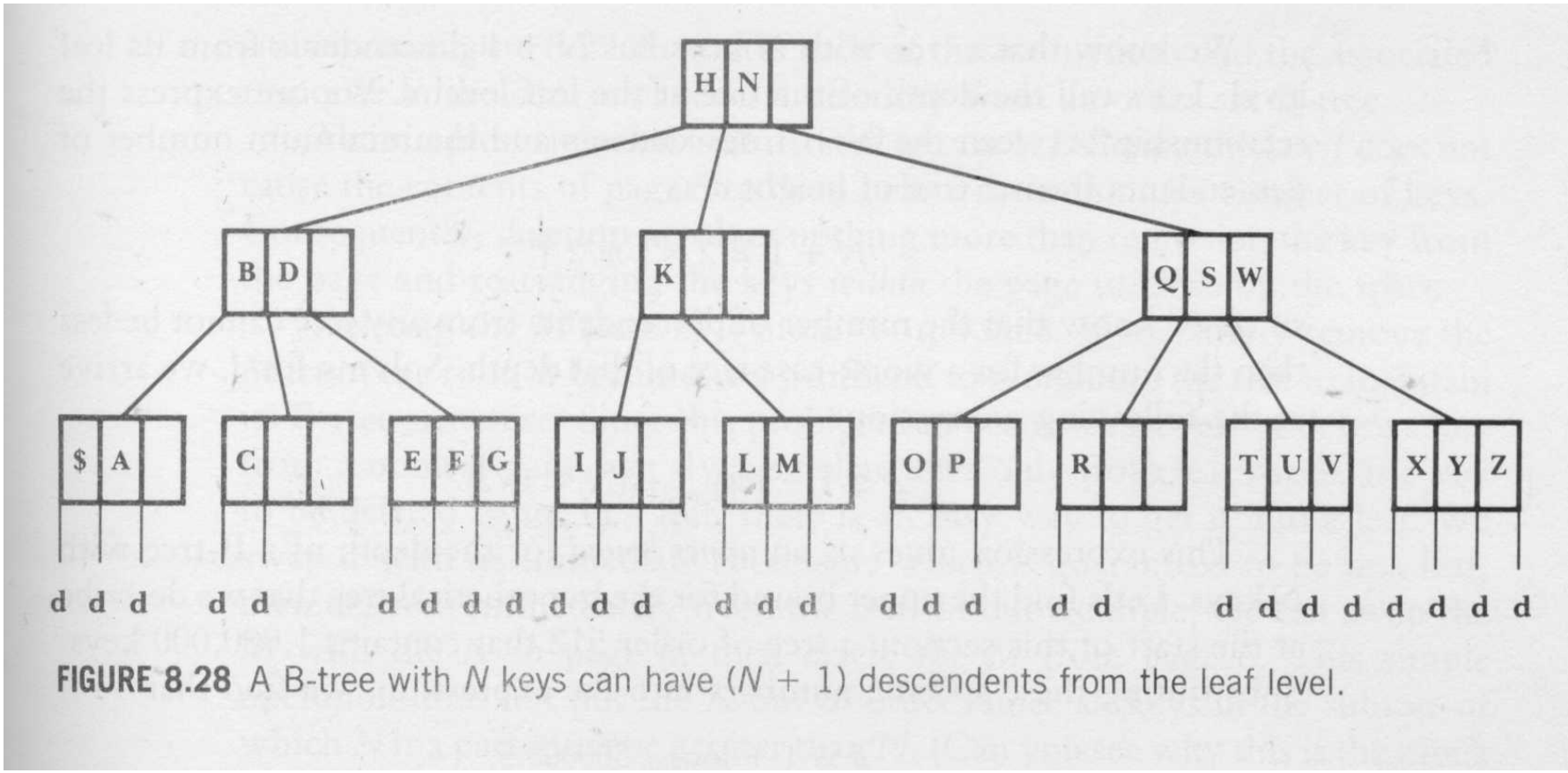


FIGURE 8.28 A B-tree with *N* keys can have (*N* + 1) descendents from the leaf level.

Profundidade da Busca no Pior Caso

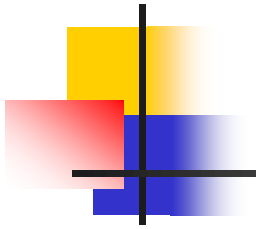
- Uma árvore de **N** chaves tem **N + 1** potenciais descendentes a partir de seu nível mais inferior, das folhas
- Seja **d** a profundidade da árvore no nível das páginas folhas. Podemos expressar a relação entre os **N + 1** descendentes e o número mínimo de descendentes de uma árvore de altura **d** como:
  - $\mathbf{d <= 1 + \log_{\lceil m/2 \rceil} \{(N+1)/2\}}$

Profundidade da Busca no Pior Caso - Exemplo

- **$d <= 1 + \log_{\lceil m/2 \rceil} \{(N+1)/2\}$**

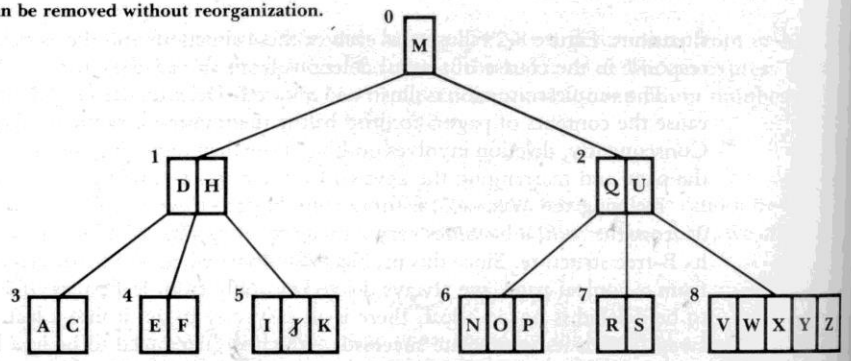
- para a árvore-B de ordem 512 com 1.000.000 de chaves, tem-se
  - $d <= 1 + \log_{256} (500.000,5) <= 3.37$
  - nossa árvore terá altura 3, no máximo, e no máximo 3 acessos serão necessários para localizar uma chave, o que é um desempenho muito bom

Eliminação, Redistribuição e Concatenação

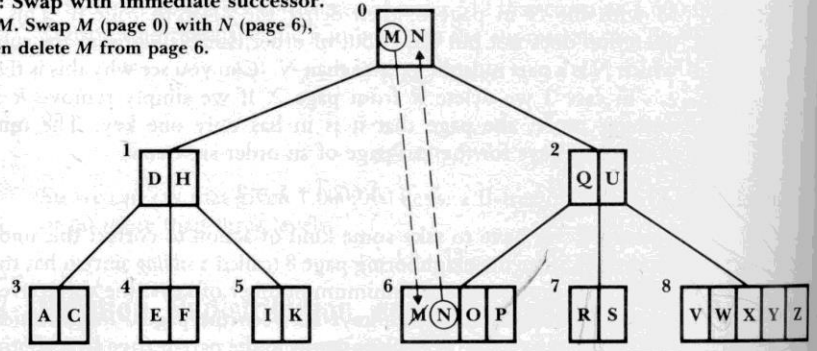


- O ***split*** garante a manutenção das propriedades da árvore-B durante a inserção
- Essas propriedades precisam ser mantidas, também, durante a eliminação de chaves

Case 1: No action.
Delete *J* from page 5. Since page 5 has more than the minimum number of keys, *J* can be removed without reorganization.



Case 2: Swap with immediate successor.
Delete *M*. Swap *M* (page 0) with *N* (page 6), and then delete *M* from page 6.



Case 3: Redistribution.
Delete *R*. Underflow occurs. Redistribute keys among pages 2, 7, and 8 to restore balance between leaves.

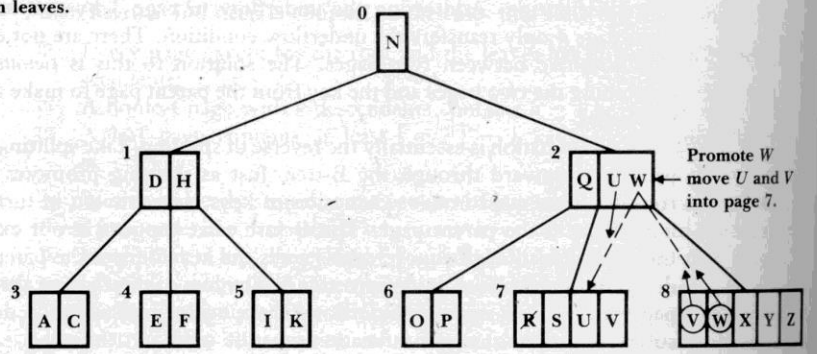
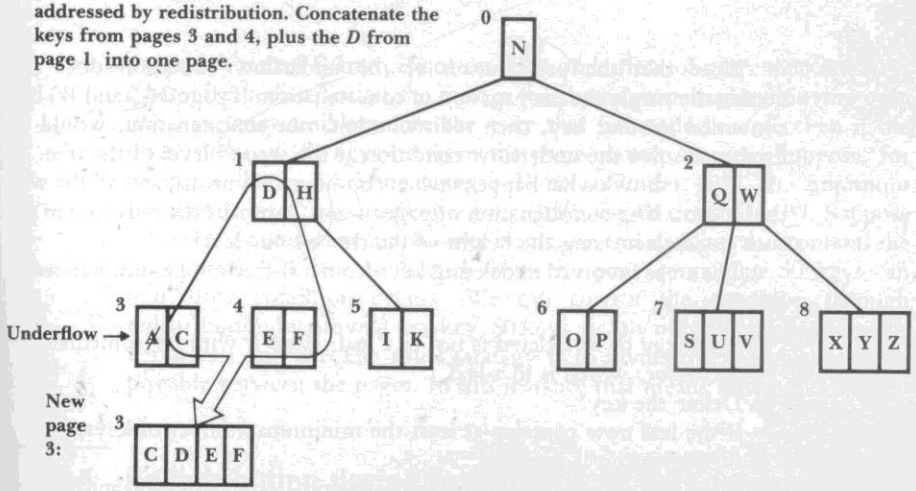
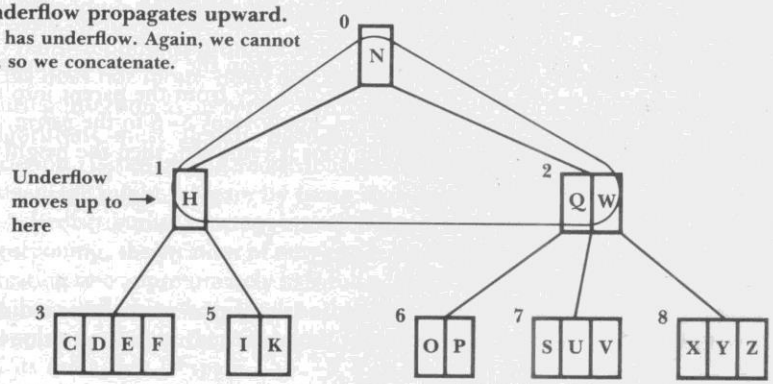


FIGURE 8.29 Six situations that can occur during deletions.

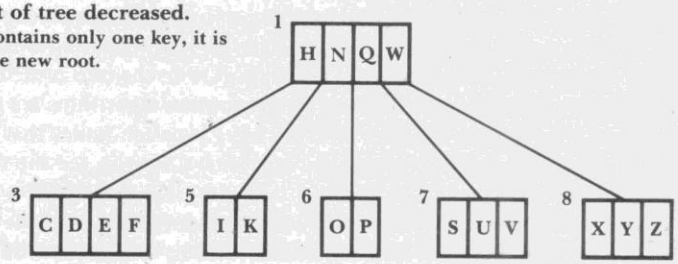
Case 4: Concatenation.
Delete *A*. Underflow occurs, but it cannot be addressed by redistribution. Concatenate the keys from pages 3 and 4, plus the *D* from page 1 into one page.

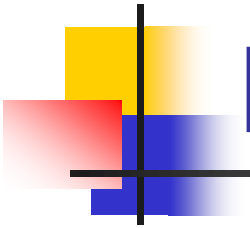


Case 5: Underflow propagates upward.
Now page 1 has underflow. Again, we cannot redistribute, so we concatenate.



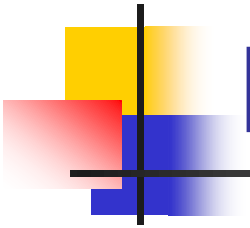
Case 6: Height of tree decreased.
Since the root contains only one key, it is absorbed into the new root.





Eliminação: Caso 1

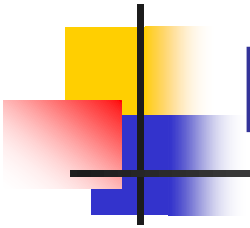
- **Caso 1:** eliminação de uma chave em uma página folha, sendo que o número mínimo de chaves na página é respeitado
- **Solução:** Chave é retirada e os registros internos à página são reorganizados



Eliminação: Caso 2

- **Caso 2:** eliminação de uma chave que não está numa folha

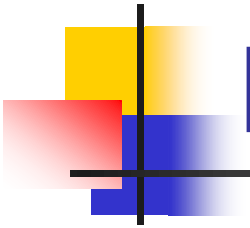
- **Solução:** Sempre eliminamos de páginas folha
  - Se uma chave deve ser eliminada de uma página que não é folha, trocamos a chave com sua sucessora imediata (ou com a predecessora imediata), que está numa folha. A seguir, eliminamos a chave da folha.



Eliminação: Caso 3

- **Caso 3:** eliminação causa *underflow* na página.

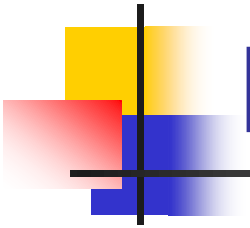
- **Solução: <u>Redistribuição</u>**
  - Procura-se uma página irmã (mesmo pai) que contenha mais chaves do que o mínimo: se existir, redistribui-se as chaves entre essas páginas
  - A redistribuição pode provocar uma alteração na chave separadora, que está no nó pai.



Eliminação: Caso 4

- **Caso 4:** ocorre *underflow* e a redistribuição não pode ser aplicada
  - Não existem chaves suficientes para dividir entre as duas páginas irmãs

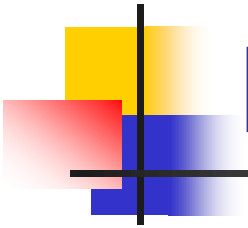
- **Solução:** Deve-se utilizar **concatenação**:
  - combina-se o conteúdo das duas páginas e a chave separadora da página pai para formar uma única página
  - A concatenação é o inverso do processo de subdivisão
    - Como conseqüência, pode ocorrer *underflow* da página pai.



Eliminação: Caso 5

- **Caso 5:** *underflow* da página pai

- **Solução:** utiliza-se redistribuição ou concatenação novamente



Eliminação: Caso 6

- **Caso 6:** Diminuição da altura da árvore
  - Ocorre quando o nó raiz tem uma única chave e aplica-se a concatenação nos seus nós filhos

Eliminação de chave em árvore-B

1. Se a chave não estiver numa folha, troque-a com sua sucessora

2. Elimine a chave da folha.

3. Se a folha continuar com o número mínimo de chaves, fim.

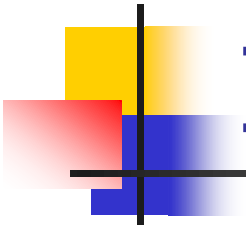
4. A folha tem uma chave a menos que o mínimo. Verifique as páginas irmãs a esquerda e a direita:

- **4.1.** se uma delas tiver mais do que o número mínimo de chaves, aplique redistribuição.
- **4.2.** senão concatene a página com uma das irmãs e a chave separadora do pai.

5. Se ocorreu concatenação, aplique os passos de 3 a 6 para a página pai.

6. Se a última chave da raiz for removida, a altura da árvore diminui.

Redistribuição durante Inserção



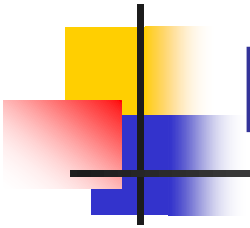
- Diferentemente da divisão e da concatenação, o efeito da redistribuição é local. Não existe propagação

- Outra diferença é que não existe regra fixa para o rearranjo das chaves
  - redistribuição pode restabelecer as propriedades da árvore-B movendo apenas uma chave de uma página irmã para a página com problema
  - Estratégia usual é redistribuir as chaves igualmente entre as páginas

Redistribuição durante Inserção

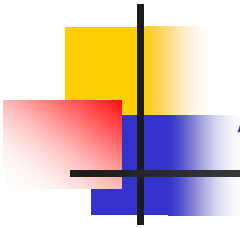
- A redistribuição é uma idéia nova, não explorada no algoritmo de inserção. Entretanto, seria uma opção desejável também na inserção

  - Ao invés de dividir uma página cheia em duas páginas novas semi-vazias, pode-se optar por colocar a chave que sobra (ou mais de uma!) em outra página. Essa estratégia resulta em uma melhor utilização do espaço alocado para a árvore



Redistribuição X Sub-divisão

- Depois da sub-divisão de uma página, cada página fica 50% vazia
  - Portanto, a utilização do espaço, no pior caso, em uma árvore-B que utiliza *splitting*, é de cerca de 50%
  - Em média, para árvores grandes, foi provado que o índice de ocupação de páginas é de ~69%
- Estudos empíricos indicam que a utilização de redistribuição pode elevar esse índice para 85%
  - Resultados sugerem que qualquer aplicação séria de árvore-B deve utilizar, de fato, redistribuição durante a inserção



Árvores-B* (*B*-trees*)

- Proposta por Knuth em 1973, essa nova organização tenta redistribuir as chaves durante a inserção antes de subdividir o nó
  - É uma variação de árvore-B na qual cada nó tem, no mínimo, 2/3 do número máximo de chaves
- A geração destas árvores utiliza uma variação do processo de subdivisão
  - a subdivisão é adiada até que duas páginas irmãs estejam cheias
  - realiza-se, então, a divisão do conteúdo das duas páginas em 3 páginas (*two-to-three split*)

two-to-three split

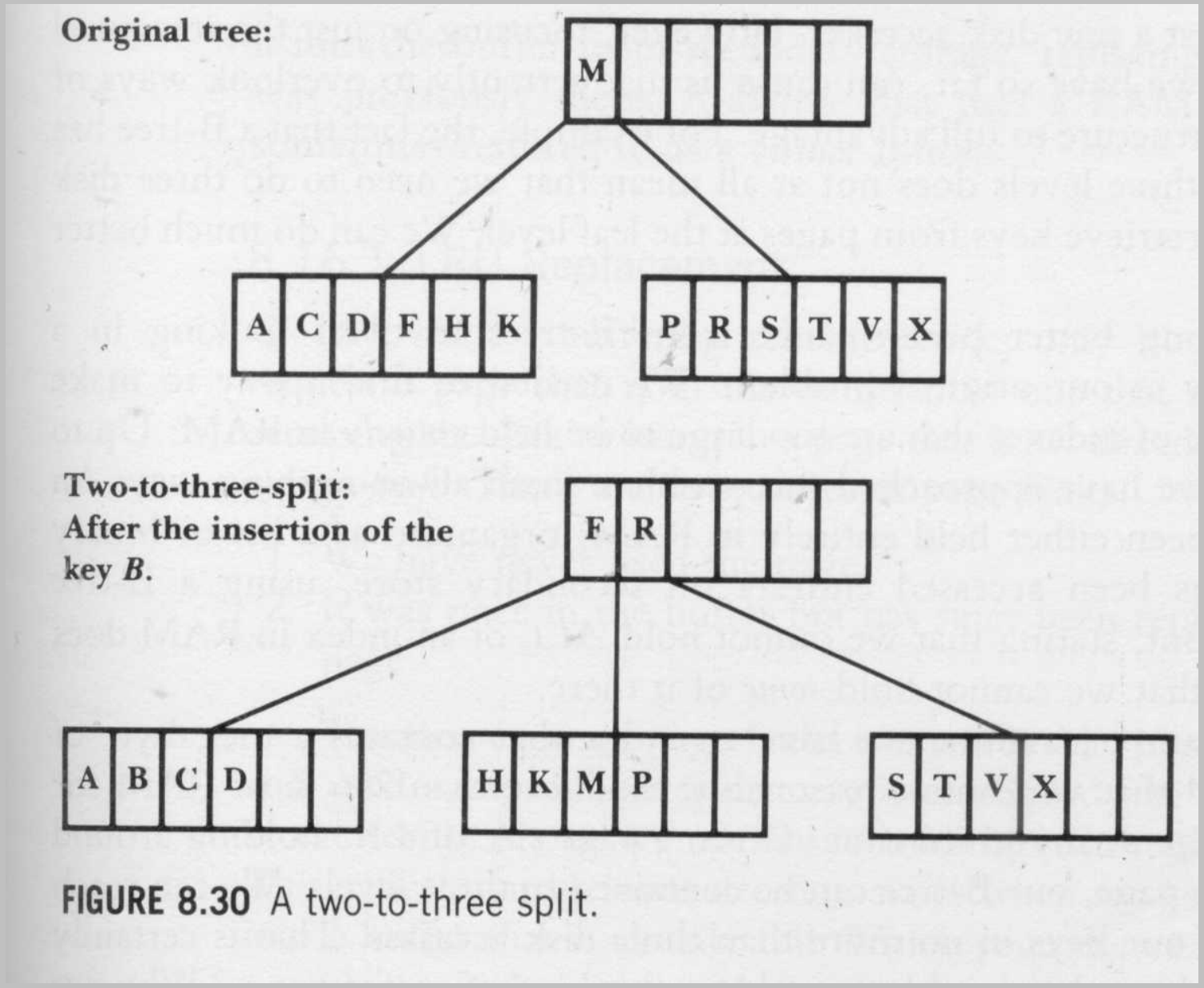
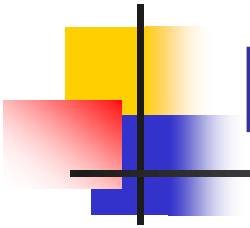
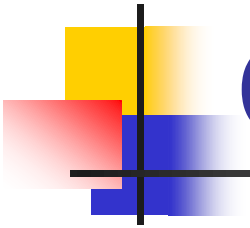


FIGURE 8.30 A two-to-three split.



Propriedades de árvores-B*

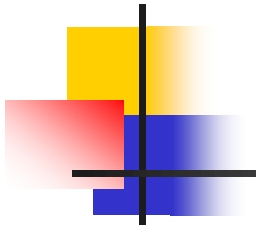
- Cada página tem no máximo **m** descendentes
- Toda página, **exceto a raiz e as folhas,** tem no mínimo **(2m-1)/3** descendentes
- A raiz tem pelo menos 2 descendentes, a menos que seja uma folha
- Todas as folhas estão no mesmo nível
- Uma página não-folha com k descendentes contém k-1 chaves
- Uma página folha contém no mínimo $\lfloor$**(2m-1)/3**$\rfloor$ e no máximo **m-1** chaves



Observações

- As principais alterações estão na segunda e na última regra
- Esta propriedade afeta as regras para remoção e redistribuição
- Deve-se tomar cuidado na implementação, uma vez que a raiz nunca tem irmã, e portanto requer tratamento especial
- Uma solução é dividir a raiz usando a divisão convencional (*one-to-two split*), outra é permitir que a raiz seja maior

Árvores-B Virtuais (*Virtual B-trees*)

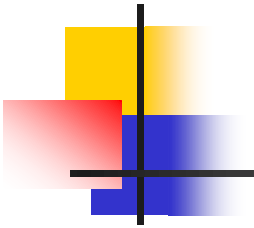


- Árvores-B são muito eficientes, mas podem ficar ainda melhores
- Observe, por exemplo, que o fato da árvore ter profundidade 3 não implica em fazer 3 acessos para recuperar as páginas folha
- O fato de não podermos manter todo o índice na RAM não significa que não se possa manter pelo menos parte dele

Árvores-B Virtuais (*Virtual B-trees*)

- **Exemplo**
  - Suponha que temos um índice que ocupa 1 MB, e que temos disponíveis 256KB de RAM
  - Supondo que uma página usa 4 KB, e armazena em torno de 64 chaves por página
  - Nossa árvore-B pode estar totalmente contida em 3 níveis
  - Podemos atingir qualquer página com, no máximo, 3 acessos a disco
  - Mas se a raiz for mantida todo o tempo na memória, ainda sobraria muito espaço em RAM
    - com essa solução simples, o pior caso do número de acessos diminui em um

Árvores-B Virtuais (*Virtual B-trees*)

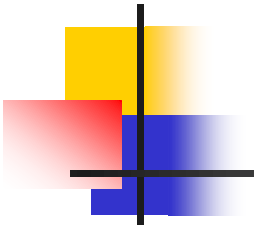


- Podemos generalizar esta idéia e ocupar toda a memória disponível com quantas páginas pudermos, sendo que, quando precisarmos da página, ela pode já estar em memória
- Se não estiver, ela é carregada para a memória, substituindo uma página que estava em memória
- Tem-se um *RAM buffer* que, algumas vezes, é chamado de árvore-B virtual

Política de gerenciamento de substituição: LRU

- Se a página não estiver em RAM, e esta estiver cheia, precisamos escolher uma página para ser substituída
- Uma opção: **LRU (Last Recently Used)**
  - substitui-se a página que foi acessada menos recentemente
- O processo de acessar o disco para trazer uma página que não está no *buffer* é denominado ***page fault***

Substituição baseada na altura da árvore



- Podemos optar por colocar todos os níveis mais altos da árvore em RAM. No exemplo de 256KB de RAM e páginas de 4KB, podemos manter 64 páginas em memória
- Isso comporta a raiz e mais, digamos, as 8 ou 10 que compõem o segundo nível. Ainda sobra espaço (utiliza-se LRU), e o número de acessos diminui em mais uma unidade.
- **Importante**
  Bufferização deve ser incluída em qualquer situação real de utilização de árvore-B.

Alocação da informação associada à chave

- E a informação associada às chaves (os demais campos dos registros), onde fica?
- Se a informação for mantida junto com a chave, ganha-se um acesso a disco, mas perde-se no número de chaves que pode ser colocado em uma página. Isso reduz a ordem da árvore, e aumenta a sua altura
- Se ficar em um arquivo separado, a árvore é realmente usada como índice, e cada chave tem o RRN, ou *byte offset*, que dá a posição do registro associado no arquivo de dados

Registros e Chaves de tamanho variável

- Até agora adotamos chaves de tamanho fixo. Em muitas situações, pode-se ter economia significativa de espaço usando chaves de tamanho variável
- Índices secundários referenciando listas invertidas são um bom exemplo desta situação...
- As árvores-B^+ adotam uma estrutura de página apropriada para acomodar chaves de tamanho variável