



Árvores-B (Parte IIIb)

SCC-203 – Algoritmos e Estruturas de Dados II

Graça Nunes



Efeitos da Redistribuição entre páginas irmãs

- Diferentemente do particionamento 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, ou
 - estratégia usual é redistribuir as chaves igualmente entre as páginas



Redistribuição durante inserção

- Redistribuição pode ser usada na inserção
- Seria uma opção desejável também na inserção
 - Em vez de particionar 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
 - Melhor utilização do espaço alocado para a árvore



Redistribuição vs. Particionamento (Splitting)

- Depois do **particionamento** 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

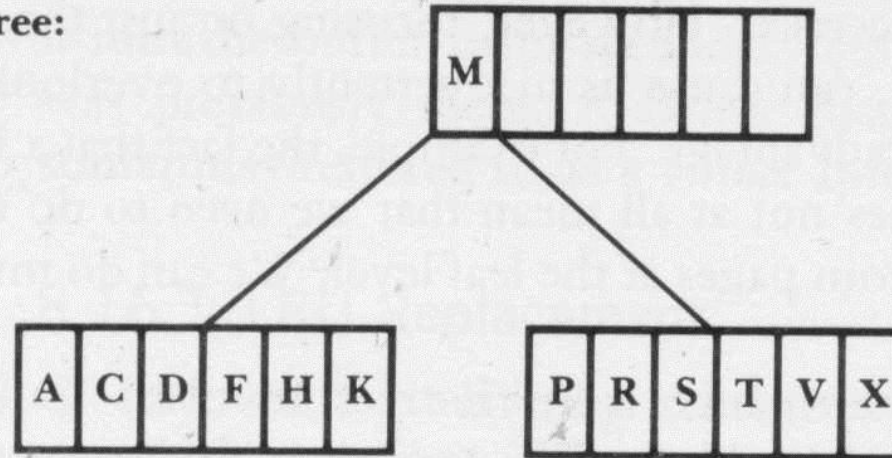


Árvores-B* (*B*-trees*)

- Proposta por Knuth em 1973, essa nova organização tenta redistribuir as chaves durante a inserção antes de particionar 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 particionamento
 - O *particionamento é adiado* 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

Original tree:



Two-to-three-split:
After the insertion of the
key *B*.

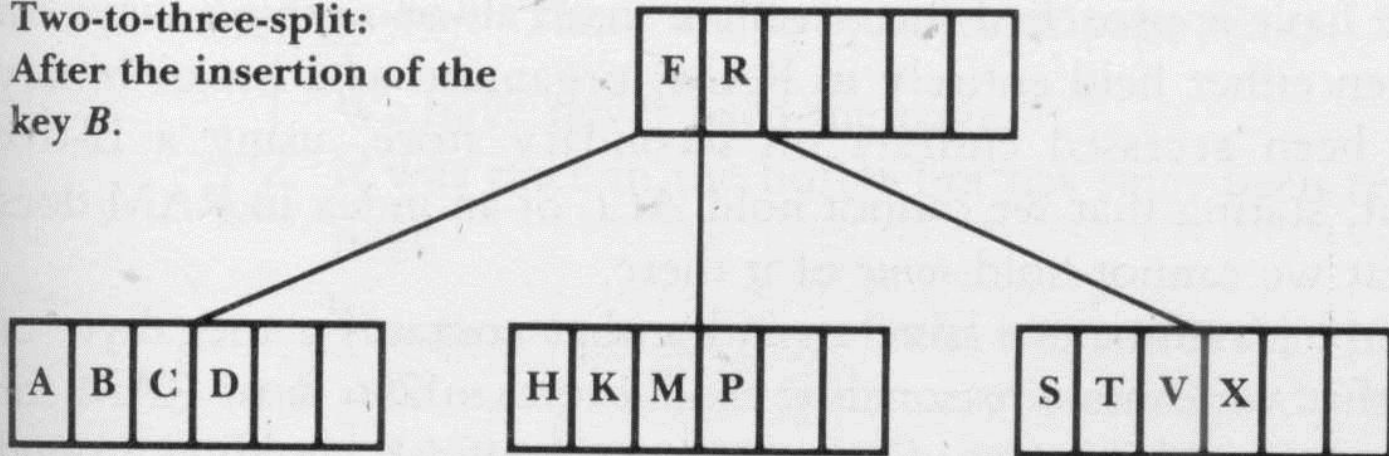


FIGURE 8.30 A two-to-three split.



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
- 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

- Esta propriedade **afeta as regras para remoção e redistribuição**
 - Com $2/3$ de ocupação, toda página irmã tem condições de redistribuir após um underflow em página vizinha.
- 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 um **nó com maior capacidade**

Árvores-B Virtuais

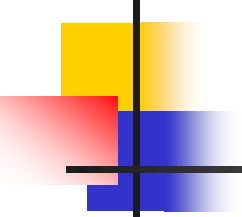
(*Virtual B-trees*)

- Árvores-B são muito eficientes, mas **podem ficar ainda melhores**
 - Observe, por exemplo, que o fato da árvore ter **profundidade 3 não implica** necessariamente fazer **3 acessos** para recuperar as páginas folhas (se a página desejada já estiver na RAM, por buferização)
 - O fato de não podermos manter todo o índice na RAM não significa que não se possa manter pelo menos **parte do índice em RAM**

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 (ordem 65)
- Nossa árvore-B pode estar totalmente contida em 3 níveis (altura 3 comporta 1.7 MB)
- 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 e, com essa solução simples, o pior caso do número de acessos diminui em 2 (um acesso a menos)



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 na RAM
- 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 até 64 páginas em memória
- Isso comporta a raiz e mais, digamos, as 8 ou 10 páginas 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