

Capítulo 8

Árvores B

Parte 2



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 Binária Paginada

- Voltando aos 2 problemas indentificados...
 - Pesquisa binária requer muitos seeks.
 - Manter um arquivo de índices ordenado é muito caro.

Árvore Binária Paginada busca resolver o problema do número de seeks.

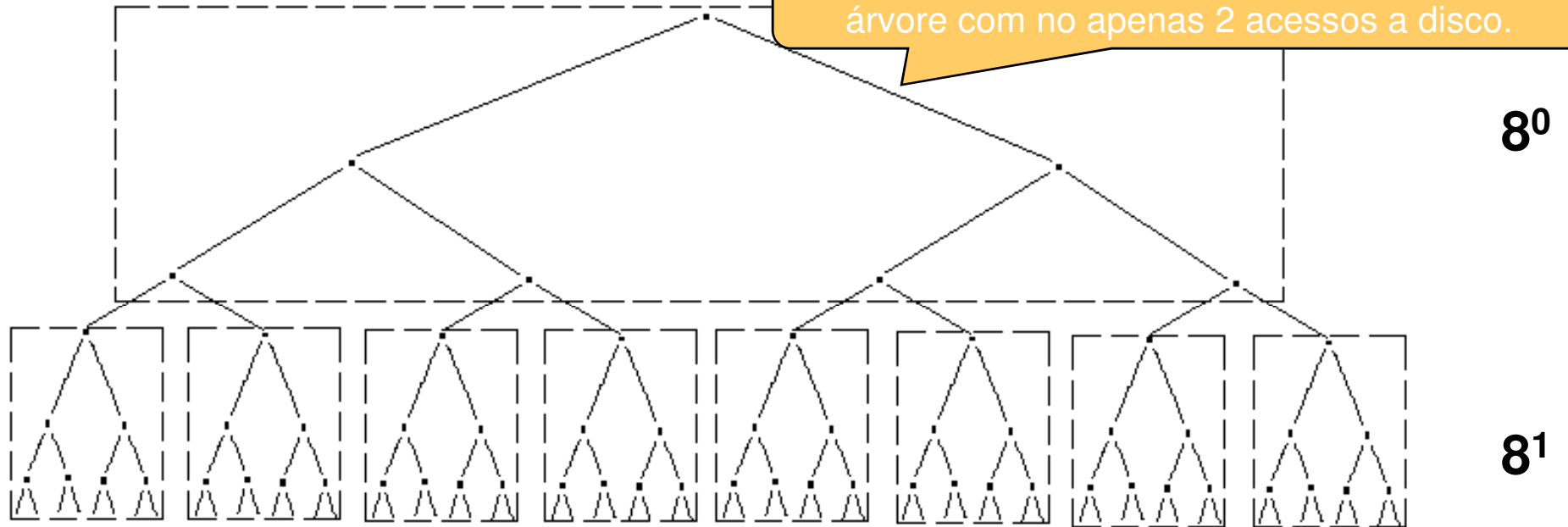


Árvore Binária Paginada

- Dividindo a árvore binária em páginas e então armazenando cada página num bloco contíguo do disco, podemos reduzir o número de seeks associado a qualquer busca.
 - Páginas (subárvores) são concatenadas em um único registro.
 - Adequado para árvores com nós pequenos:
 - Árvores onde cada nó possui um ponteiro p/ o arquivo de registros, ou
 - Árvores com registros de dados realmente pequenos.
 - A nível de arquivo de índices, um registro do arquivo possui vários nós da árvore.

Árvore Binária Paginada

Podemos localizar qualquer um dos 63 nós na árvore com no apenas 2 acessos a disco.



Árvore octal
Fator de ramificação 8
Chaves por página $2^3 - 1 = 7$

Temos:	nível	páginas	chaves	
	0	1	$7 \cdot 8^0$	8⁰
	1	8	$7 \cdot 8^1$	8¹
	2	64	$7 \cdot 8^2$	8²
total:	3	73	511	



Árvore Binária Paginada

- Fator de ramificação
 - Cada página contém ponteiros para outras páginas.
 - O número de ponteiros de uma página é o fator de ramificação.
- Ordem da árvore
 - O fator de ramificação dá a ordem da árvore.
 - Uma árvore paginada de ordem m tem:
 - m ponteiros em cada página
 - $m-1$ chaves em cada página
 - m é sempre uma potência de 2
 - Ex.: 2, 4, 8, 16, ..., 512,

Árvore Binária Paginada

- Árvore de ordem $m=8$
 - 8 ponteiros em cada página
 - 7 chaves por página

nível	páginas	chaves	Total chaves
0	1	7	7
1	8	56	63
2	64	448	511
3	512	3584	4095

Se adicionarmos um nível na árvore inicial, adicionamos 64 páginas, e podemos achar qualquer um dos 511 nós em apenas 3 seeks (são 3 níveis).

- Total de páginas em k níveis: $(8^k - 1) / 7$
 - para $k=4$, tem-se $(8^4 - 1)/7 = 585$
- Total de chaves em k níveis: $8^k - 1$
 - Para $k = 4$ tem-se 4095 chaves
- Mantendo a raiz na memória
 - Precisa-se no máximo $(k-1)$ seeks para acessar as $(8^k - 1)$ chaves

A pesquisa binária em uma lista de 4095 itens, pode precisar até 12 seeks!!!

Árvore Binária Paginada

- Árvore de ordem m , com k níveis
 - m ponteiros em cada página
 - $m-1$ chaves por página

nível	páginas	chaves	Total chaves
0	1	$m-1$	$m-1$
1	m	$m(m-1)$	(m^2-1)
2	m^2	$m^2(m-1)$	(m^3-1)
3	m^3	$m^3(m-1)$	(m^4-1)
.			
.			
$k-1$	m^{k-1}	$m^{k-1}(m-1)$	(m^k-1)
Total:	k	$(m^k-1)/(m-1)$	(m^k-1)

Mantendo a raiz na memória

Precisa-se de no máximo $(k-1)$ seeks para acessar as $(m^k-1)/(m-1)$ páginas.

Árvore Binária Paginada

- Árvore de ordem 512, com 3 níveis
 - 512 ponteiros em cada página
 - 511 chaves por página

$\log_2 68.719.476.735$
é aproximadamente 36!!!

	nível	páginas	chaves
	0	1	511
	1	512	261.632
	2	262.144	133.955.584
+	3	134.217.728	68.585.259.008
Total:	4	134.480.385	68.719.476.735

- Mantendo a raiz na memória
 - precisa-se de no MÁXIMO 3 seeks para acessar as 134.480.385 páginas.
 - em cada uma das páginas faz-se uma busca binária.



Árvore Binária Paginada

- Características da paginação:
 - Quanto maior a página, menos seeks serão necessários para a busca de uma chave.
 - O que limita o tamanho da página é a quantidade de memória disponível.
 - Cada página tem uma representação em árvore binária no disco.

● ● ● | Árvore Binária Paginada

- Comparando o pior caso para árvores binárias e árvores paginadas (completamente equilibradas e completas):
 - Árvore binária: $\log_2 (N+1)$
 - Árvore paginada: $\log_m (N+1)$
 - onde $m-1$ é o número de chaves em uma página.
 - para uma árvore não paginada, $m=2$

Lembrem-se que m é a ordem da árvore (ou seja, o número de ponteiros para as outras páginas)



Árvore Binária Paginada

- Desvantagens:
 - Para um m grande, aumenta o tempo de transmissão dos dados.
 - Quanto mais chaves, mais dados para levar do disco para a memória.
- Dificuldades em manter a árvore balanceada:
 - Cada página é um nó.
 - Cada nó é uma árvore.
 - Criar e manter balanceada tal árvore é complicado.
 - Mesmo usando rotações do tipo usado em árvores AVL.

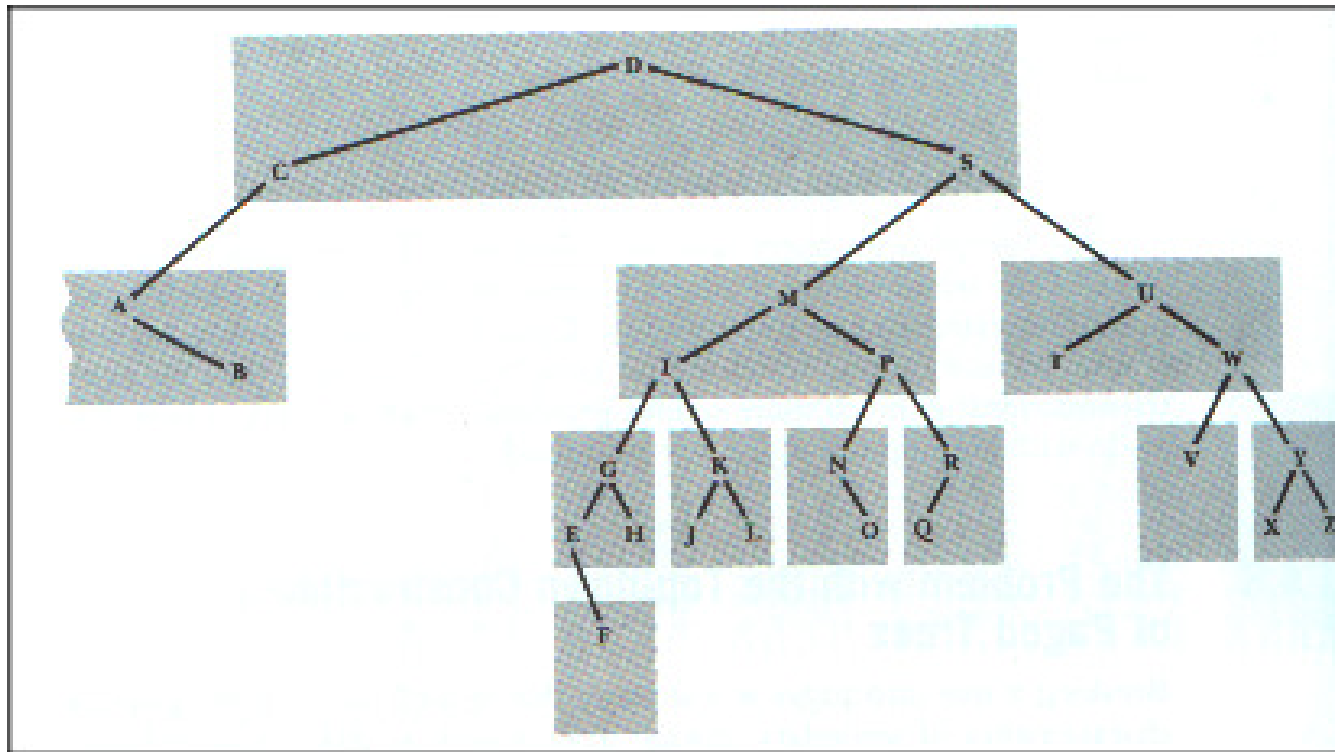
• • • | Árvore Binária Paginada

- Ilustrando o problema
 - Dado o seguinte exemplo:
 - Construir uma árvore binária com 3 chaves em cada página com as chaves sendo inseridas na seguinte ordem:

C	S	D	T	A	M	P	I	B	W	N	G	U	R	K	E	H	O	L	J	Y	Q	Z	F	X	V
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Lembrar que:
 - As páginas a esquerda tem chaves menores que a página raiz.
 - As páginas a direita tem chaves maiores que a página raiz.

Árvore Binária Paginada



C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



Árvore Binária Paginada

- C, D e S, por serem inseridos primeiro, desequilibram a árvore
- Mover (rotar) essa página à esquerda deixaria S fora de lugar
- C, D e S não deveriam estar na mesma página
- D não é um bom separador.



Árvore Binária Paginada

- O ideal seria classificar as chaves antes e inserir de forma bem balanceada (na maioria dos casos as chaves surgem, em qualquer ordem).
- Como balancear a árvore?
 - Não podemos simplesmente rotar páginas inteiras da árvore como rotaríamos chaves individuais numa árvore não paginada.
 - Precisaríamos quebrar as páginas: isso implica em reorganizá-las, criar novas páginas que sejam internamente balanceadas e bem arranjadas com relação as outras páginas.

É extremamente difícil criar um algoritmo que tenha apenas efeitos locais, reorganizando poucas páginas.

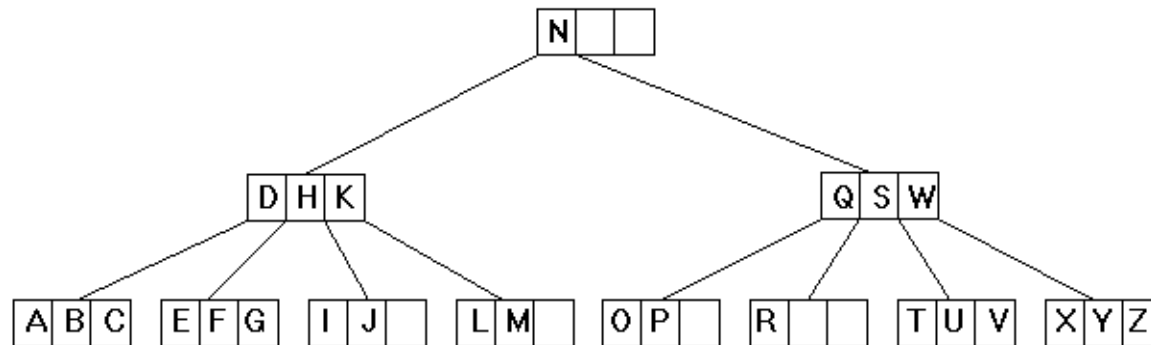


Árvore Binária Paginada

- A idéia de coletar chaves em páginas reduz o número de seeks. Porém ainda não encontramos uma maneira de coletar as chaves certas!
- Três perguntas importantes:
 - Como garantir que as chaves na raiz (ou raiz de sub-árvore) sejam bons “separadores”, dividindo de forma adequada o conjunto das outras chaves?
 - Como evitar que grupos de chaves tais como C, D e S no nosso exemplo venham a compartilhar a mesma página?
 - Como garantir que cada página tenha pelo menos um número mínimo de chaves se as páginas são muito grandes? (Para evitar situações em que um grande número de páginas contém poucas chaves.)

Árvore-B

- Proposta por Bayer e McCreight (1972)



● ● ● | Árvore-B

- O melhor modo de lidar com buscas em situações dinâmicas é usar árvores paginadas balanceadas.
 - Colocar muitas chaves por nó, aumenta o fator de ramificação e reduz a altura da árvore, diminuindo o esforço de busca.
 - Tais árvores foram denominadas Árvore-B

B de Boeing, balanceada, broad (larga), Bayer ou *bushy*?



Árvore-B

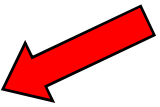
- Como os índices são implementados
 - Hoje a estrutura de dados mais comum é chamada árvore-B.
 - Árvore-B é uma generalização de árvore de pesquisa binária balanceada.
 - Enquanto a árvore binária tem no máximo dois filhos em cada nó, a Árvore-B tem um grande número de filhos. O nó de uma árvore-B é projetado para ocupar um bloco inteiro de disco.
 - Assim, uma busca em Árvore-B raramente envolve mais que três níveis.



Árvore-B

- Mesma idéia de páginas
 - Para explorar armazenagem em disco e paginar grandes tabelas de dados.
- Cada página é uma sequência de $m-1$ chaves e m ponteiros para outras páginas.
- m representa a ordem da árvore.
- A árvore pode representar o arquivo índice ou o próprio arquivo básico.
- Não há uma representação explícita de árvore em uma página. Cada página contém uma lista ordenada de chaves e ponteiros.

● ● ● | Árvore-B

- As árvores-B representam uma mudança de paradigma na construção de árvores
 - Passam a ser construídas de baixo para cima. 
 - Borbulha para a página raiz as chaves mais adequadas:
 - Resultado da inclusão em uma página lotada:
 - A chave de valor **médio** na página lotada (+ chave nova) sobe para a página pai.
 - A página lotada é dividida em duas.
 - Assim, os elementos da raiz emergem das páginas filhas (da base).



Árvore-B

- Numa árvore-B, cada página tem a seguinte forma:

p_0	ch_1	p_1	ch_2	p_2	ch_3	p_3	ch_j	p_j
-------	--------	-------	--------	-------	--------	-------	------	--------	-------

- Onde:
 - $j \leq m-1$
 - p_i ($0 \leq i \leq j$) são ponteiros para as páginas filhas.
 - Cada página aponta para no máximo m páginas filhas.
 - ch_i ($1 \leq i \leq j$) são as chaves armazenadas na página.
 - Armazenamos $m-1$ chaves numa página.



Árvore-B

- Nas folhas, cada p_i é nulo
- Uma representação computacional para cada nó pode ser:

contador=j	$ch_1 \ ch_2 \ ch_3 \ \dots \ ch_j$	$prr_1 \ prr_2 \ prr_3 \ \dots prr_j$	$p_0 \ p_1 \ p_2 \ p_3 \ \dots \ p_j$
------------	-------------------------------------	---------------------------------------	---------------------------------------

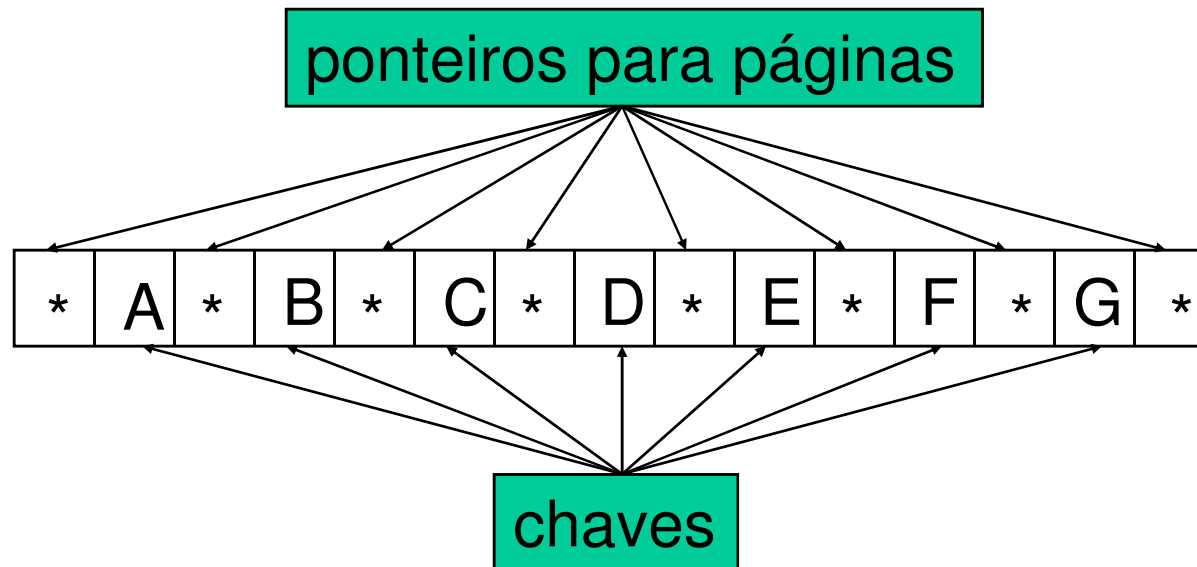
- Um registro com
 - contador: inteiro
 - chave: vetor $[1 \ .. \ m-1]$ do tipo chave
 - prr: vetor $[1 \ .. \ m-1]$ de inteiros
 - filho: vetor $[0 \ .. \ m-1]$ de inteiros

Árvore-B

- Exemplo: Árvore-B de ordem $m=8$
 - pagina inicial
 - 8 ponteiros e 7 chaves

OBS:

- Ponteiros para as páginas com * indicam vazio.
- Não estão representados os ponteiros para as chaves.





Próxima aula...

- Continuação das Árvores B