Estruturas de Indexação

1

Índices

- São estruturas de dados (arquivos) adicionais àquelas contendo os registros de dados (vide tópico anterior)
- Provêm caminhos de acesso alternativos aos registros sem afetar a disposição física dos registros no arquivo
- Um índice acelera a recuperação de registros baseada no campo de indexação
  - **Campo de indexação ou campo chave: atributos indexadores usados para construir o índice e para encontrar o end. do registro buscado.**
  - **Chave de busca: outro termo utilizado para fazer referência ao conjunto de campos usado para indexação de um arquivo. Não confundir com o conceito de chave.**
  - **A princípio, qualquer subconjunto de campos do registro de um arquivo pode compor uma chave de busca para construção de um índice sobre tal arquivo.**

2

2

Índices

O que é um índice:
Estrutura de dados interna ao SGBD que permite acesso mais rápido às informações do banco.

Exemplo (simplificado):

Fornecedor

CODIGO	NOME	STATUS	CIDADE
F1	SMITH	20	LONDRES
F2	JONES	10	PARIS
F3	BLAKE	30	PARIS
F4	CLARK	20	LONDRES
F5	ADAMS	30	ATENAS

Índice sobre o atributo Nome de Fornecedor

Nome	Endereço (Bloco)
ADAMS	5
BLAKE	3
CLARK	4
JONES	2
SMITH	1

Observação: Índices devem ser utilizados com critério pois afeta desempenho das consultas e das demais operações.

3

3

Índices

- Vantagens:
  - Acesso mais rápido ao registro quando a procura é sobre campo indexado.
  - Menos I/O: arquivo de índice menor que o arquivo de dados.
- Desvantagens:
  - Inclusão, exclusão e alteração ficam mais lentas.
  - Mais espaço de armazenamento.

4

4

Tipos de Índices

- Ind. Primário x Índ. Secundário x Índ. Clustering
- Índice Denso x Índice não Denso (Esparso)
- Índice de um único nível x índices de Múltiplos Níveis
- Índices Invertidos

5

5

Tipos de Índices

- Um **Índice Primário** é construído sobre o campo-chave de classificação de um arquivo ordenado de registros
  - Lembrando que: campo-chave de classificação é o campo usado para **ordenar fisicamente** os registros do arquivo no disco, e cada registro deve possuir um valor único para o campo
- Um **Índice Clustering** é construído sobre um campo de ordenação que não é um campo chave e por isso, diversos registros no arquivo podem ter o mesmo valor para este campo
- Um **Índice Secundário** é construído sobre quaisquer outros campos que não os de ordenação física do arquivo.

Obs: Um arquivo pode ter, no máximo, um campo de classificação física, portanto, ele só terá um índice primário ou um índice de cluster, mas não ambos.

6

6

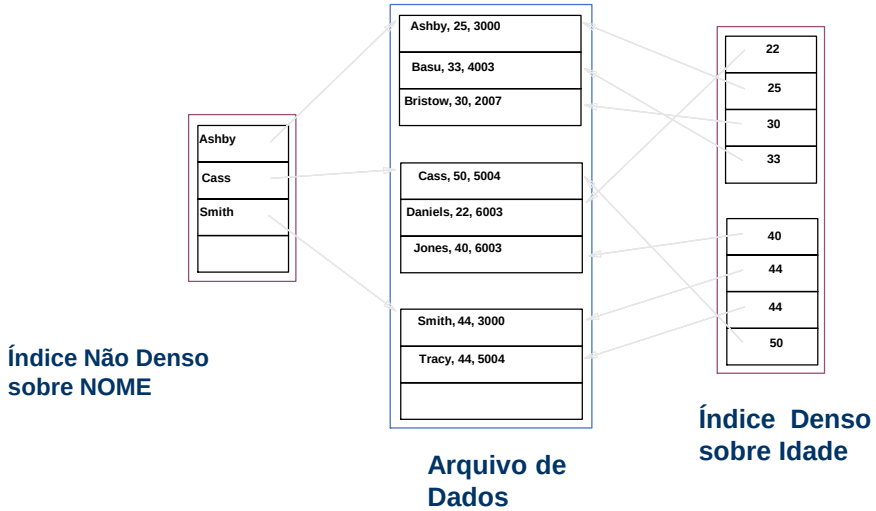
Tipos de Índices

- Um **Índice Denso** possui uma entrada no índice para cada registro no arquivo de dados.
  - Os registros podem estar armazenados em qualquer ordem no arquivo.
- Um **Índice Não Denso (índice esparso)** consiste num índice para blocos ou páginas do arquivo, cada um dos quais contendo um grupo de registros.
  - Os registros precisam estar organizados segundo o atributo indexador.

7

7

Tipos de Índices



8

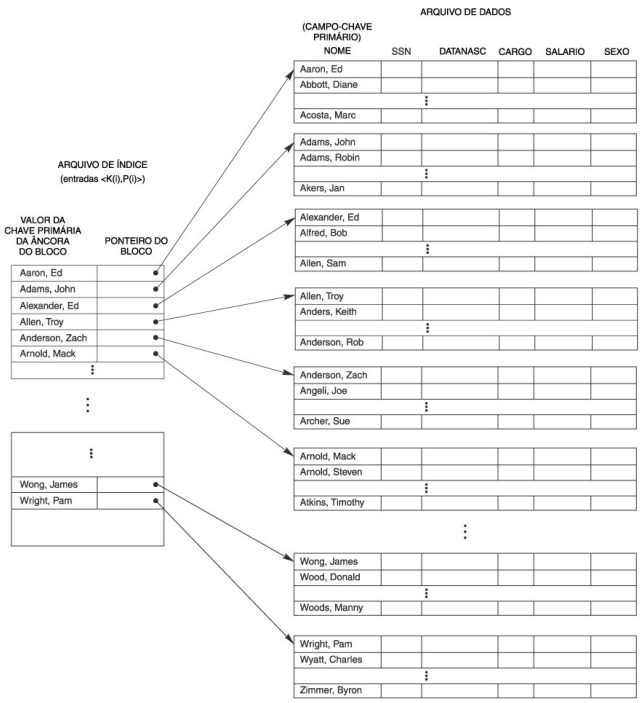
8

Índices Primários

- Registros com 2 campos:
  - Campo de mesmo domínio da chave de classificação do arquivo de dados
  - Ponteiro para um bloco de disco (end. bloco)
- Uma entrada de índice para cada bloco
  - Em cada entrada do índice, o valor do campo chave contém o valor do mesmo campo no primeiro registro do bloco – **registro âncora**.
  - No. de entradas do índice = No. de blocos que ocupa o arquivo
  - É um **índice esparso**
  - Precisa de muito menos blocos que o arquivo que indexa, portanto uma busca binária em um arquivo de índices exige muito menos acessos a blocos

9

9



10

10

Exemplo 1 – Sem Índices

r = 30.000 registros

- bloco B = 1.024 bytes.
- registro R = 100 bytes (reg. tamanho fixo)
- bfr(fator de blocagem) = $\lfloor B/R \rfloor$ = $\lfloor 1024/100 \rfloor$ = 10 registros por bloco
- número total de blocos: b = $\lceil r/bfr \rceil$ = $\lceil 30000/10 \rceil$ = 3000 blocos

Uma busca binária no arquivo necessitaria aproximadamente $\lceil \log_2 b \rceil = \lceil \log_2 3000 \rceil$ = 12 acessos ao bloco (supondo arquivo ordenado).

Uma busca linear, precisaria, no pior caso, de 3000 acessos (registro desejado no último bloco)

11

11

Exemplo 2 – com Índice Primário

- Cont. Exemplo 1
- chave de ordenação: V = 9 bytes
- ponteiro (para arquivo) : P = 6 bytes,
- cada entrada no índice :R_i = 9 + 6 = 15 bytes;
- fator de blocagem para o índice $bfr_i = \lfloor B/R_i \rfloor$ = $\lfloor 1024/15 \rfloor$ = 68 entradas por bloco.
- n^0 total de entradas no índice = n^0 de blocos do arq. dados: $\lceil r_i/bfr_i \rceil$ = 3000/68 = 45 blocos.

Utilizando pesquisa binária no índice: $\lceil \log_2 b_i \rceil = \lceil \log_2 45 \rceil$ = 6 acessos a bloco.

pesquisa do registro usando o índice: 6 + 1 = **7 acessos** (acesso adicional ao bloco no arquivo de dados), contra **12** da pesquisa binária sobre o arquivo de dados.

12

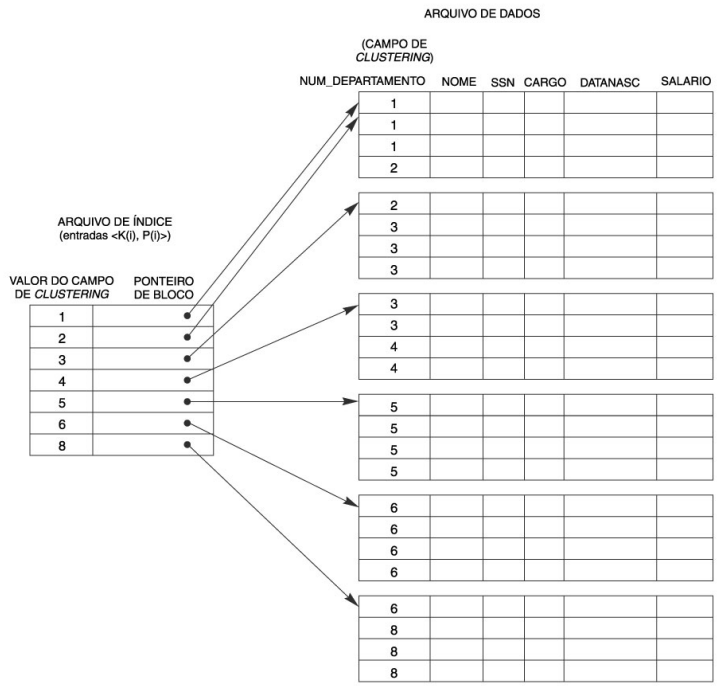
12

Índices Clustering

- Campo de indexação não é campo chave
  - Pode haver valores repetidos
  - É chamado de campo de agrupamento
- Registros com 2 campos:
  - Campo de mesmo domínio do campo de indexação do arquivo de dados
  - Ponteiro para um bloco de disco (end. bloco)
- Há uma entrada no índice para cada valor distinto que o campo de indexação assume
  - Índice esparso
  - Não necessariamente o registro âncora contém o mesmo valor do campo de indexação do índice
  - Para aliviar o problema da inclusão usa-se reservar um bloco ou conj de blocos para cada valor distinto

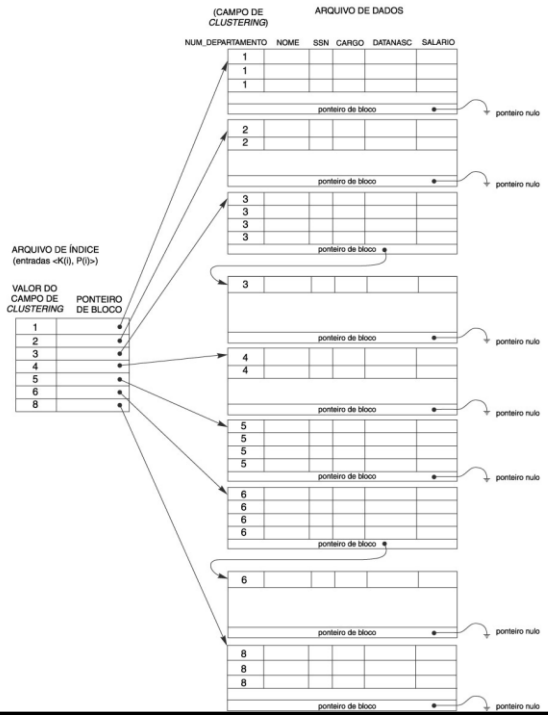
13

13



14

14



15

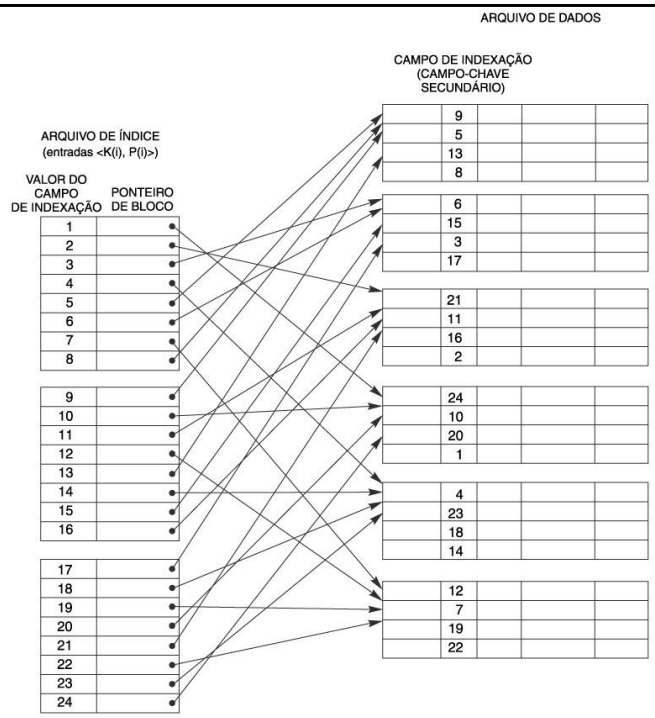
15

Indices Secundários

- Pode haver vários para um mesmo arquivo
- Estrutura similar aos outros tipos: 2 campos
- Pode ser construído
  - sobre chave-candidata – valor único por registro
    - Uma entrada para cada entrada do índice (índice denso), pois como o arquivo não está ordenado por chave-candidata, não podem ser utilizadas âncoras de bloco
    - Como há um maior número de entradas, então é maior tempo de busca em relação ao índice primário
    - Mas comparativamente ao arquivo não indexado, o ganho é maior, pois sem o índice seria necessário realizar uma busca linear

16

16



17

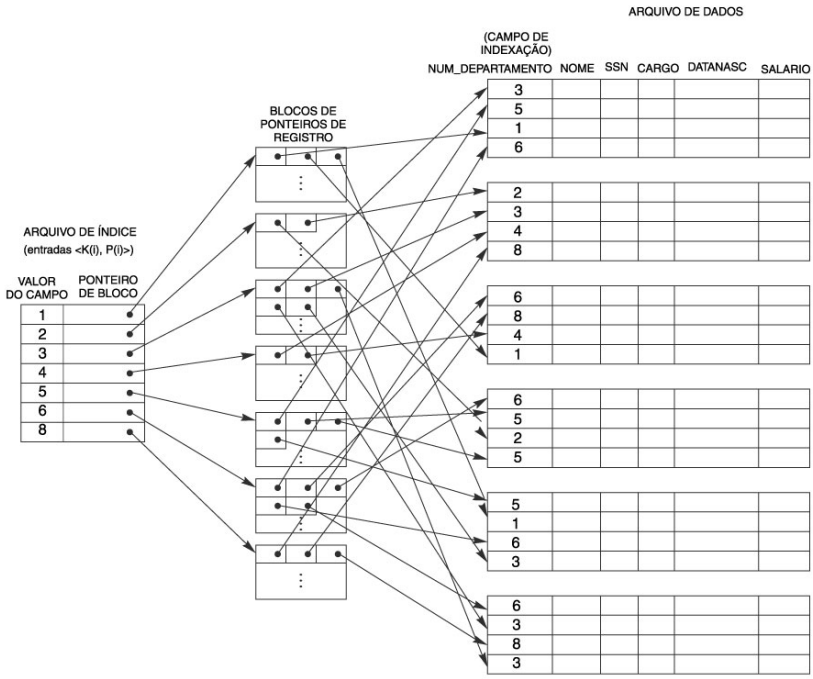
17

Indices secundários

- Também pode ser construído
  - Sobre campo não-chave – com valores repetidos
    - Diversas entradas no índice com um mesmo valor
    - Uma outra opção seria manter registros de tamanho variável nas entradas de índice (campo multivalorado), com vários endereços de bloco para um dado valor de indexação
    - A opção mais usada mantém registros de índice fixos e acrescenta mais um nível com os endereços de bloco

18

18



19

19

Exemplo 3 – com Índice Secundário

- r = 30.000 registros de R = 100 bytes ;
- B = 1.024 bytes (tamanho do bloco);
- n^0 blocos(b) = 3.000;
- busca linear no arquivo de dados*: b/2= 3000/2 = 1.500 acessos (média);

Utilizando um índice secundário

- tamanho campo chave V = 9 bytes; ponteiro de bloco (P) = 6 bytes;
- tamanho registro(índice) R_i = 9 + 6 = 15 bytes;
- fator de blocagem (índice) $bfr_i = \lfloor B/R_i \rfloor = \lfloor 1024/15 \rfloor = 68$
- número total de blocos (índice) $b_i = \lceil r_i/bfr_i \rceil = \lceil 30000/68 \rceil = 442$ blocos. (arq. denso, onde o n^0 de entradas = n^0 de registros no arq. de dados)
- busca binária : $\log_2 b_i = \lceil \log_2 442 \rceil = 9$ acessos a bloco
- pesquisa num registro 9 + 1 = 10 acessos (acesso adicional ao bloco)

10 contra 1500 acessos !!!

*: dados não ordenados

20

20

Resumo

	Campo de Indexação Utilizado para Classificar o Arquivo	Campo de Indexação Não Utilizado para Classificar o Arquivo
Campo de indexação é chave (não admite repetição de valor)	Índice Primário	Índice Secundário (sobre chave candidata)
Campo de indexação não é chave (admite repetição de valor)	Índice Clustering	Índice Secundário (sobre qualquer atributo não chave)

21

21

Resumo

Tipo de Índice	Número de Entradas de Índice (Primeiro Nível)	Denso ou Esparso	Âncora de Bloco no Arquivo de Dados
Primário	Número de blocos do arquivo de dados	Esparso	Sim
Clustering	Número de valores distintos do campo de indexação	Esparso	Sim/Não (*)
Secundário (sobre Chave Candidata)	Número de registros do arquivo de dados	Denso	Não
Secundário (sobre campo que não é chave)	Número de registros ou número de valores distintos do campo de indexação	Denso ou Esparso	Não

(*) – Sim, se todo valor distinto do campo de indexação iniciar um novo bloco, caso contrário, não

22

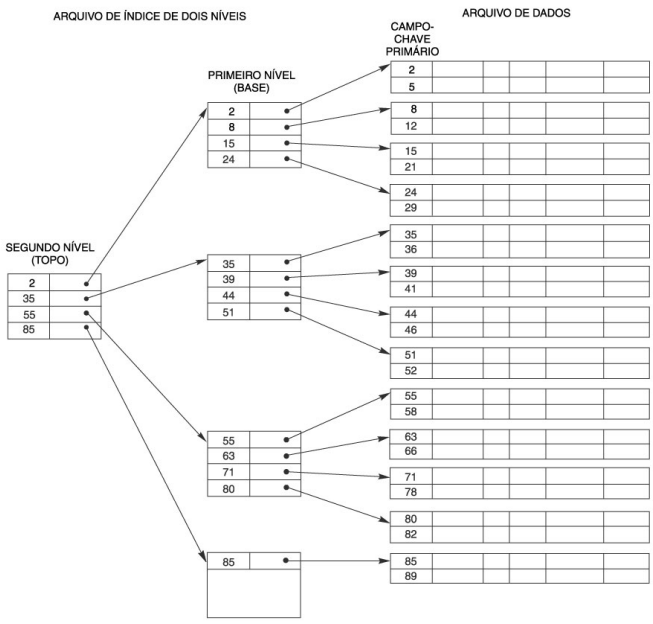
22

Índices de múltiplos níveis

- Um índice de um único nível é um arquivo ordenado.
- Por isso, é possível criar um índice não denso sobre um índice. Criamos assim um índice de dois níveis.
- Esse processo pode ser repetido criando-se um índice de múltiplos níveis, ou uma estrutura de árvore.

23

23



24

24

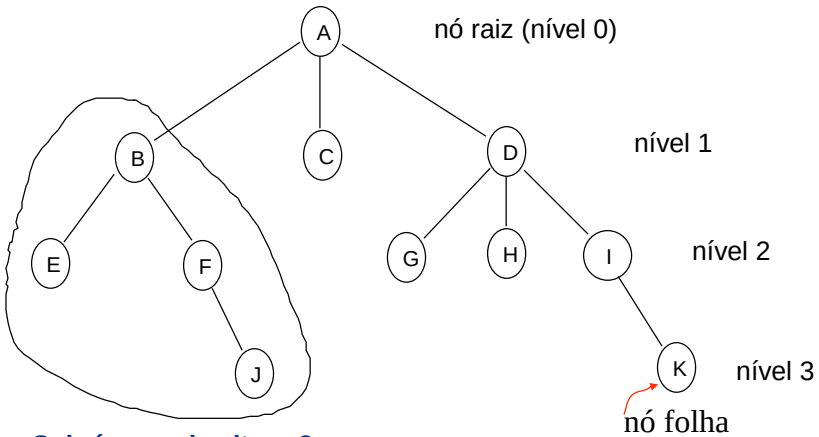
Índices de múltiplos níveis

- **Obs: Esta estrutura é mais eficiente pq:**
  - **A busca em um índice de um nível leva Log $_2$ b_i acessos**
  - **A busca em um índice multinível leva Log $_{brfi}$ b_i acessos**
    - **Cada nível reduz o n. de entradas do nível anterior dividindo-o por bfr_i**
- Pode-se construir índices multiníveis sobre qq índice seja ele primário, clustering ou secundário
- O problema com esta abordagem surge qdo precisamos incluir e excluir: pode gerar desbalanceamento
- É necessária uma estrutura que se reorganize na medida da necessidade. Ex: árvores B e B+ - que são classificadas como índices multiníveis dinâmicos

25

25

Árvores



26

26

Árvores de Busca

- Usada p/ pesquisa de registro, a partir do valor de um dos campos
- Uma árvore de **ordem p** é uma árvore em que cada nó contém no **máximo p-1 valores de busca** e p ponteiros na ordem

  $< P_1, K_1, P_2, K_2, \ldots, P_{q-1}, K_{q-1}, P_q >$

  onde:
  - $q \leq p$, P_i é um ponteiro para um nó filho (ou um ponteiro nulo)
  - K_i é um valor de pesquisa (todos os valores são distintos).
- Restrições:
  - Dentro de cada nó, $K_1 < K_2 < \ldots < K_{q-1}$
  - Para todos os valores X em uma subárvore apontada por P_i

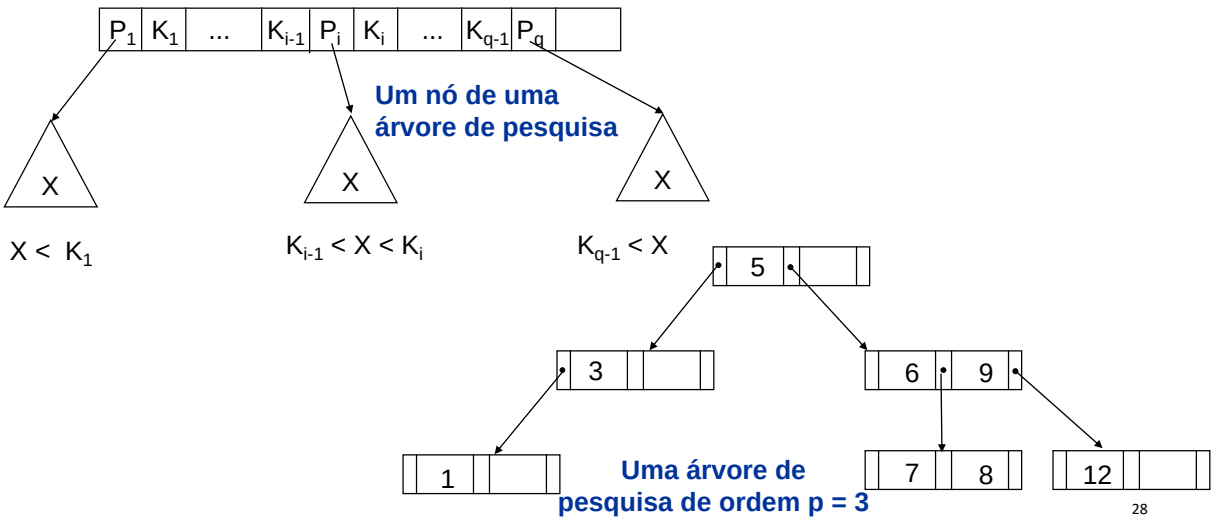
    $K_{i-1} < X < K_i$ para $1 < i < q$

    $X < K_i$ para $i = 1$

    $K_{i-1} < X$ para $i = q$

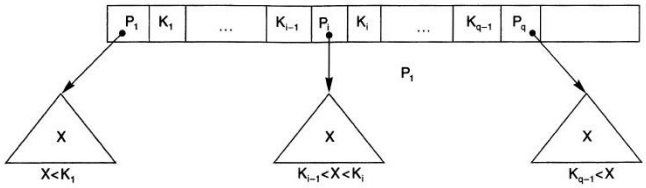
27

Árvores de Busca



Índices Multiníveis Dinâmicos

- Árvores de Busca
  - De ordem p (máximo de endereços apontados por um nó)
  - $<P_1,k_1,P_2,k_2, \ldots k_{q-1}, P_q>$ onde q <= p
  - $k_1 < k_2 < \ldots < k_{q-1}$
  - Pode haver ponteiros nulos



- Cada nó da árvore pode ser armazenado em um bloco
- Um ponteiro P_i pode apontar para registros ou blocos de registro que contenham um dado k_i

29

29

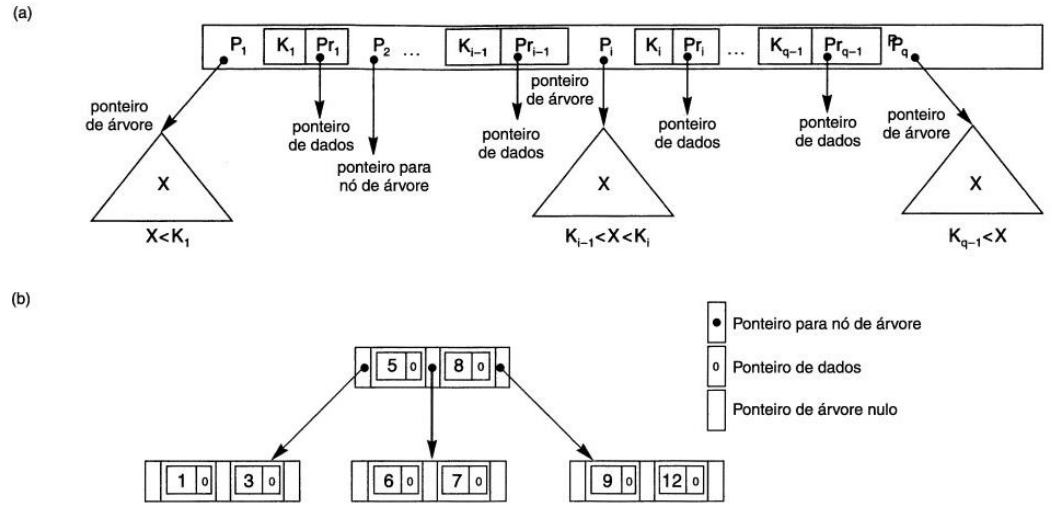
Estruturas do tipo Árvore B

- São estruturas balanceadas de múltiplos níveis, cada bloco do índice contendo espaço para um número fixo de ponteiros.
- Constituem estruturas dinâmicas, cujos nós se re-arranjam automaticamente com inserções e deleções de forma a manter a estrutura balanceada.
- B significa Balanced, pois todas as folhas estão a mesma distância do nó raiz.
- Assim as Árvores B garantem uma eficiência previsível.
- Permitem rápida recuperação de dados tanto randômica quanto sequencialmente.

30

30

Exemplo de Árvore-B



31

31

Árvore B: Balanceamento

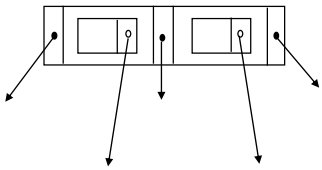
- Ao tentar inserir em um nó completo
  - Se for raiz, o nó se divide em dois nós de nível 1, onde somente o valor do meio se mantém na raiz.
  - Se não for raiz, o nó se divide em dois, e o valor do meio sobe para o nó pai, e se o nó pai estiver completo, propaga-se a divisão até chegar a raiz
- Ao excluir um nó com metade da capacidade
  - Ele é combinado com seus vizinhos

32

32

Inserções e Deleções em Árvores B

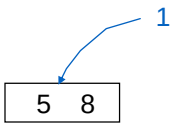
Valores a inserir: 8,5,1,7,3,12,9,6,4



nó de uma árvore B
de ordem p = 3

33

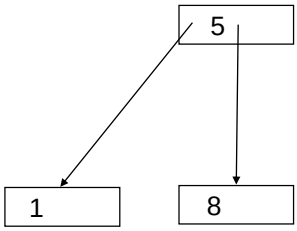
Inserções e Deleções em Árvores B



insere 8, 5
insere 1 (overflow)

34

Inserções e Deleções em Árvores B

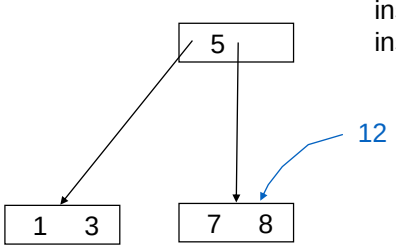


separa, novo nível

35

35

Inserções e Deleções em Árvores B

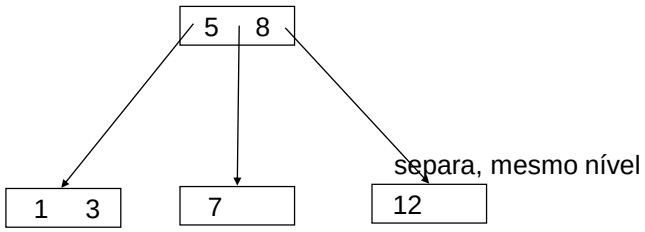


insere 7, 3
insere 12 (overflow)

36

36

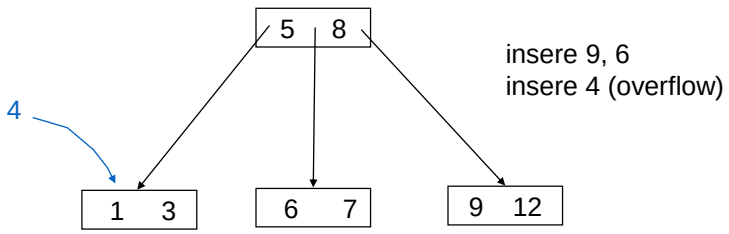
Inserções e Deleções em Árvores B



37

37

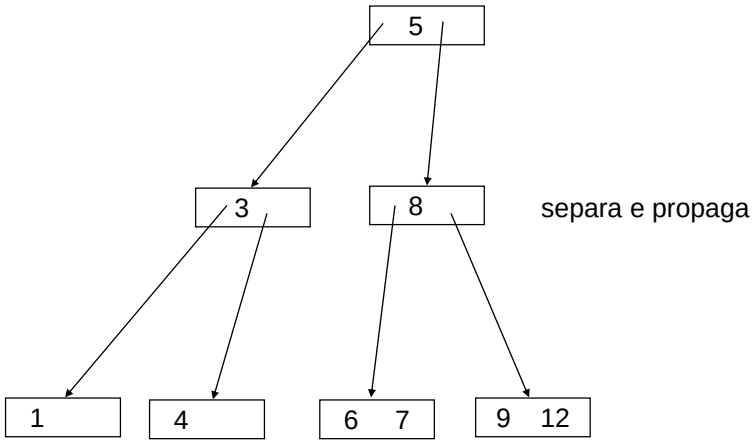
Inserções e Deleções em Árvores B



38

38

Inserções e Deleções em Árvores B



39

Exemplo 4

- Campo de busca V = 9 bytes
- Tamanho do bloco B = 512 bytes
- Tamanho do Ponteiro de Registro P_r = 7 bytes
- Tamanho do Ponteiro de Bloco P = 6 bytes
- Cálculo da ordem p de uma árvore: cada nó pode conter no máximo p ponteiros de dados e p-1 valores
  - (p * 6) + ((p-1)*(7+9) <= 512
  - (22 * p) <= 528
  - Escolhe-se p de modo que se aproveite bem o bloco (23)
  - Poderia ser 24, mas reserva-se espaço no bloco para informações adicionais como o número q de entradas no nó

40

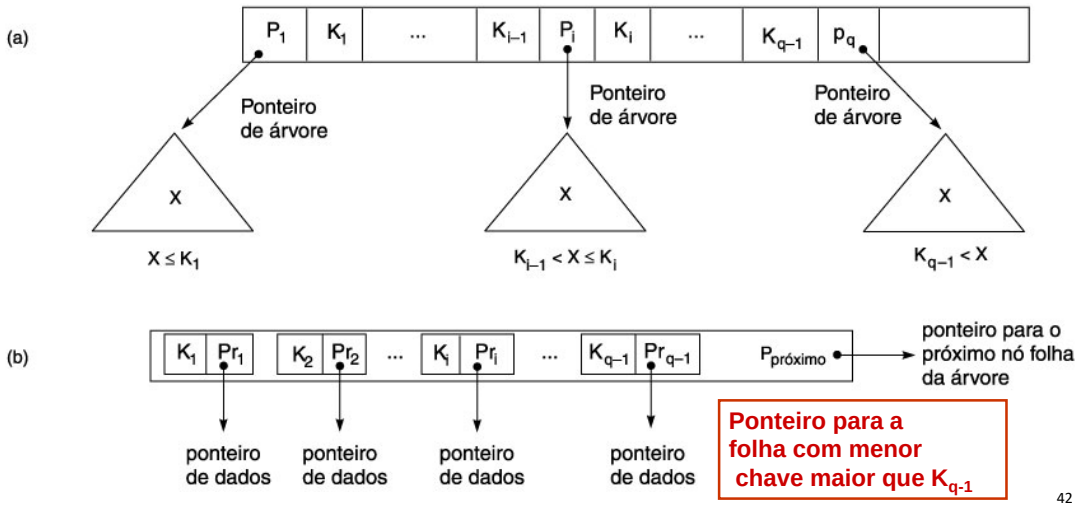
Estruturas do tipo Árvore B+

- Há diversas variedades de árvores B, a maior parte das implementações usa a árvore B+.
- Na árvore B+ apenas os nós folha têm ponteiros para os registros (ou blocos de registros) de dados.
- Uma árvore B+ de grau m tem as seguintes propriedades:
  - Todo nó tem entre [m/2] e m filhos (onde m é um inteiro > 3 e usualmente ímpar), exceto a raiz que não tem um limite inferior (m pode ser 0).
  - Todas as folhas estão no mesmo nível, ou seja, a mesma distância da raiz.
  - Um nó não folha com n filhos contém n-1 chaves.

41

41

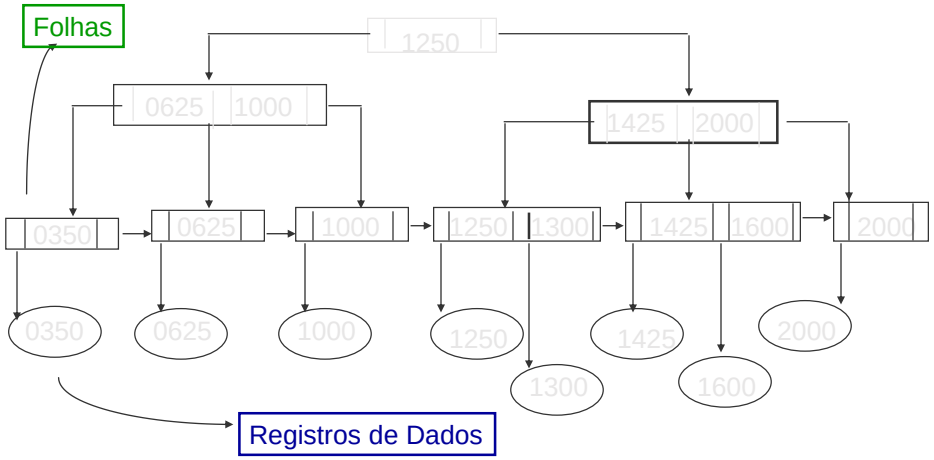
Exemplo de Árvore B+



42

42

Um exemplo simples de Árvore B+



43

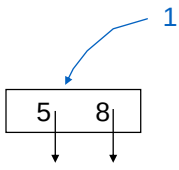
43

Inserções e Deleções em Árvores B+

Inserir: 8,5,1,7,3,12,9,6,4

árvore B^+ de ordem p = 3

$p_{folha} = 2$

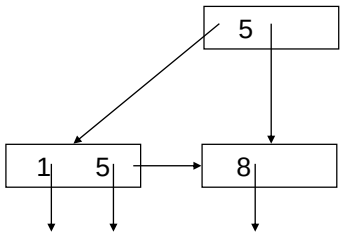


Insere 8, 5
Tenta inserir 1
(overflow)

44

44

Inserções e Deleções em Árvores B+

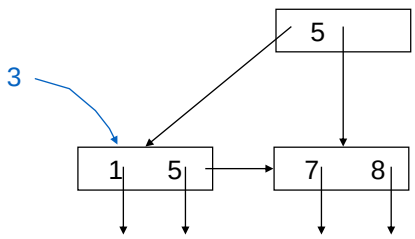


separação:
novo nível

45

45

Inserções e Deleções em Árvores B+



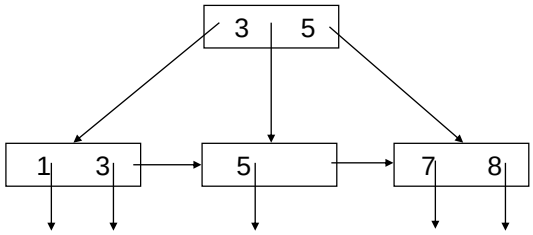
insere 7
tenta inserir 3
(overflow)

46

46

Inserções e Deleções em Árvores B+

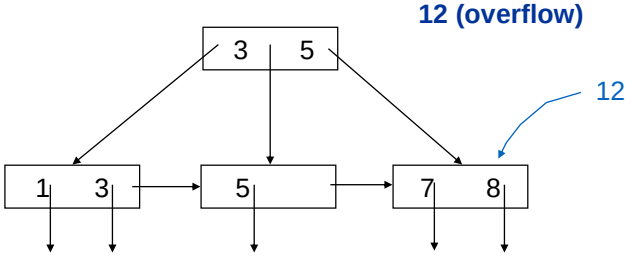
separação mesmo nível



47

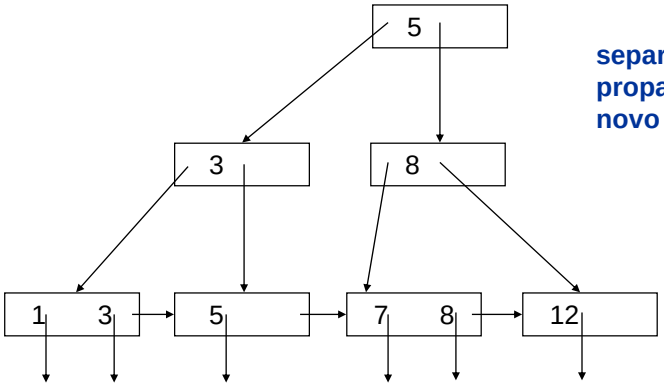
Inserções e Deleções em Árvores B+

tenta inserir 12 (overflow)



48

Inserções e Deleções em Árvores B+

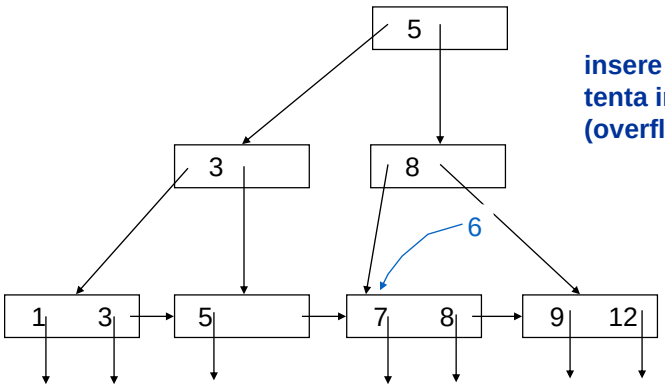


separação,
propagação,
novo nível

49

49

Inserções e Deleções em Árvores B+

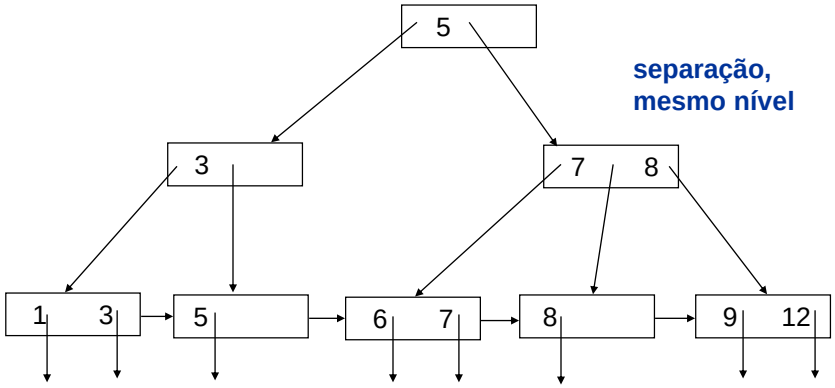


insere 9
tenta inserir 6
(overflow)

50

50

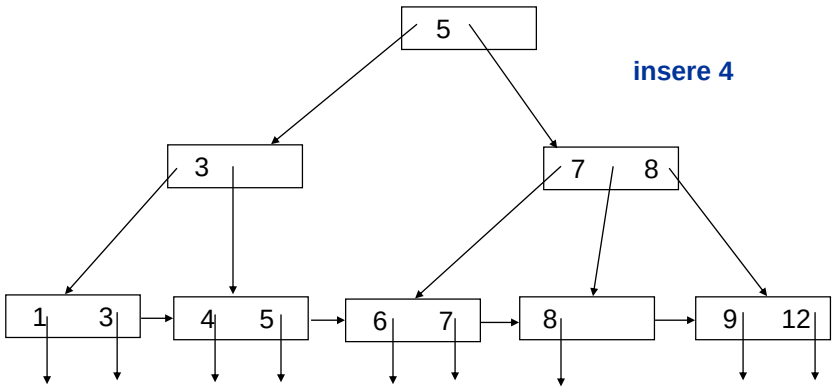
Inserções e Deleções em Árvores B+



51

51

Inserções e Deleções em Árvores B+

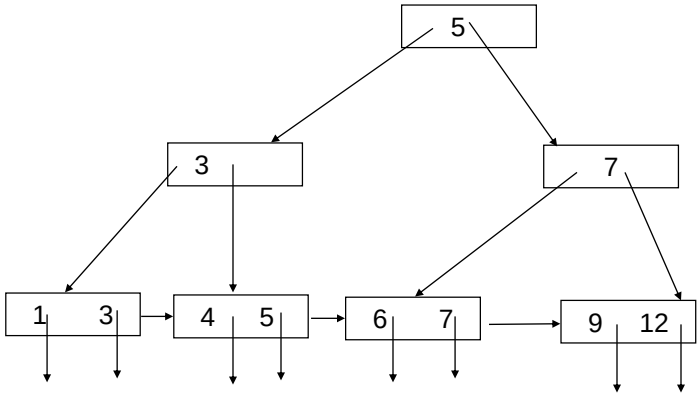


52

52

Inserções e Deleções em Árvores B+

Eliminando 8

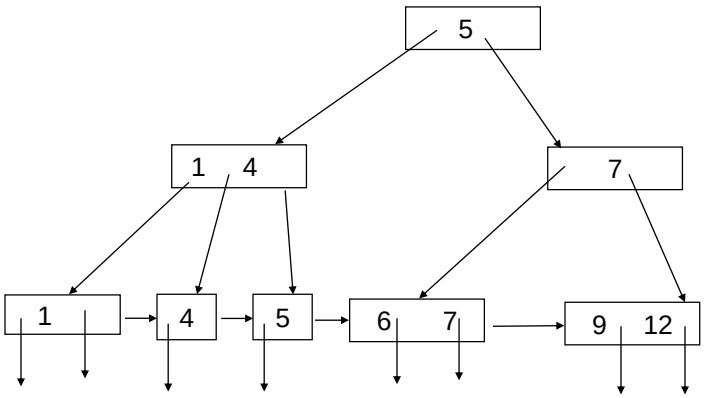


53

53

Inserções e Deleções em Árvores B+

Eliminando 3



54

54

Exemplo 5

- Campo de busca V = 9 bytes
- Tamanho do bloco B = 512 bytes
- Tamanho do Ponteiro de Registro P_r = 7 bytes
- Tamanho do Ponteiro de Bloco P = 6 bytes
- Cada nó interno tem no máximo p ponteiros de dados e p-1 valores e cada nó folha tem no máximo p_{folha} ponteiros
  - (p * 6) + ((p-1)*(9) <= 512
  - (15 * p) <= 521
  - Escolhe-se p de modo que se aproveite bem o bloco (34)
    - Cabem mais entradas que na árvore B correspondente
  - (p_{folha} * (7+ 9)) + 6 <= 512
  - 16 * p_{folha} <= 506
  - Escolhe-se p_{folha} = 31
  - Também no caso da árvore B+ se reserva um espaço do bloco para informações adicionais

55

55

Bons/ Maus candidatos para índice

- Examinar as consultas e operações sobre as relações

Bons Candidatos

- *Atributos da Primary key*
- *Atributos usados em junções (chaves estrangeiras)*
- *Atributos usados na cláusula WHERE: escolha aqueles usados em mais consultas ou nas consultas consideradas críticas*
- *Atributos usados na ordenação do resultado das consultas*
- *Atributos onde agregados são frequentemente calculados*

Maus Candidatos

- *Atributos com alta taxa de atualização*
- *Atributos com poucos valores distintos, com má distribuição de valores*
- *Atributos muito longos*

56

56