Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

Seção 18
Indexação de Arquivos

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Tipos de índices ordenados de único nível

Tipos de índices ordenados de único nível

Índice

- Índices são utilizados para **agilizar a recuperação de registros** em resposta a certas condições de pesquisa.
- As estruturas de índice são arquivos adicionais no disco que oferecem caminhos alternativos para acesso a registros.
- Para um arquivo com determinados campos (atributos), uma estrutura de acesso a índice normalmente é definida em um único campo.
- Chamado **campo de índice** (ou **atributo de indexação**).

Tipos de índices ordenados de único nível

Índice

- O índice armazena cada valor do **campo de índice** junto com uma **lista de ponteiros** para todos os blocos de disco que contêm registros com esse valor de campo.
- Para encontrar um registro, o índice é pesquisado, o que leva a ponteiros para um ou mais blocos de disco onde os registros exigidos estão localizados.

Tipos de índices ordenados de único nível

Índices de único nível

- Baseados em arquivos ordenados.
- Os valores no índice são *ordenados* de modo que possamos realizar uma *pesquisa binária* no índice.
- Tipos de índices ordenados:
  - Índice primário;
  - Índice de agrupamento (clustering);
  - Índice secundário.

Tipos de índices ordenados de único nível

Índice primário

- Atua como uma estrutura de acesso para procurar e acessar os registros em um arquivo.
- É um **arquivo ordenado** cujos registros são de tamanho fixo com dois campos:
  - o primeiro campo é do mesmo tipo do **campo de chave de ordenação (chave primária)** do arquivo;
  - e o segundo campo é um ponteiro para um bloco de disco (um endereço de bloco).

Tipos de índices ordenados de único nível

Índice primário

- O índice primário é especificado no *campo de chave de ordenação*.
- Um **campo de chave de ordenação** é usado para *ordenar* fisicamente os registros de arquivo no disco.
- Cada registro tem um valor *único* para esse campo.

Tipos de índices ordenados de único nível

Índice primário

- No arquivo de índice há uma **entrada de índice (registro de índice)** para cada *bloco no arquivo de dados*.
- Na entrada de índice há o valor da chave primária para o *primeiro* registro em um bloco e um ponteiro para esse bloco com seus dois valores de campo.
- Valores de campo da entrada de índice i:

```
<K(i), P(i)>
```

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*

Bloco 1

Nome	Cpf	Data_nascimento	Cargo	Salario	Sexo
Aaron, Eduardo					
Abílio, Diana					
•••					
Acosta, Marcos					

Bloco 2

Adams, João					
Adams, Roberto					
•••					
Akers, Janete					

Bloco 3

Alexandre, Eduardo					
Alfredo, Roberto					
•••					
Allen, Samuel					

Alguns blocos de um arquivo ordenado (sequencial) de registros de **FUNCIONARIO** com **Nome** como campo-chave de ordenação (campo exclusivo).

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*

- Sendo **Nome** exclusivo, usamos esse campo como chave primária (chave de ordenação do arquivo).
- Cada entrada no índice tem um valor de *Nome* e um ponteiro.

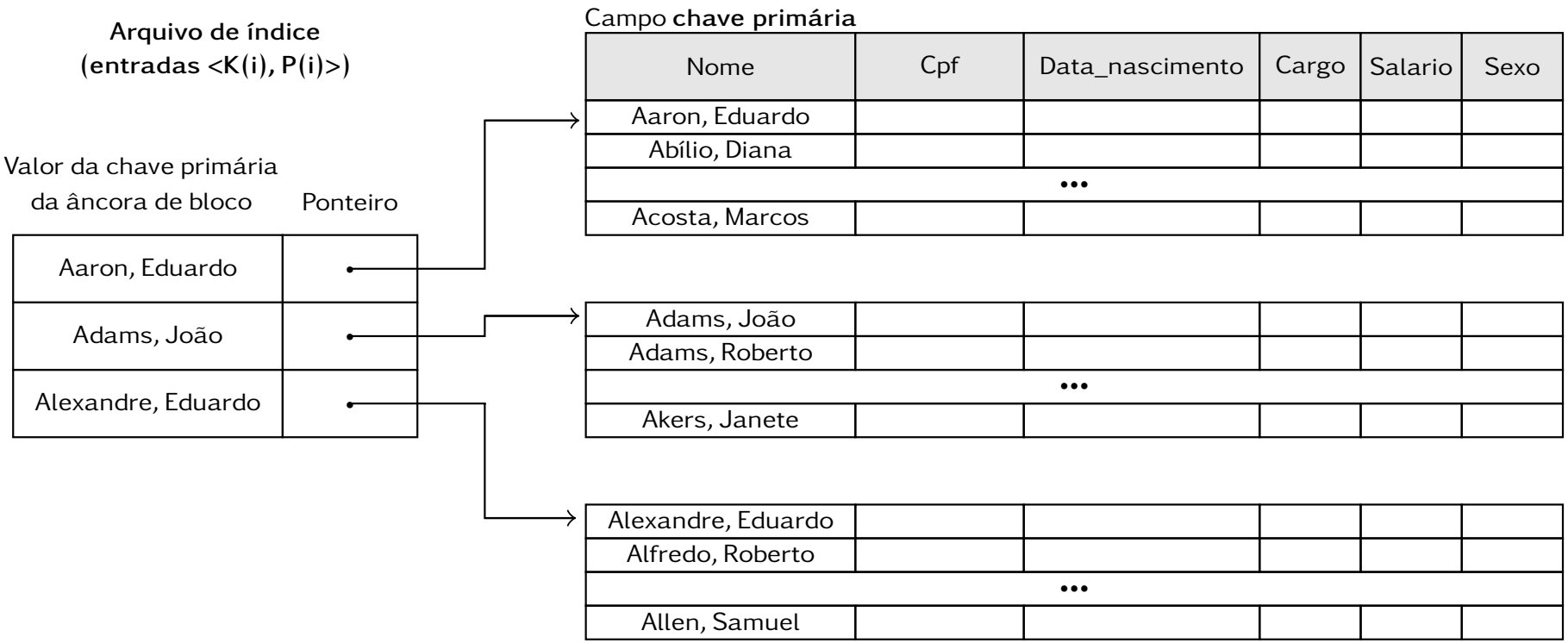
<K(1) = (Aaron, Eduardo), P(1) = endereco de bloco 1>

<K(2) = (Adams, Joao), P(2) = endereco de bloco 2>

<K(3) = (Alexadre, Eduardo), P(3) = endereco de bloco 3>

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*



Índice primário no campo de chave de ordenação.

Tipos de índices ordenados de único nível

Índice primário

- O número de entradas no índice é igual ao *número de blocos de disco* no arquivo ordenado.
- O primeiro registro em cada bloco é **registro de âncora** do bloco (**âncora do bloco**).

Tipos de índices ordenados de único nível

Índice primário

- Os índices também podem ser caracterizados como **densos** ou **esparsos**.
  - **Índice denso** tem uma entrada de índice para *cada valor de chave de pesquisa* (cada registro) no arquivo.
  - **Índice esparso** (ou **não denso**) tem entradas de índice para somente alguns dos valores de pesquisa.
- Um índice primário é um índice esparso, pois inclui uma entrada para cada bloco de disco do arquivo e as chaves de seu registro de âncora (ao invés de cada registro).

Tipos de índices ordenados de único nível

Índice primário

- O arquivo de índice (para um índice primário) ocupa um espaço menor do que o arquivo de dados.
  - existem *menos entradas de índice* do que registros;
  - cada entrada de índice normalmente é *menor em tamanho* do que um registro de dados.
- Em consequência, mais entradas de índice do que registros podem caber em um bloco.

Tipos de índices ordenados de único nível

Índice primário

- Portanto, uma pesquisa binária no arquivo de índice requer menos acessos de bloco do que uma pesquisa binária no arquivo de dados.
- Uma pesquisa binária requer aproximadamente $\mathbf{log_2 b_i}$ acessos de bloco para um índice com b_i blocos.
- Se o arquivo de índice primário possuir b_i blocos, localizar um registro com um valor de chave de pesquisa exige uma pesquisa binária desse índice e o acesso ao bloco que contém esse registro: $\log_2 b_i + 1$ acessos.

Tipos de índices ordenados de único nível

Índice primário

- Um registro cujo valor da chave primária é K se encontra no bloco de endereço $P(i)$, onde $K(i) <= K < K(i + 1)$.
- O i-ésimo bloco no arquivo de dados contém todos os registros.
- Para recuperar um registro, realiza-se uma pesquisa binária no arquivo de índice para encontrar a entrada de índice apropriada i.

  Depois recupera-se o bloco do arquivo de dados cujo endereço é $P(i)$.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

Suponha um arquivo ordenado com r = 30.000 registros em um disco com tamanho de bloco B = 1.024 bytes. Os registros são de tamanho fixo com tamanho R = 100 bytes.

- O fator de bloco para o arquivo seria
  $bfr = [(B/R)] = [(1.024/100)] = 10$ registros por bloco.
- O número de blocos necessários para o arquivo é
  $b = [(r/bfr)] = [(30.000/10)] = 3.000$ blocos.
- Uma pesquisa binária no arquivo de dados precisaria de
  $[\log_2 b] = [(\log_2 3.000)] = 12$ acessos de bloco.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

Suponha que o campo de chave de ordenação do arquivo seja V = 9 bytes de extensão, ponteiro de bloco seja P = 6 bytes de extensão e tenhamos construído um índice primária para o arquivo.

- O tamanho de cada entrada de índice é

  $R_i = (9 + 6) = 15$ bytes.

- O fator de bloco para o índice é

  $bfr_i = [(B/R_i)] = [(1.024/15)] = 68$ entradas por bloco.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

- O número total de entradas de índice r_i é 3.000 (igual ao número de blocos no arquivo de dados).
- O número de blocos de índice é

$$b_i = [(r_i/bfr_i)] = [(3.000/68)] = 45 \text{ blocos.}$$

- Para uma pesquisa binária no arquivo de índice seriam necessários $[(\log_2 b_i)] = [(\log_2 45)] = 6$ acessos de bloco.
- Para procurar um registro usando o índice, precisa-se de um acesso de bloco adicional ao arquivo de dados,

  $6 + 1 = 7$ acessos a bloco de disco (melhoria em relação a pesquisa binária no arquivo de dados).

Tipos de índices ordenados de único nível

Índice primário: Inserção e exclusão de registros

- Em um índice primário temos de mover registros e mudar algumas entradas de índice para inserção e exclusão,

  pois a movimentação de registros mudará os *registros de âncora* de alguns blocos.

- Possibilidades:
  - Usar um arquivo de overflow desordenado.
  - Usar uma lista ligada de registros de overflow para cada bloco.

Tipos de índices ordenados de único nível

Índice primário: Inserção e exclusão de registros

- Os registros em cada bloco e sua lista ligada de overflow podem ser classificados para melhorar o tempo de recuperação.
- A exclusão de registro é tratada com marcadores de exclusão.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Se registros no arquivo puderem ter o mesmo valor para o campo de ordenação (campo de ordenação não é um campo de chave).
- O arquivo de dados é chamado de **arquivo agrupado**.
- Um arquivo pode ter no máximo um índice primário ou de agrupamento (máximo um campo de ordenação), *mas não ambos*.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Se os registros forem fisicamente ordenados em um campo não chave (que *não* tem um valor distinto para cada registro) esse campo é um **campo de agrupamento**.
- O arquivo de dados é chamado de **arquivo agrupado**.
- O **índice de agrupamento** agiliza a recuperação de todos os registros que têm o mesmo valor para o campo de agrupamento.

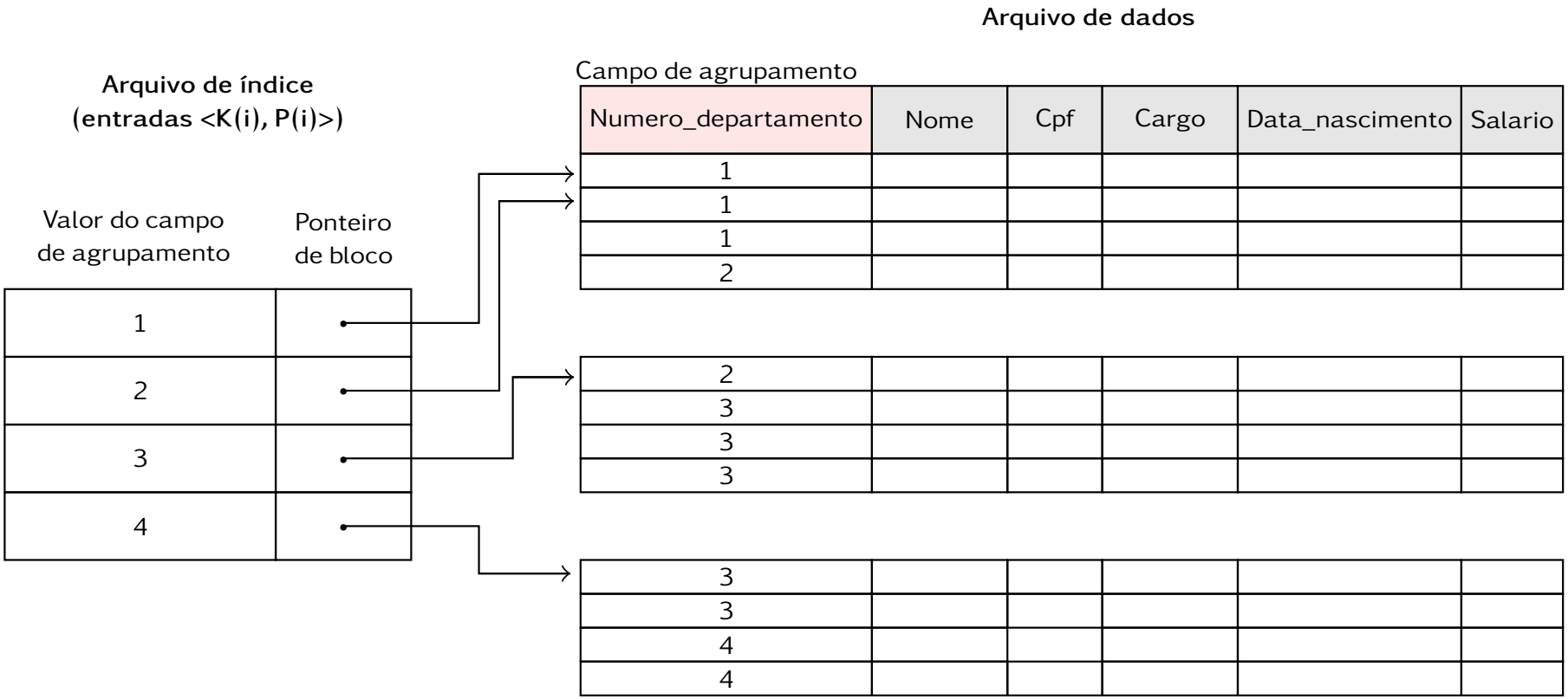
Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Um índice de agrupamento é um arquivo ordenado com dois campos:
  - o primeiro campo é do mesmo tipo do campo de agrupamento;
  - e o segundo campo é um ponteiro para o ***primeiro bloco*** no arquivo que tem um registro com esse valor para seu campo de agrupamento.
- Há uma entrada no índice para cada ***valor distinto*** do campo de agrupamento.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)



Índice de agrupamento no campo não chave de ordenação **Numero_departamento** de um arquivo **FUNCIONARIO**.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering): Inserção e exclusão de registro

- A movimentação de registros ainda causa problemas, pois os registros estão fisicamente ordenados.
- Possibilidade:
  - Reservar um bloco inteiro (ou um cluster de blocos contínuos) para ***cada valor*** do campo de agrupamento;
  - todos os registros com esse valor são colocados no bloco (ou cluster de bloco).

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Um índice de agrupamento é um índice ***não denso***, pois ele tem uma entrada para cada ***valor distinto*** do campo de índice.

Tipos de índices ordenados de único nível

Índice secundário

- Especificado em qualquer campo *não* ordenado de um arquivo.
- Um arquivo de dados pode ter vários índices secundários além de seu método de acesso primário.

Tipos de índices ordenados de único nível

Índices secundários

- Oferecem um meio secundário para acessar um arquivo de dados para o qual algum acesso primário já existe.
- Registros do arquivo podem ser ordenados, desordenados ou hashed.
- Consiste em um arquivo ordenado com dois campos:
  - o primeiro campo é do mesmo tipo de algum *campo não ordenado* do arquivo que seja um **campo de índice**.
  - o segundo campo é um ponteiro de *bloco* ou um ponteiro de *registro*.

Tipos de índices ordenados de único nível

Índices secundários

- O índice secundário é criado em um campo que é uma chave candidata e tem um valor único em cada registro;
- Ou em um campo não chave com valores duplicados.
- *Muitos* índices secundários (e, portanto, campos de indexação) podem ser criados para o mesmo arquivo.
- Cada um representa um meio adicional de acessar esse arquivo com base em algum campo específico.

Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)

- Esse campo que tem um *valor distinto* para cada registro, é chamado de chave **secundária**.
- Correspondente ao atributo de chave UNIQUE (Modelo Relacional) ou ao atributo de chave primária de uma tabela.
- Existe uma entrada de índice para *cada registro* no arquivo que contém:
  - o valor do campo para o registro;
  - e um ponteiro para o bloco em que o registro está armazenado ou para o próprio registro (índice **denso**).

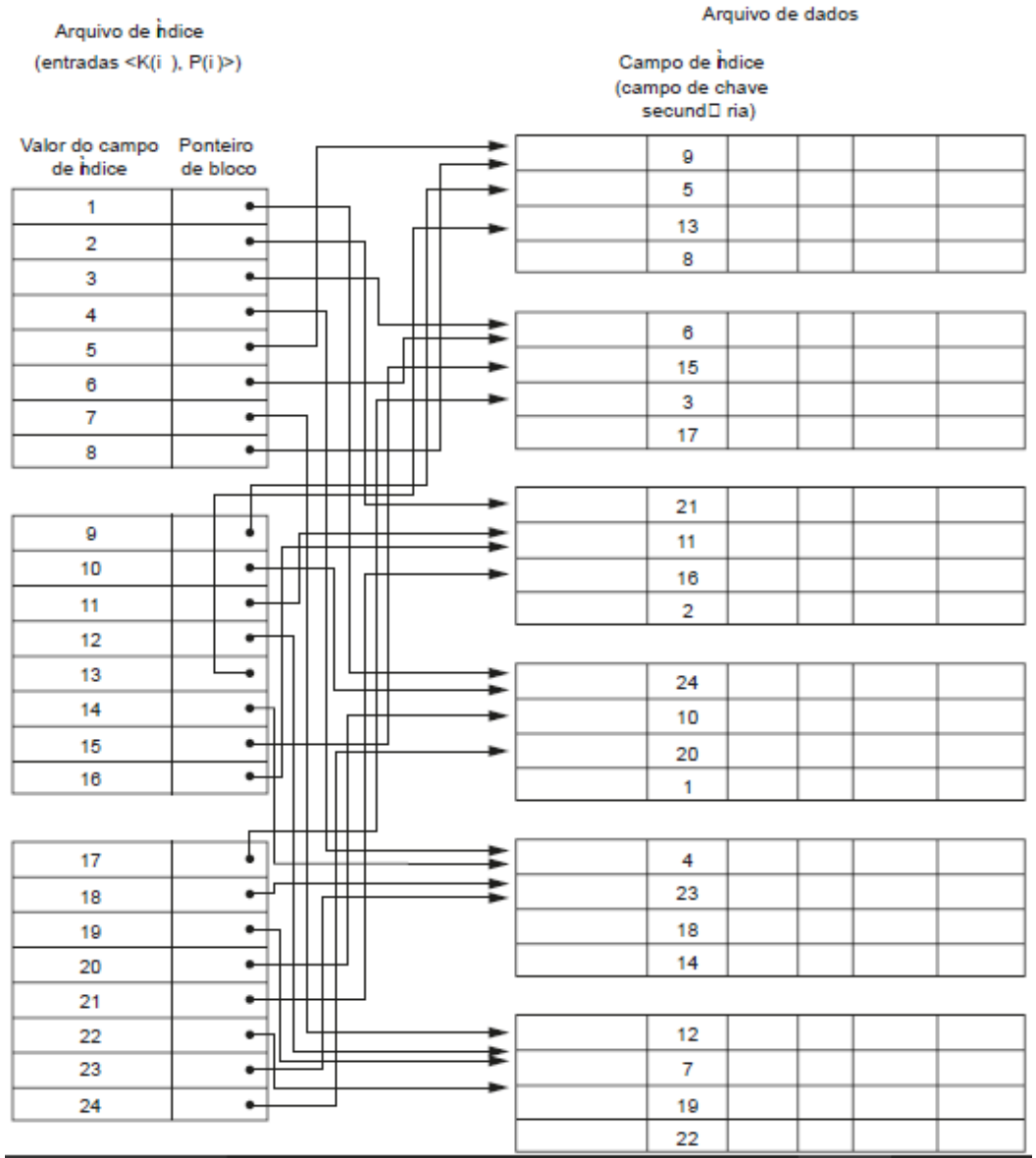
Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)

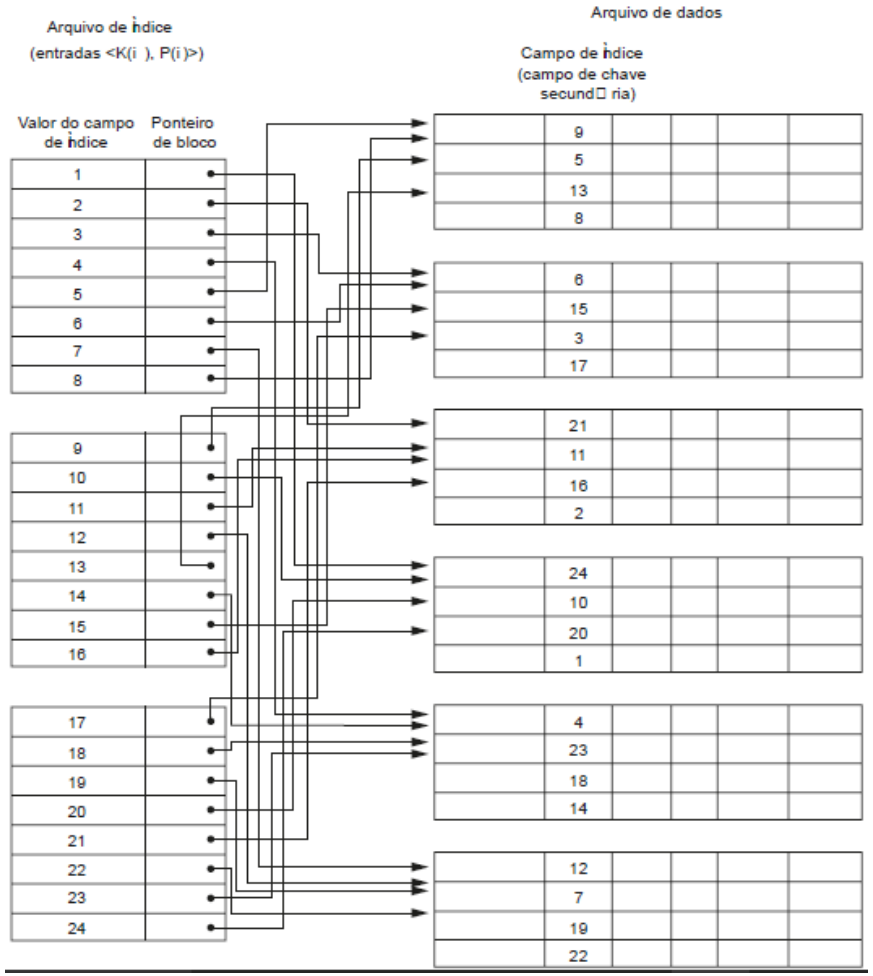
- Valores de campo da entrada de índice *i*: *<K(i), P(i)>*
- As entradas são **ordenadas** pelo valor de *K(i)* (é possível realizar uma pesquisa binária).
- Como os registros do arquivo *não são* fisicamente ordenados pelos valores do campo de chave secundário, *não podemos* usar âncoras de bloco.

Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)



Tipos de índices ordenados de único nível



Quando o bloco de disco apropriado é transferido para um buffer da memória principal, uma pesquisa pelo registro desejado no bloco pode ser executada.

Tipos de índices ordenados de único nível

Índices secundários

- Índice secundário em um campo chave (único)
- Índice secundário, em geral, precisa de mais espaço de armazenamento e tempo de busca maior do que um índice primário, devido a seu maior número de entradas.
- Porém, a *melhoria* no tempo de pesquisa para um registro é muito maior para um índice secundário.
- Visto que teríamos de fazer uma *pesquisa linear* no arquivo.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

- Nesse caso, diversos registros no arquivo podem ter o mesmo valor para o campo de índice. Opções para implementar tal índice:

OPÇÃO 1

- Incluir entradas de índice duplicadas com o mesmo K(i) - um para cada registro (índice denso).

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 2

- Registros de tamanho variável para as entradas de índice, com um campo repetitivo para o ponteiro.
- Mantemos um ponteiro para cada bloco que contém um registro cujo valor do campo de índice é igual a $K(i)$.

Na opção 1 ou 2, o algoritmo de pesquisa binária deve ser modificado para considerar um número variável de entradas de índice por valor de chave de índice.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 3

- Esquema não denso e mais utilizado.
- Mantem as próprias entradas de índice em um tamanho fixo e tem uma única entrada para cada *valor de campo de índice*.
- Mas cria *um nível de indireção extra* para lidar com os múltiplos ponteiros.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

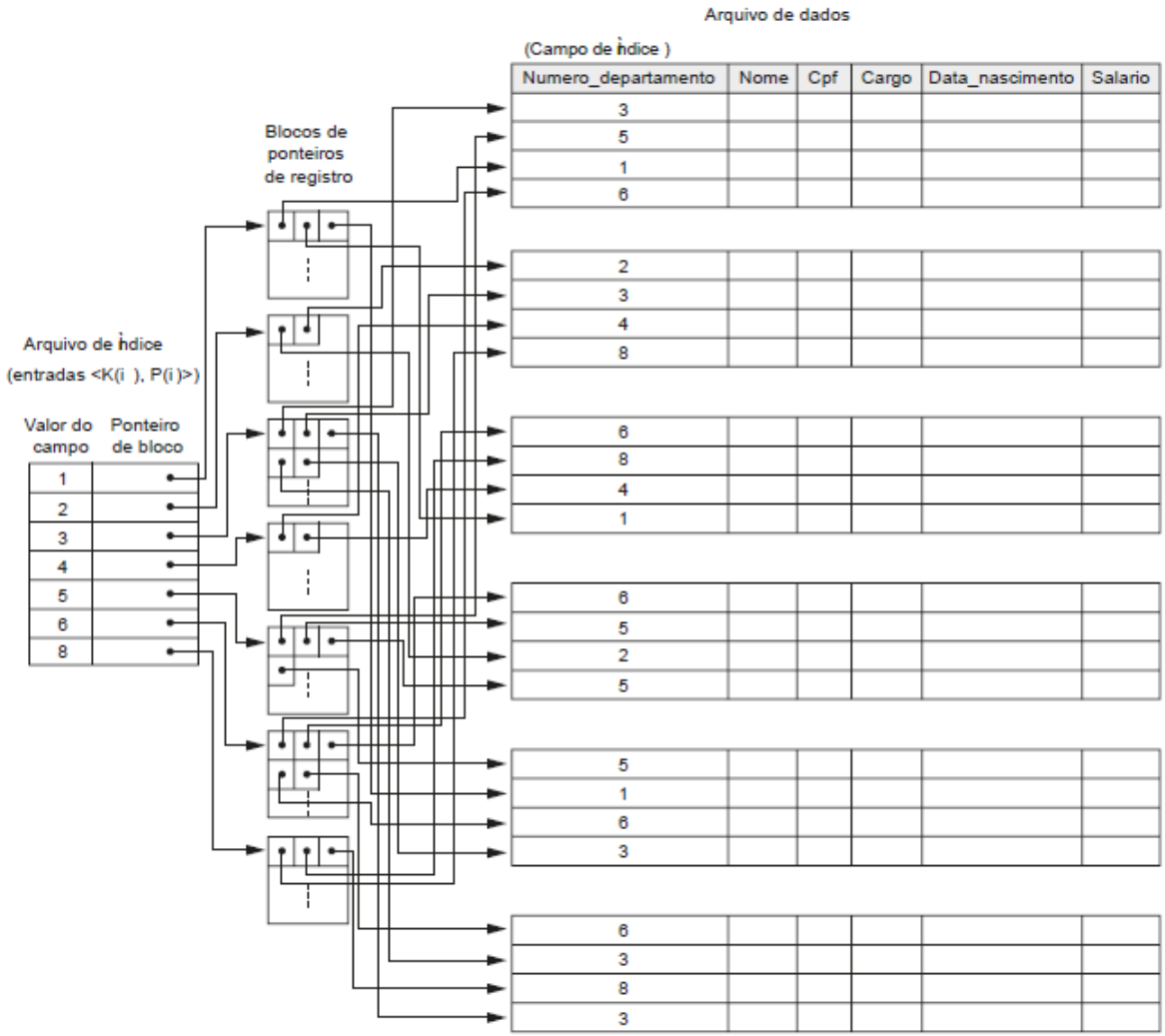
OPÇÃO 3

- O ponteiro *P(i)* na entrada de índice <*K(i)*, *P(i)*> aponta para um bloco de disco, que contém um *conjunto de ponteiros de registro*.
- Cada ponteiro de registro aponta para um dos registros de arquivo com valor *K(i)* para o campo de índice.
- Se algum valor *K(i)* ocorrer em muitos registros, um cluster ou lista ligada de blocos é utilizada.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)



Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 3

- A recuperação por meio do índice requer um ou mais acessos de bloco adicionais.
- Os algoritmos para pesquisar o índice e para inserir novos registros no arquivo são simples.
- Recuperações em condições de seleção complexas podem ser tratadas referindo-se aos ponteiros de registro.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

- Um índice secundário oferece uma **ordenação lógica** nos registros pelo campo de índice.
- Os índices primário e de agrupamento assumem que o campo utilizado para a **ordenação física** dos registros no arquivo é o mesmo que o campo de índice.

Tipos de índices ordenados de único nível

Tipos de índices baseados nas propriedades do campo de índice:

	Campo de índice p/ ordenação física do arquivo	Campo de índice não usado p/ ordenação física do arquivo
Campo de índice é chave	Índice primário	Índice secundário (chave)
Campo de índice é não chave	Índice de agrupamento	Índice secundário (não chave)

Tipos de índices ordenados de único nível

Índices baseados nas propriedades do campo de índice:

Tipo de índice	Número de entradas de índice (primeiro nível)	Denso ou não denso (esparso)	Ancoragem de bloco no arquivo de dados
Primário	Número de blocos no arquivo de dados	Não denso	Sim
Agrupamento	Número de valores de campo de índice distintos	Não denso	Sim/não
Secundário (chave)	Número de registros no arquivo de dados	Denso	Não
Secundário (não chave)	Número de registros ou de valores de campo de índice distintos	Denso ou não denso	Não

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Índices multiníveis

Índices multiníveis

- Implementam uma extensão da ideia de pesquisa binária com uma técnica mais eficiente.
- O objetivo é reduzir mais rapidamente o espaço de pesquisa.
- Para isso, se reduz a parte do índice que continuamos a pesquisar por fator de bloco para o índice (bfr_i), que é maior que 2.

Índices multiníveis

- O valor bfr_i é chamado de **fan-out** do índice multinível (*fo*).
- O *espaço de pesquisa de registro* é dividido *n* vezes (onde *n* = o fan-out) a cada passo de pesquisa, usando o índice multinível.
- A pesquisa em um índice multinível requer logfobi acessos de bloco.

Índices multiníveis

Primeiro nível (ou base) do índice multinível

- Arquivo de índice como um ***arquivo ordenado*** com um ***valor distinto*** para cada *K(i)*.
- Ou seja, consiste em um arquivo de dados classificado.

Índices multiníveis

Segundo nível do índice multinível

- Índice primário para o primeiro nível.
- Pode-se usar âncoras de bloco de modo que o segundo nível tenha uma entrada para ***cada bloco*** do primeiro nível.
- Todas as entradas de índice são do mesmo tamanho (valor de campo e um endereço de bloco).
- O fator de bloco bfr_i para o segundo nível (e para os subsequentes) é o mesmo daquele para o índice do primeiro nível.

Índices multiníveis

Segundo nível do índice multinível

- Se o primeiro nível tiver r_1 entradas e o fator de bloco (fan-out) para o índice for $bfr_i = fo$;
- então o primeiro nível precisará de $[(r_1/fo)]$ blocos.
- Portanto, o número de entradas r_2 necessárias no segundo nível do índice.

Terceiro nível do índice multinível

- Índice primário para o segundo nível.
- Possui uma entrada para cada bloco do segundo nível, de modo que o número de entradas do terceiro nível é $r_3 = [(r_2/fo)]$.

Índices multiníveis

- Exige-se um segundo nível se o primeiro nível precisar de mais de um bloco de armazenamento de disco.
- Exige-se um terceiro nível somente se o segundo nível precisar de mais de um bloco.

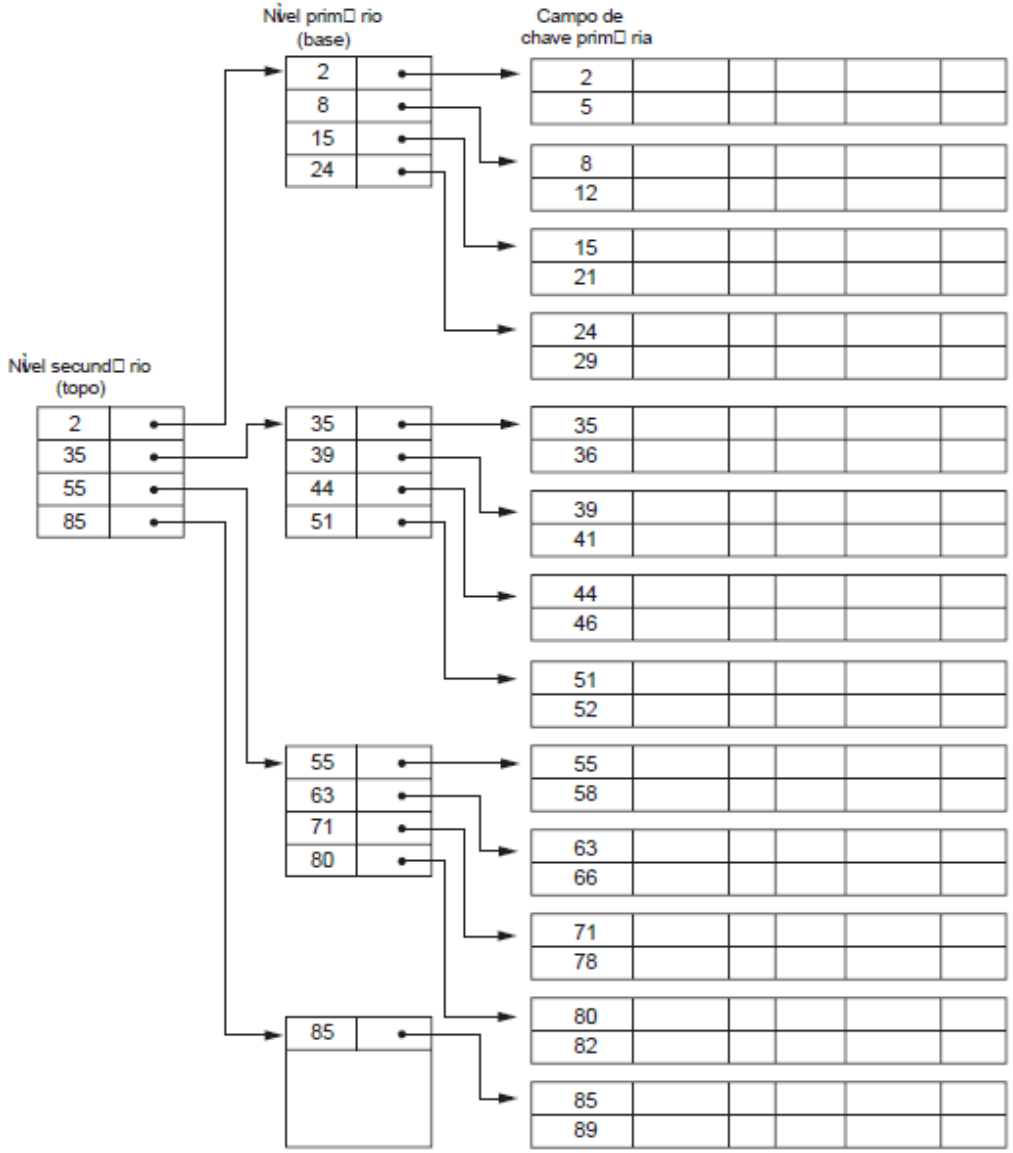
Índices multiníveis

- Bloco no t-ésimo nível.
- Cada nível reduz o número de entradas nos níveis anteriores por um fator de *fo*.
- Pode-se usar a fórmula $1 <= (r_1/((fo)t))$ para calcular *t*.
- Portanto, um índice multinível com r_1 entradas de primeiro nível terá aproximadamente *t* níveis, onde $t = [(\log_{fo}(r_1))]$.

Índices multiníveis

- Ao pesquisar o índice, um único bloco de disco é recuperado em cada nível.
- Logo, t blocos de disco são acessados para uma pesquisa de índice, onde i é o número de níveis de índice.
- Desde que o índice de primeiro nível tenha ***valores distintos para K(i) e entradas de tamanho fixo***, o esquema multinível pode ser usado em qualquer tipo de índice (primário, de agrupamento, ou secundário).

Índices multiníveis



Índices multiníveis

- Pode-se também ter um índice primário multinível não denso.
- Nesse caso, *temos* de acessar o bloco de dados do arquivo antes de podermos determinar se o registro sendo pesquisado está no arquivo.
- Para um índice denso, isso pode ser determinado acessando o primeiro nível de índice (sem ter de acessar um bloco de dados).
- Já que existe uma entrada índice para *cada* registro no arquivo.

Índices multiníveis

Arquivo sequencial indexado

- Organização de arquivo comum usada no processamento de dados comercial.
- É um arquivo ordenado com um índice primário multinível em seu campo de chave de ordenação.

Índices multiníveis

- Um índice multinível reduz o número de blocos acessados quando se pesquisa um registro, dado seu valor de campo de indexação.
- Como todos os níveis de índice são *arquivos ordenados* fisicamente ainda ocorre problemas em inserções e exclusões de índice.

Índices multiníveis

Índice multinível dinâmico

- Deixa espaço em cada um de seus blocos para inserir novas entradas.
- Usa algoritmos apropriados de inserção/exclusão para criar e excluir novos blocos de índice.
- Normalmente implementado ao usar estruturas chamadas B-trees e B+-trees.

Índices multiníveis dinâmicos usando B-trees e B+-trees

Árvore

- Uma **árvore** é formada de **nós**.
- Cada nó (exceto pela **raiz**) tem um nó **pai**.
- O nó raiz não tem pai.
- Cada nó tem zero ou mais nós **filhos**.
- Um nó que não possui filhos é denominado **folha**.
- Um nó não folha é chamado de nó **interno**.
- O **nível** de um nó é sempre um a mais que o nível de seu pai (nível do nó raiz é *zero*).

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Índices de multiníveis dinâmicos usando B-trees e B^+-trees

Subárvore

- Uma subárvore de um nó consiste nesse nó e todos os seus nós **descendentes**.
- Definição recursiva: Uma subárvore consiste em um nó n e as subárvores de todos os nós filhos de n.

Árvores de pesquisa

- Tipo especial de árvore utilizada para orientar a pesquisa por um registro, a partir do valor de um dos seus campos.
- Ligeiramente diferente de um índice multinível.

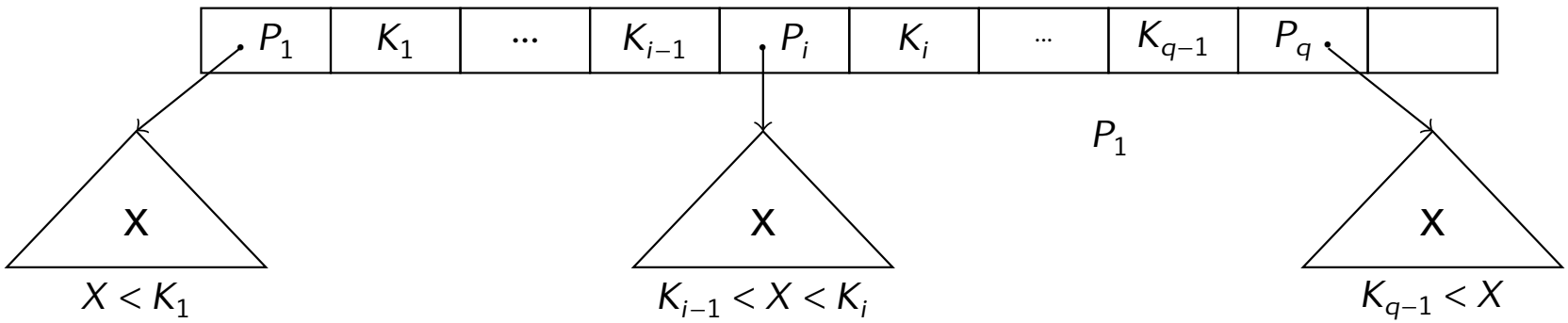
Árvores de pesquisa

- Em uma **árvore de pesquisa de ordem *p*** cada nó contém no ***máximo p - 1 valores*** de pesquisa;
- E ***p*** ponteiros na ordem <P_1, K_1, P_2, K_2, ..., P_{q-1}, K_{q-1}, P>, onde q <= p.
- Cada P_i é um ponteiro para um nó filho (ou um ponteiro NULL);
- Cada K_i é um valor de pesquisa de algum conjunto ordenado de valores.

Árvores de pesquisa

Restrições:

- Em cada nó, $K_1 < K_2 < ... < K_{q-1}$



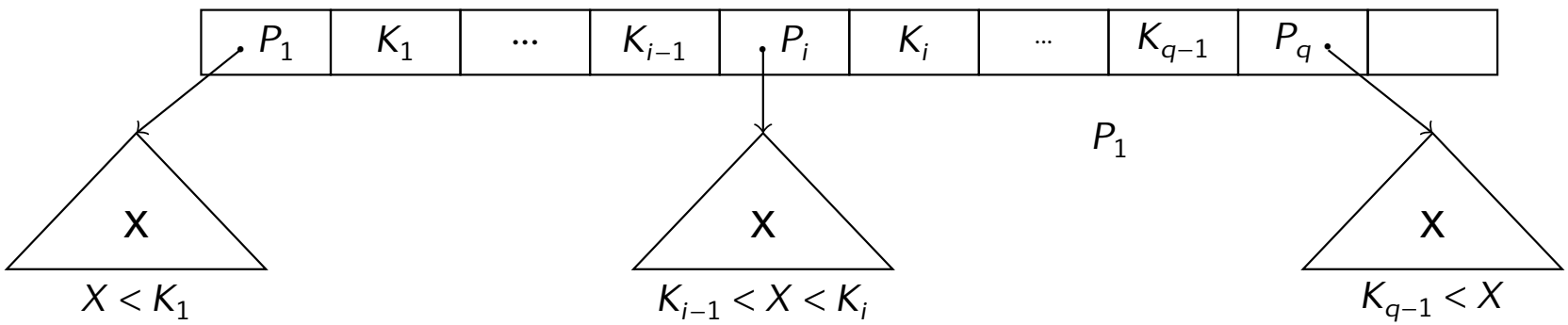
Um nó em um árvore de pesquisa com ponteiros para subárvores abaixo dela.

Árvores de pesquisa

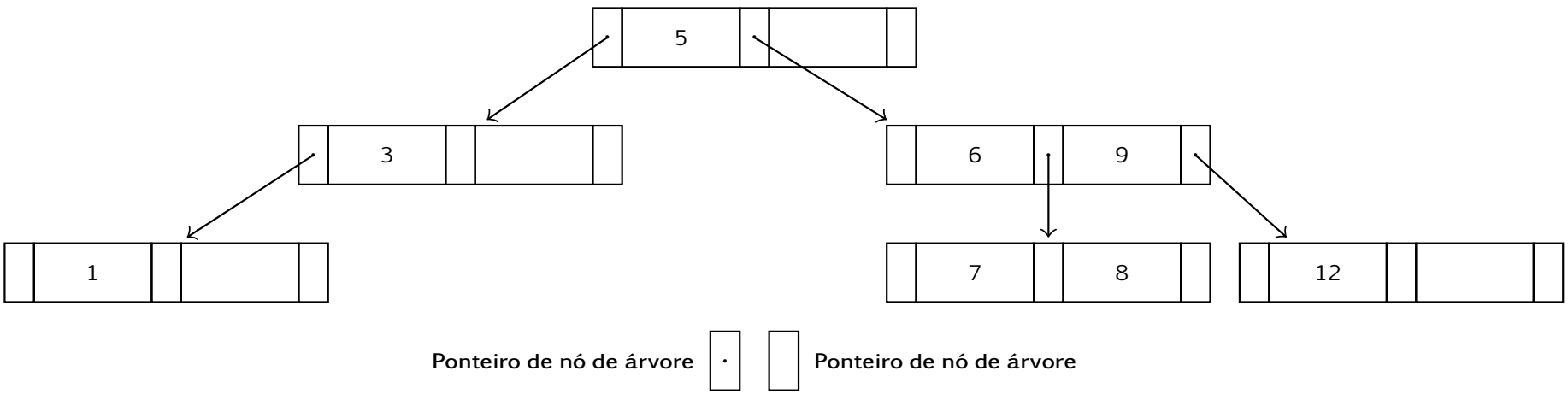
Restrições:

- Para os valores X na subárvore, temos $K_{i-1} < X < K_i$ para $1 < i < q$;
- $X < K_i$ para $i = 1$;
- e $K_{i-1} < X$ para $i = q$.

Ao procurar um valor X, segue-se o ponteiro P_i apropriado, de acordo com as condições acima.



Árvores de pesquisa



Uma árvore de pesquisa de ordem $p = 3$ e valores de pesquisa inteiros (alguns dos ponteiros P_i em nós podem ser NULL).

Árvores de pesquisa

- Podemos usar uma árvore de pesquisa para procurar registros armazenados em um arquivo de disco.
- Os valores na árvore podem ser os valores de um dos **campos de pesquisa** do arquivo.
- Cada valor de chave na árvore é associado a um ponteiro para o registro no arquivo (ou bloco de disco) que tem esse valor.
- A própria árvore de pesquisa pode ser armazenada no disco ao atribuir cada nó a um bloco de disco.

Árvores de pesquisa

- São necessários algoritmos para inserir e excluir valores de pesquisa na árvore (para se manter as restrições).
- Em geral, esses algoritmos não garantem que uma árvore de pesquisa seja **balanceada**.
- Os objetivos para balancear uma árvore de pesquisa são:
  - Garantir que os nós sejam distribuídos por igual, de modo que a profundidade da árvore seja minimizada e que a árvore não fique distorcida.
  - Tornar a velocidade de pesquisa uniforme.

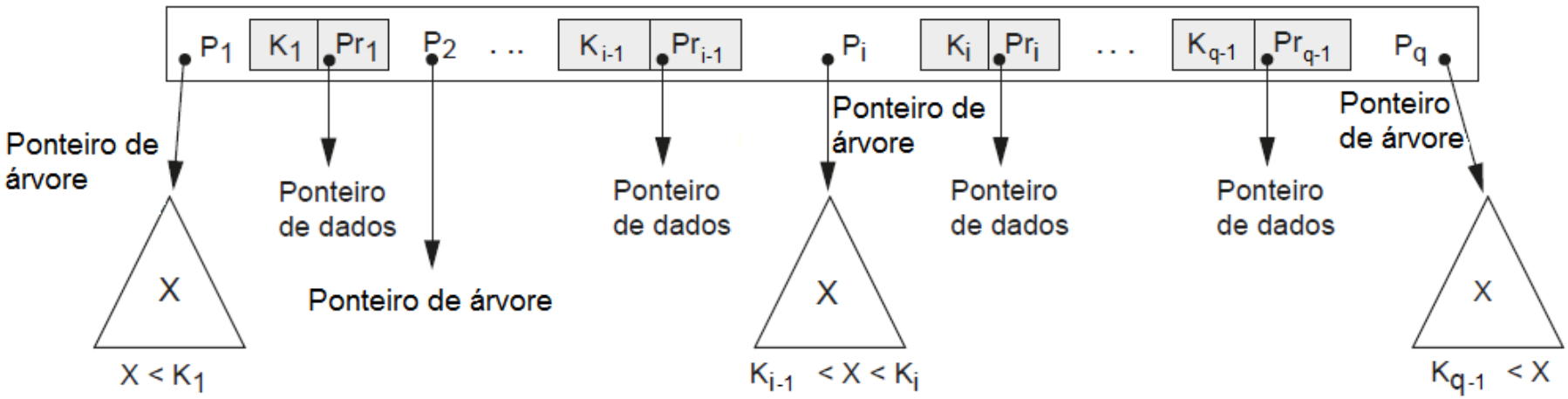
Índices de multiníveis dinâmicos

B-trees

- Restrições adicionais:
  - A árvore deve estar sempre balanceada;
  - O espaço desperdiçado pela exclusão nunca é excessivo.
- Estrutura de acesso multinível;
  - Estrutura de árvore balanceada em que cada nó está cheio pelo menos até a metade.
- Estrutura básica para B-tree de ordem **p** (estrutura de acesso de um campo chave).

Índices de multiníveis dinâmicos

B-trees



Índices de multiníveis dinâmicos

B-trees

- Cada nó interno tem a forma:

  $<P_1, <K_1, Pr_1>, P_2, <K_2, Pr_2>, ..., <K_{q-1}, Pr_{q-1}<, P_q>$

  onde:

  - $q <= p$
  - P_i: ponteiro de árvore (para outro nó);
  - Pr_i: ponteiro de dados (para o registro cujo valor do campo chave é igual a K_i);

  - Máximo de p ponteiros;
  - Máximo de $p-1$ ponteiros de dados;
  - Máximo de $p-1$ valores de campo de chave;

  - $K_1 < K_2 < ... < K_{q-1}$

Índices de multiníveis dinâmicos

B-trees

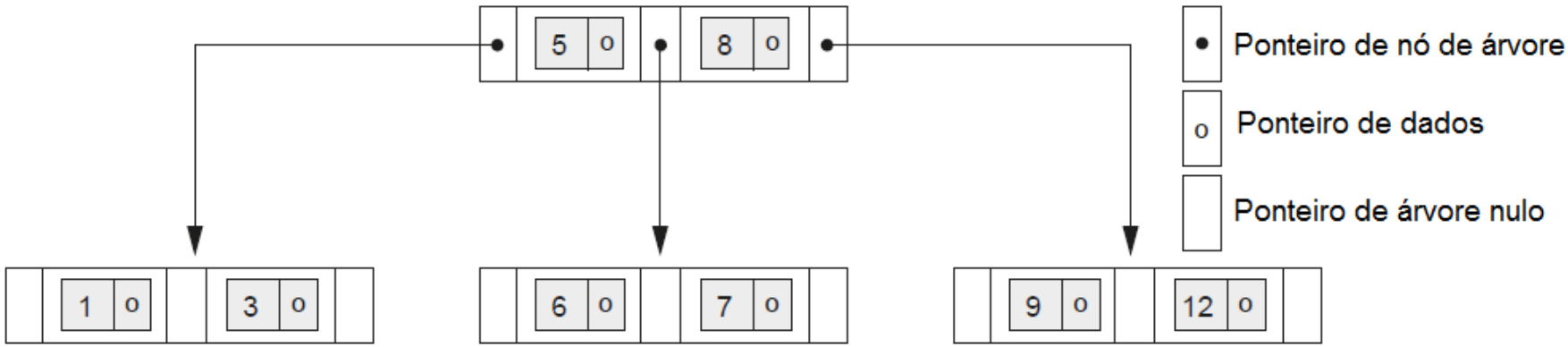
- Para todos os valores de campo da chave de pesquisa **X**, na subárvore apontada por P_i, temos:

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

  onde:

  - Cada nó tem no máximo **p** ponteiros de árvore e, ao menos **p/2** ponteiros de árvore (exceto os nós raiz e folha);
  - Um nó com **q** ponteiros de árvore (sendo **q <= p**) tem **q-1** valores de campo chave de pesquisa.

B-trees - Exemplo



- Observe:
  - p = 3
  - Todos os valores de pesquisa K são únicos.

Índices de multiníveis dinâmicos

B-trees

- Procedimentos para inserção:
  1. Começa com um único nó raiz, no nível 0;
  2. SE o nó raiz está cheio com **p-1** valores de chave de pesquisa:
     - Se divide em dois de nível 1;
     - Somente o valor do meio é mantido no nó raíz.
  3. SENÃO:
     - Se divide em dois nós no mesmo nível;
     - A entrada do meio é movida para o nó pai junto com os ponteiros aos divididos;
     - Se o nó pai estiver cheio, este também é dividido;
       A divisão se propaga até o nó raiz.

B-trees

- Se a exclusão de um valor fizer que um nó fique com menos da metade cheio, este é combinado com seus nós vizinhos;
  - Pode reduzir o número de níveis da rede.
- Os nós ficam aproximadamente 69% cheios quando o número de valores na árvore se estabiliza;
  - A divisão e a combinação de nós ocorrerão raramente, de modo que a inserção e a exclusão se tornam muito eficientes.

B-trees - Ex (Cálculo do número de blocos e níveis):

- A pesquisa será por um campo de chave não ordenado, em uma B-tree com **p = 23**. Cada nó está 69% cheio, portanto, cada nó terá, em média, **p * 0,69 = 23 * 0,69**, ou aproximadamente **16 ponteiros**, ou seja, 15 valores de campo chave de pesquisa.
    - Começamos na raíz, verificando quantos ponteiros podem existir, na média, em cada nível subsequente.

B-trees - Ex (Cálculo do número de blocos e níveis):

Raíz	1 nó	15 entradas	16 ponteiros
Nível 1	16 nós	240 entradas	256 ponteiros
Nível 2	256 nós	3.840 entradas	4.096 ponteiros
Nível 3	4.096 nós	61.440 entradas	

Uma B-tree de três níveis mantém 65.536 entradas de chave na média.

Índices de multiníveis dinâmicos

B^+-trees

- Variação da estrutura de dados da B-tree;
- Utilizado na maioria das implementações de um índice multinível dinâmico;
- Os ponteiros de dados são armazenados apenas nos **nós folha** da árvore.

B$^+$-TREES

- Nós folha:
  - Armazenam todos os ponteiros de dados;
  - Uma entrada para cada valor do campo de pesquisa;
  - SE o campo de pesquisa for chave
    Também contém o ponteiro de dados para o registro.
    SENÃO
    Também contém o ponteiro que aponta para o bloco com os ponteiros para os registros do arquivo de dados.
  - São ligados para oferecer acesso ordenado no campo de pesquisa;
  - Alguns valores do campo de pesquisa são repetidos para os nós internos para guiar a pesquisa.

B^+-trees

- Nós folha - forma:

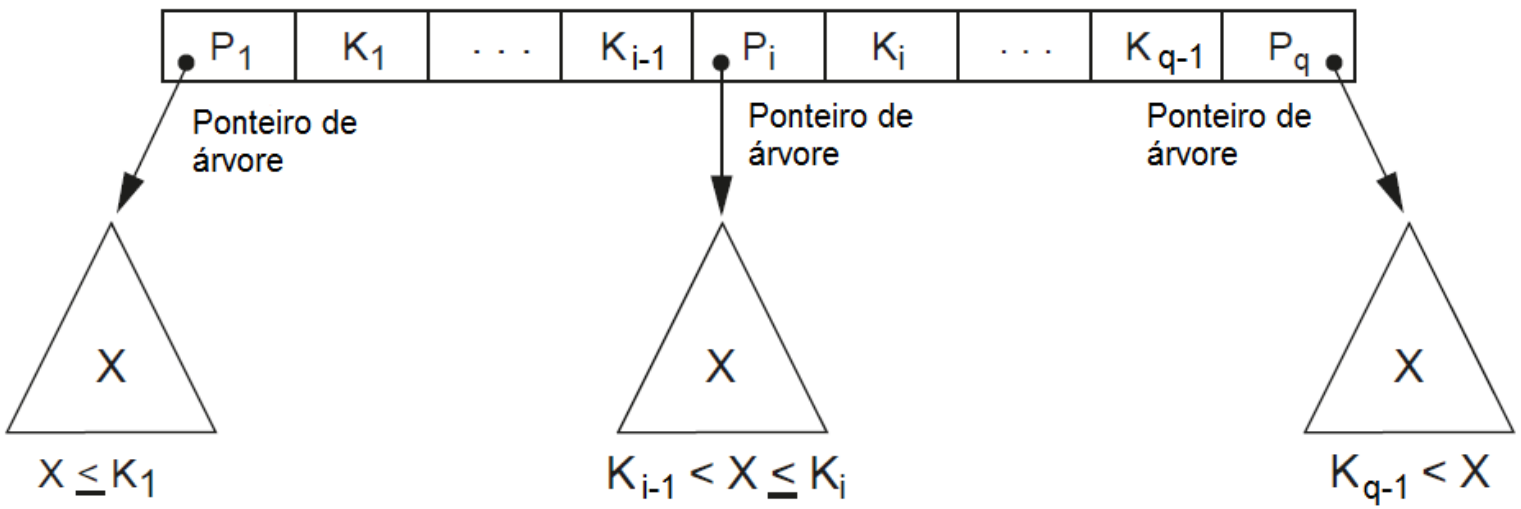
  $< <K_1, Pr_1>, <K_1, Pr_2>, ..., <K_{q-1}, Pr_{q-1}>, P_{proximo}>$

  onde **q** <= **p**, cada Pr_i é um ponteiro de dados e $P_{proximo}$ aponta para o próximo nó folha da B^+-tree.

  - $K_1 <= K_2 ... , K_{q-1}$ em que **q** <= **p**;
  - **Pri**: ponteiro de dados - aponta para o registro cujo valor do campo pesquisa é K_i ou para um bloco de arquivo que contém registro;
  - Tem, ao menos, **p/2** valores;
  - Todos os nós folha estão no mesmo nível.

B$^+$-TREES

- Nós internos:
  - Correspondem aos outros níveis de índice multinível.
  - Estrutura básica para B$^+$-tree de ordem **p**:



Índices de multiníveis dinâmicos

B^+-trees

- Nós internos - forma:

$$<P_1, K_1, P_2, K_2, ..., P_{q-1}, K_{q-2}, P_q>$$

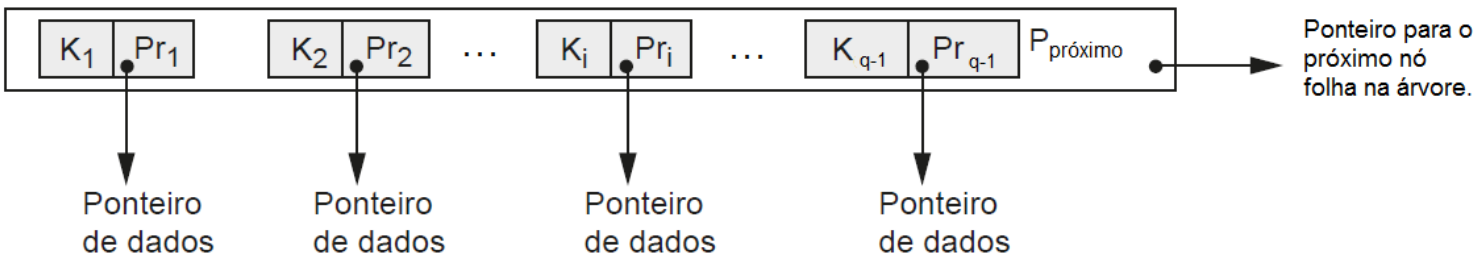
- onde $q<=p$ e cada P_i é um ponteiro de árvore.
  - $K1 < K2 < ... < Kq-1$.
  - Para todos os valores de campo de pesquisa X na subárvore apontada por Pi, temos:

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

Índices de multiníveis dinâmicos

B^+-trees

- Nós internos - forma:
  - Tem, no máximo, **p** ponteiros de árvore;
  - Exceto a raíz, tem pelo menos **p/2** ponteiros de árvore;
  - Um nó interno com **q** ponteiros (**q <= p**) tem **q - 1** valores de campo de pesquisa.
- Observe:



B^+-trees

- Ao começar a leitura no nó folha mais Ãă esquerda, é possível atravessar os nós folha como uma lista, usando os ponteiros $P_{proximo}$;
  - Acesso ordenado aos registros de dados no campo de índice.
- Pode ser incluído um ponteiro $P_{anterior}$ (adicional);
- Comparações entre **B^+-tree** com **B-tree**:
  - Mais entradas podem ser compactadas em um nó interno de uma B^+-tree do que para uma B-tree semelhante;

B^+-TREES

- Comparações entre **B^+-tree** com **B-tree**:
  - Para o mesmo tamanho de bloco, a ordem **p** será maior para a B^+-tree do que para a B-tree;
  - Proporciona menos níveis a B^+-tree, otimizando seu tempo de pesquisa.
- A seguir, **p** será usado para indicar a ordem dos nós internos e $\mathbf{p_{folha}}$ para indicar a ordem para os nós folha.

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - O campo chave da pesquisa é **V = 9 bytes**, o tamanho do bloco é **B = 512 bytes**, o ponteiro de registro seja **Pr = 7 bytes** e o ponteiro de bloco seja de **6 bytes**.
  - Um nó interno da B^+-tree pode ter até **p** ponteiros de árvore e **p-1** valores de campo de pesquisa - estes precisam caber em um único bloco.

    Temos:

    (p * P) + ((p-1) * V) <= B
    (P * 6) + ((p-1) * 9) <= 512
    (15 * p) <= 512
    p <= 34

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - O valor encontrado em **p** é maior do que o valor 23 para a B-tree, resultando em um *fun-out* maior e mais entradas em cada nó interno de uma B^+-tree do que na B-tree correspondente.
  - Os nós folha da B^+-tree terão o mesmo número de valores e ponteiros, exceto que os ponteiros são de dados e de ponteiro próximo.

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - A ordem p_{folha} pode ser calculada:

$(p_{folha} * (Pr * V)) + P <= B$
$(pfolha * (7+9) + 6 <= 512$
$(16 * p_{folha}) <= 506$
$p_{folha} <= 31$

Conclui-se que cada nó folha poderá manter até $\mathbf{p_{folha} = 31}$ combinações de ponteiro de valor/dados de chave.

B$^+$-TREES

- Para implementar algoritmos de inserção e exclusão, será necessário inserir *informações adicionais* em cada nó:
  - Tipo de nó (interno ou folha);
  - Número de entradas atuais q no nó;
  - Ponteiros para os nós pai e irmão.
- Antes de proceder com os cálculos para **p** e $\mathbf{p_{folha}}$, se faz necessário reduzir o tamanho do bloco pela quantidade de espaço necessária para toda a informação.

B^+-trees - Ex (Cálculo número de entradas):

- No cálculo do número aproximado de entradas, considera-se que cada nó esteja 69% cheio. Em média, cada nó interno terá **34 * 0,69** ou, aproximadamente, 23 ponteiros e 22 valores.

- Cada nó folha manterá **0,36 * p_{folha} = 0,69 * 31 = 21** ponteiros de registro de dados.

B^+-trees - Ex (Cálculo número de entradas):

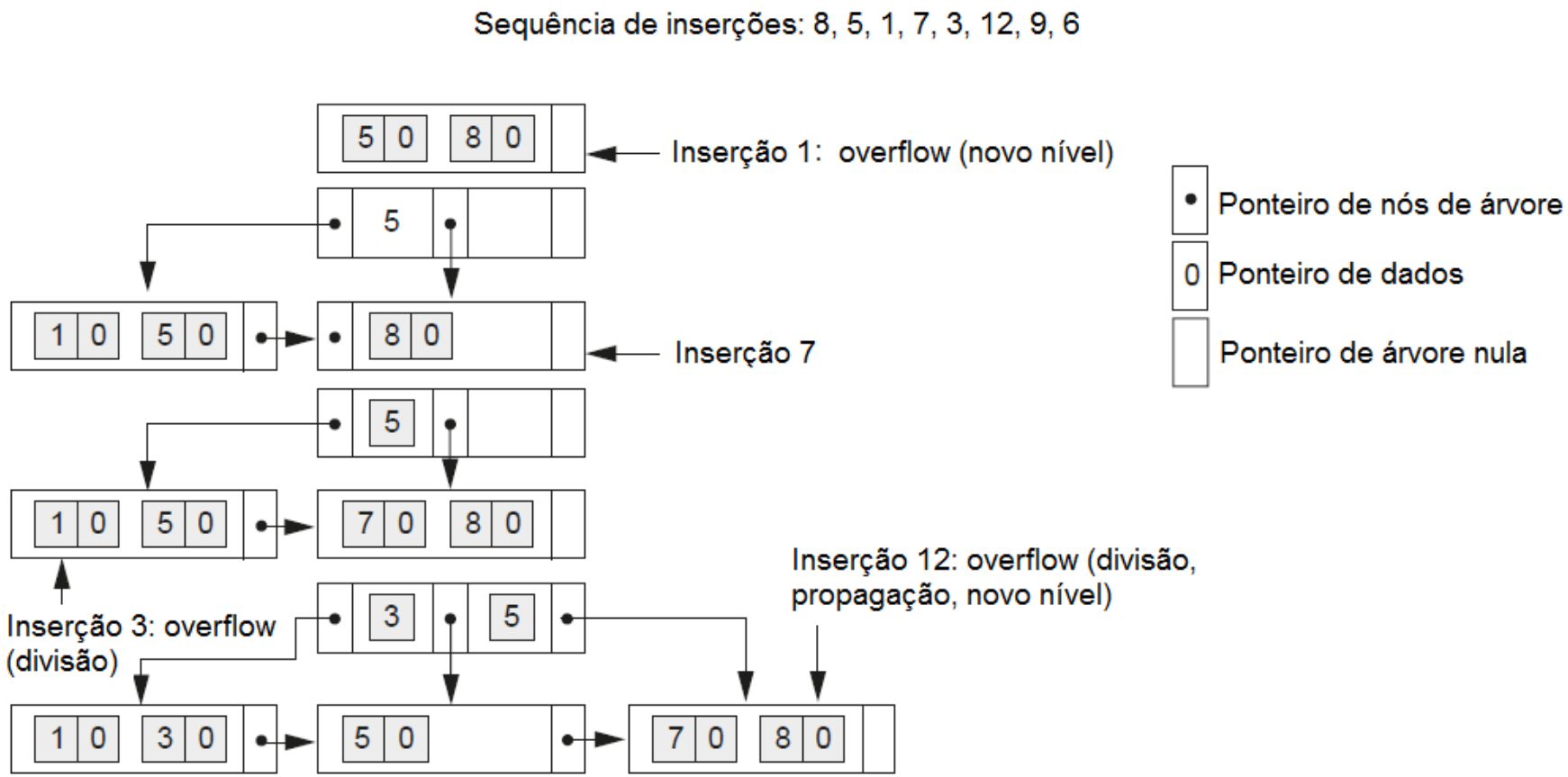
- Número médio de entradas para cada nível:

Raíz	1 nó	22 entradas	23 ponteiros
Nível 1	23 nós	506 entradas	529 ponteiros
Nível 2	529 nós	11.638 entradas	12.167 ponteiros
Nível folha	12.167 nós	255.507 entradas	

- Conclusão:
  - Para o tamanho do bloco, do ponteiro e do campo de pesquisa, uma B^+-tree de três níveis mantém até 255.507 ponteiros de registro, com a média de 69% de ocupação de nós.

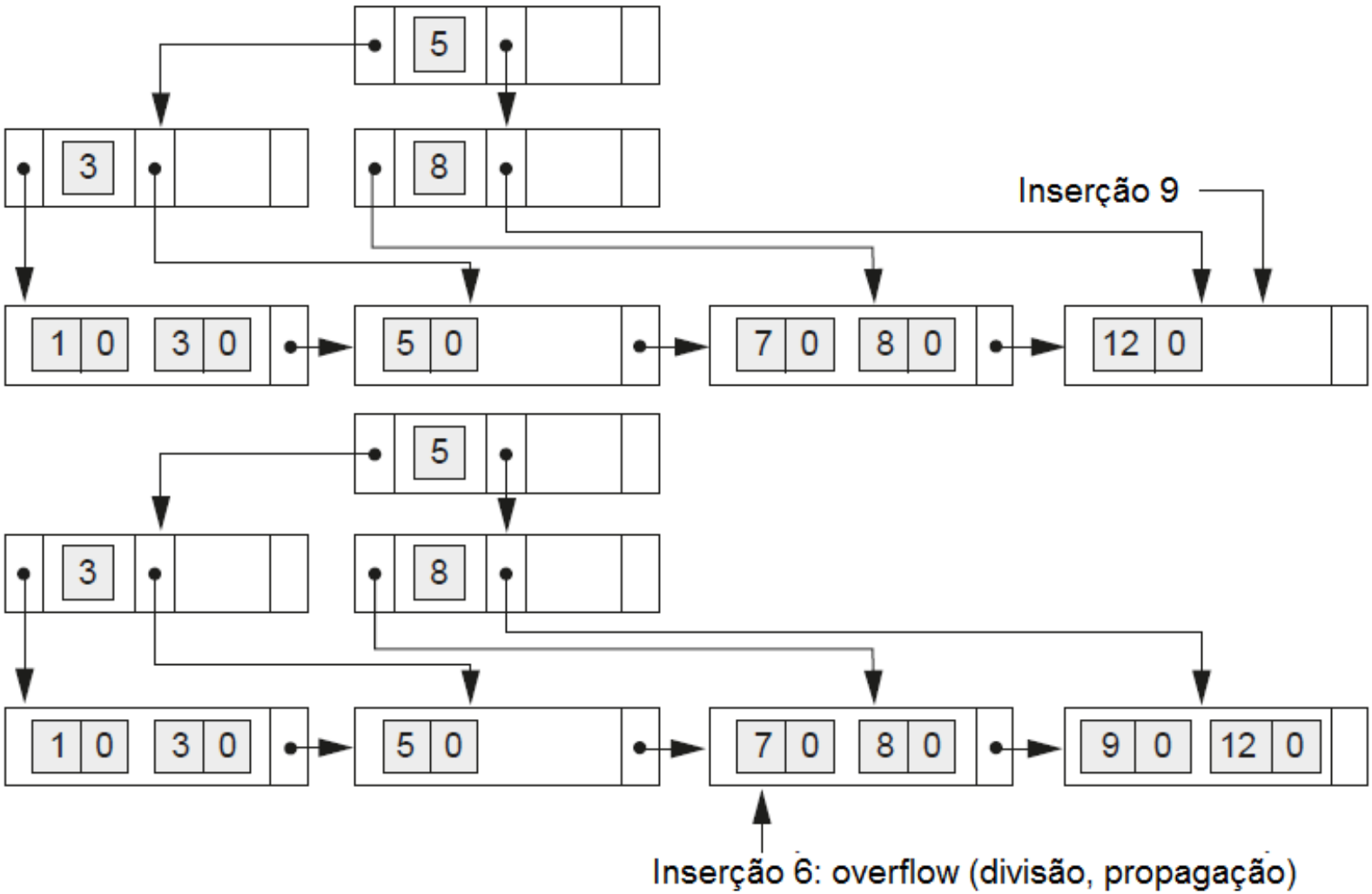
B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



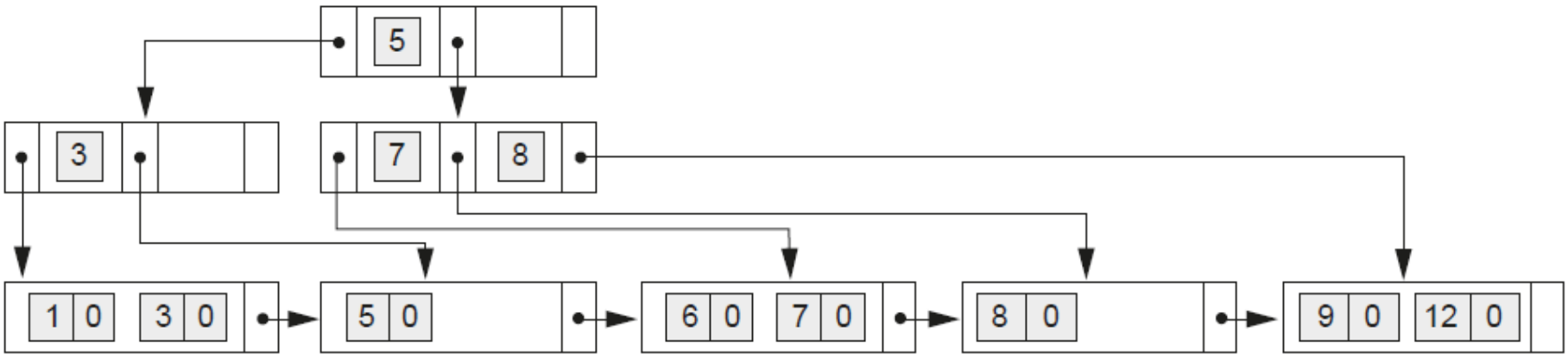
B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B^+-TREES - INSERÇÃO DE REGISTROS:

Ordem $p=3$ e $p_{folha}=2$

- Observe:
  - A raíz é o único nó da árvore que também é um nó folha;
  - Assim que mais de um nível é criado, a árvore é dividida em nós internos e nós folha;
  - Cada valor de chave precisa existir no nível de folha, por conta dos ponteiros;
  - Somente alguns valores existem nos nós internos para guiar a pesquisa;

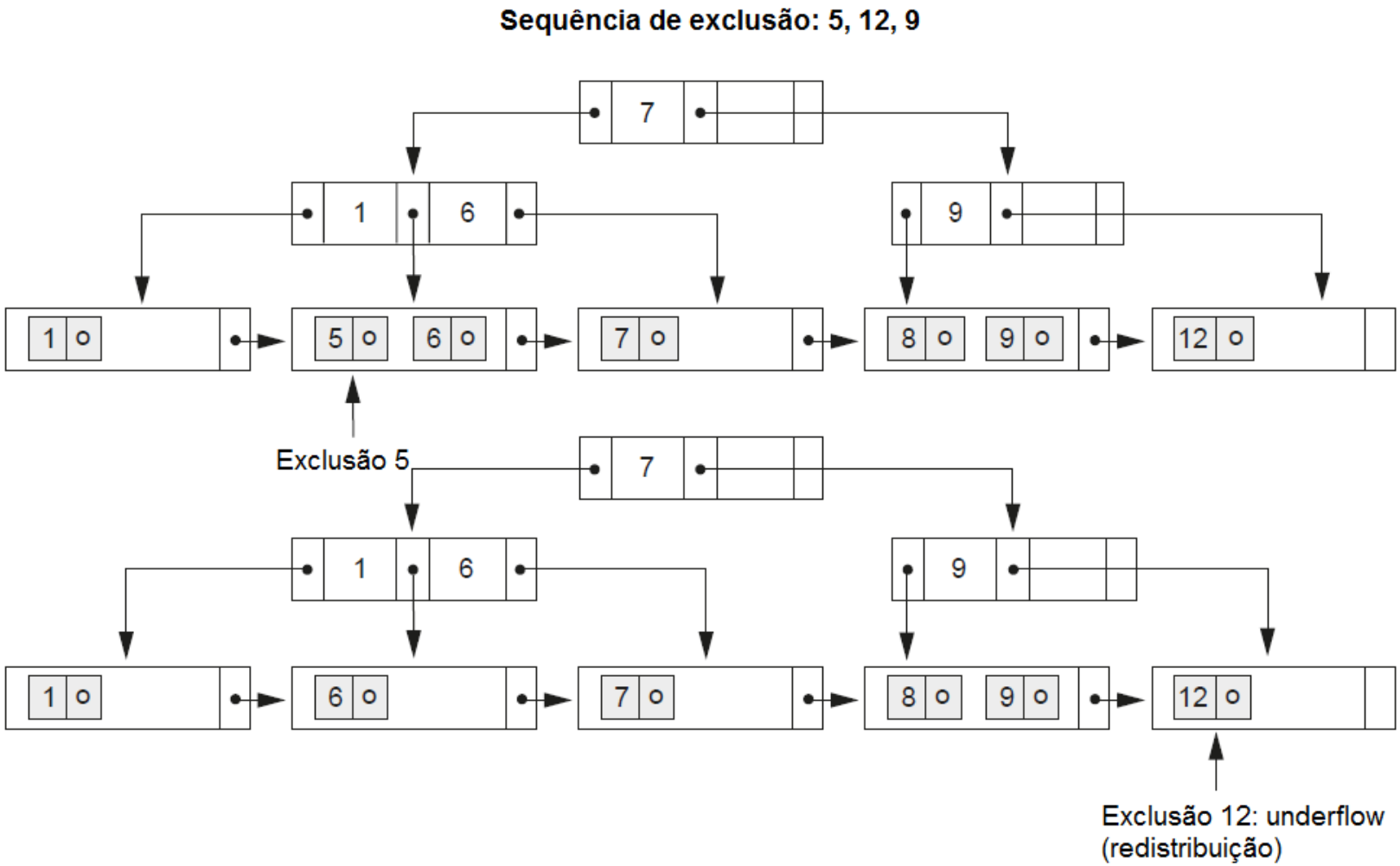
B^+-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2

- Observe:
  - Necessário dividir o nó folha quando é inserida uma nova entrada e este estiver cheio;
  - As primeiras entradas $i = [((p_{folha} + 1)/2)]$ no nó original são mantidas e as entradas restantes são movidas para um novo nó folha;
  - O *i*-ésimo valor de pesquisa é replicado no nó interno do pai, e um ponteiro para este é criado;
  - Um novo nó interno manterá as entradas de P_{i+1} para o final das entradas no nó.

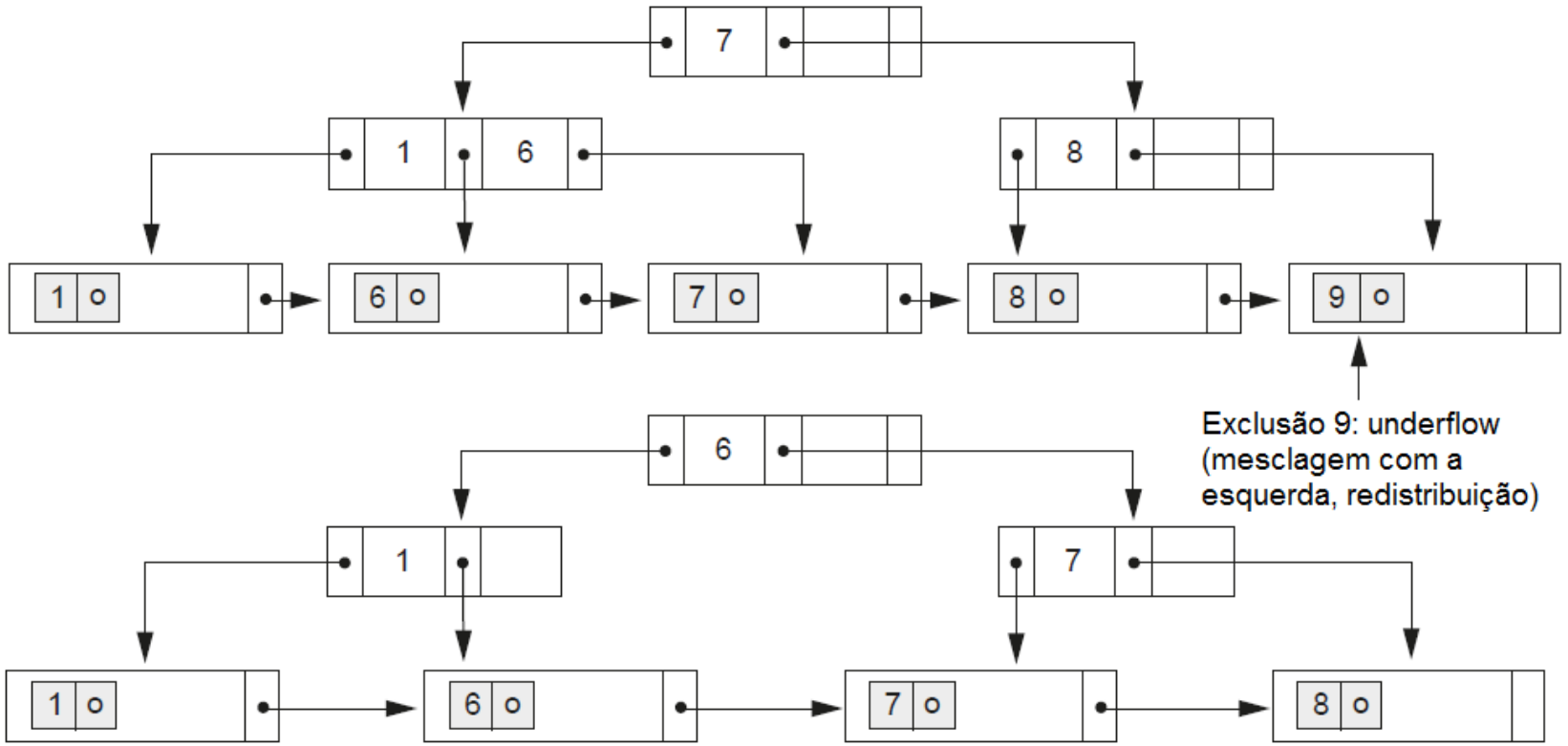
B$^+$-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B$^+$-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B$^+$-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2

- Observe:
  - Quando uma entrada é excluída, ele é sempre removida do nível folha;
  - A exclusão pode gerar *overflow*, reduzindo o número de entradas no nó folha abaixo do mínimo exigido;
  - Tenta-se encontrar um nó folha "irmão"e redistribuir suas entradas, os deixando cheios até a metade;
  - Caso contrário, os nós são mesclados é o número de folhas reduzido.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

ÍNDICES EM CHAVES MÚLTIPLAS

Índices em chaves múltiplas

- Solicitações de atualização e recuperação frequente com a mesma combinação de atributos.
  - Configura-se uma estrutura de acesso para oferecer acesso eficaz.
- Exemplo:

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Idade

Índices em chaves múltiplas

- Considere: Listar os funcionários no **departamento número 4**, cuja **idade é 59**.
  - **Dnr** nem **Idade** são atributos não chave, ou seja, poderá apontar para vários registros.
  - Estratégias:
    1. Se **Dnr** for indexado, mas **Idade** não: acessar os registros **Dnr = 4**, usando o índice, e depois selecionar os registros com **Idade = 59**;
    2. Se **Idade** for indexada, mas **Dnr** não: acessar os registros **Idade = 59**, usando o índice, e depois selecionar os registros com **Dnr = 4**;

Índices em chaves múltiplas

- Considere: Listar os funcionários no **departamento número 4**, cuja **idade é 59**.
  - Estratégias:
    3. Se os índices forem criados sobre **Dnr** e **Idade**: cada registro contém um conjunto de registros ou ponteiros. Utiliza-se uma interseção entre esses conjuntos para obter o resultado.
  - Se o volume de retorno for grande, nenhuma das técnicas será eficiênte.
  - Necessário utilizar as estratégias para tratar a combinação como chave de pesquisa (chaves compostas).

Índices em chaves múltiplas

ÍNDICE ORDENADO EM MÚLTIPLOS ATRIBUTOS

- Se um índice for criado nos atributos $<A_1, A_2, ..., A_n>$, os valores de chave de pesquisa são tuplas com **n** valores: $<v_1, v_2, ..., v_n>$;
- Uma ordenação lexicográfica desses valores estabelece uma ordem na chave de pesquisa composta.

Índices em chaves múltiplas

Hashing particionado

- Extensão do hashing externo estático - permite o acesso a múltiplas chaves;
- Adequado a comparações de igualdade;
- Nenhuma estrutura de acesso separada precisa ser mantida para os atributos individuais;
- Para uma chave que consiste **n** componentes, a função de hash produz um resultado com **n** endereços separados;
- O endereço do bucket é uma concatenação desses **n** endereços.

Hashing particionado - Exemplo

- Chave composta <**Dnr, Idade**>
  - **Dnr**: hash para endereço de 3 bits
  - **Idade**: hash para endereço de 5 bits
  - Bucket de 8 bits (3 bits + 5 bits)

- Se **Dnr = 4** tiver um endereço hash ′100′ e **Idade = 59** tiver um endereço hash ′10101′, na pesquisa dos valores, temos que ir ao endereço:

100 10101

Índices em chaves múltiplas

Arquivos de grade

- É construído um vetor de grade com uma escala (ou dimensão) linear para cada um dos atributos de pesquisa;
- Objetivo de alcançar uma distribuição uniforme dos atributos;
- Cada célula aponta para o endereço de bucket onde os registros correspondentes são armazenados;

Índices em chaves múltiplas

Arquivos de grade

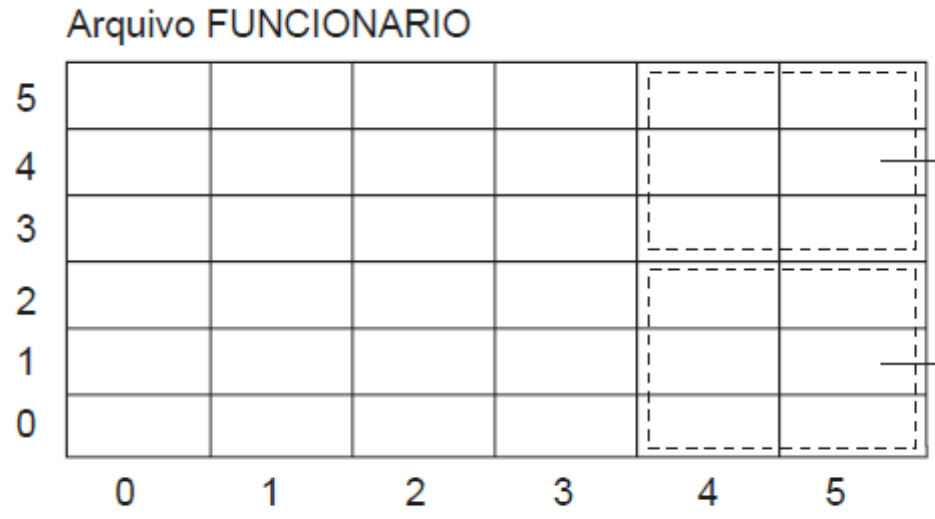
- Adequado a consultas de intervalo (mapeadas para um conjunto de células - grupo de valores ao longo de escalas lineares);
- Aplicado a qualquer quantidade de chaves de pesquisa;
- Apresentam *overhead* de espaço - uma reorganização frequênte do arquivo (aumentando o custo de manutenção).

Arquivos de grade - Exemplo

Dnr

0	1, 2
1	3, 4
2	5
3	6, 7
4	8
5	9, 10

Escala linear para Dnr



Escala linear para Idade

0	1	2	3	4	5
< 20	21-25	26-30	31-40	41-50	> 50

Arquivos de grade - Exemplo

- Vetor de grade para o arquivo FUNCIONARIO com uma escala linear para **Dnr** e outra para o atributo **Idade**;
- A solicitação de **Dnr = 4** e **Idade = 59** é mapeada para a célula **(1,5)**. Os registros para esta combinação serão encontrados no bucket correspondente.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Outros tipos de índices

Outros tipos de índices

Índices de hash

- Estrutura secundária para acessar o arquivo usando hashing em uma chave de pesquisa diferente da que é usada para a organização do arquivo de dados primário;
- Índice físico;
- Entradas de índice:

  tipo <K, Pr> ou <K, P>

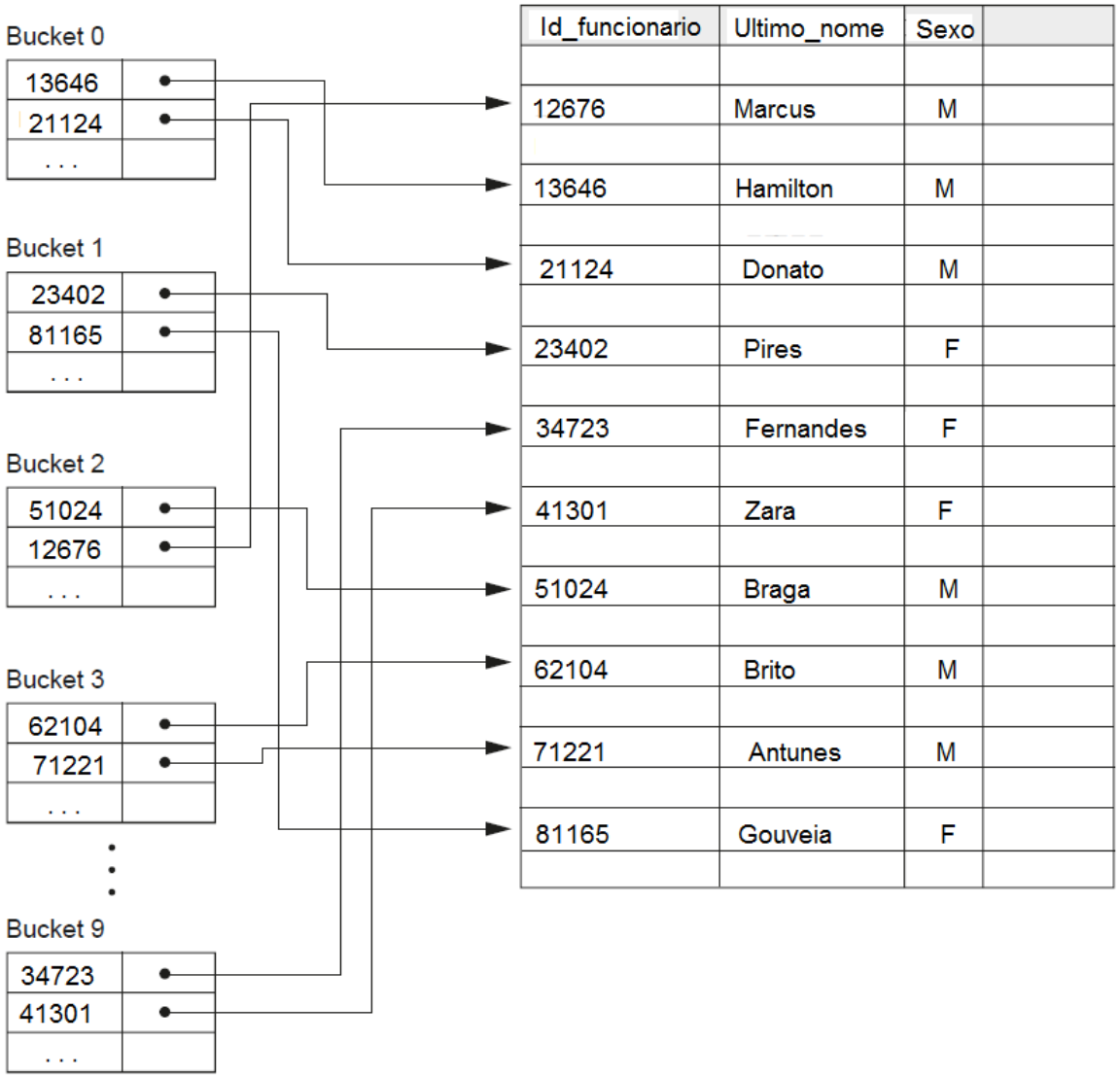
  onde
  - **Pr**: ponteiro para o registro que contém a chave;
  - **P**: ponteiro para o bloco que contém o registro para a chave.

Outros tipos de índices

Índices de hash

- O arquivo de índice pode ser organizado como um arquivo de hash dinamicamente expansível;
- Pesquisa por uma entrada:
  - Algoritmo de pesquisa de hash em K;
  - Quando uma entrada é localizada, o ponteiro Pr é utilizado para localizar o registro no arquivo de dados.

Índices de hash - Exemplo



Outros tipos de índices

Índices de hash - Exemplo

- O **Func_id** passa pelo hashing para obter um número de butcket usando a função de hashing: a soma dos dígitos de **Func_id módulo 10**.
- Encontrando o **Func_id = 51024**:
  1. 5 + 1 + 0 + 2 + 4 = 12
  2. 12 módulo 10 = 2
- Neste caso, o bucket 2 é acessado primeiro. Este contém a entrada de índice **<51024, Pr>**;
- O ponteiro **Pr** nos leva ao registro real no arquivo.

Outros tipos de índices

Índices bitmap

- Estrutura que facilita a consulta em múltiplas chaves;
- Utilizada em relações que contém um grande número de linhas;
- Cria um índice para uma ou mais colunas e cada valor (ou intervalo de valores) é indexado;
- Eficiêntes em relação ao espaço de armazenamento;
- Para criação do índice em um conjunto de registros, estes devem estar devidamente numerados.

Outros tipos de índices

Índices bitmap

- O índice é criado em um valor específico de um campo (a coluna em uma relação) e é apenas um vetor de bits;
- Considere um índice bitmap para a coluna **C** e um valor **V** para esta coluna:
  - Para relação com **n** linhas, contém **n** bits;
  - O *i*-ésimo bit é definido como 1 se a linha **i** tiver o valor **V** para a coluna **C**;
  - Se **C** contiver o conjunto de valores $<v_1, v_2, ..., v_m>$, então **m** índices bitmap seriam criados para a coluna.

Outros tipos de índices

Índices bitmap - Exemplo

- A seguir, a relação FUNCIONARIO com colunas **Func_id, Unome, Sexo, Cep** e **Faixa_salarial** e um índice bitmap para as colunas **Sexo** e **Cep**:

Linha_id	Func_id	Unome	Sexo	Cep	Faixa_salarial
0	51024	Braga	M	09404011	-
1	23402	Pires	F	03002211	-
2	62104	Beto	M	01904611	-
3	34723	Fernandes	F	03002211	-
4	81165	Gouveia	F	01904611	-
5	13646	Hamilton	M	01904611	-
6	12676	Marcus	M	03002211	-
7	41301	Zara	F	09404011	-

Índice bitmap para Sexo:

M: 101001100 - **F:** 01011001

Outros tipos de índices

Índices bitmap - Exemplo

Índice bitmap para Cep:

Cep 01904655: 00101100 - **Cep 03002211:** 01010010 - **Cep 09409433:** 100000001

- Para encontrar funcionários com **Sexo = F** e **Cep = 30022512345**, realiza-se a inserção dos bitmaps ′01011001′ e ′01010010′, resultando nos Linha_id 1 e 3.
- Os funcionários que não moram no **Cep = 09404066** são obtidos pelo complemento do vetor de bits ′10000001′, produzindo Linha_id de 1 a 6.

Outros tipos de índices

Índices bitmap (de existência)

- Evita o gasto de renumeração de linhas e o deslocamento de bits, na exclusão de registros;
- Tem-se um bit 0 para as linhas que foram excluídas mas ainda estão presentes;
- Tem-se um bit 1 para as linhas que realmente existem;
- Nas inserções, uma entrada deve ser criada em todos os bitmaps de todas as colunas que têm índice bitmap.

Outros tipos de índices

ÍNDICES BITMAP (PARA NÓS FOLHA DE B^+-TREE)

- Bitmaps podem ser usados:
  - Em nós folha dos índices de B^+-tree;
  - Para apontar o conjunto de registros que contém cada valor específico do campo indexado no nó folha.
- Quando a B^+-tree está embutida em um campo de pesquisa não chave:
  - O registro folha contém uma lista de ponteiros de registro ao longo de cada valor do atributo indexado;
  - Para os valores frequêntes, um índice bitmap pode ser armazenado em vez de ponteiros.

Outros tipos de índices

Indexação baseada em função

- Cria um índice tal que o valor que resulta da aplicação de alguma função em um campo (ou coleção de campos) torna-se chave para o índice;

- Garantem que o sistema Oracle Database, por exemplo, usará o índice em vez de realizar uma varredura completa na tabela.

Outros tipos de índices

Indexação baseada em função - Exemplo

- Criação de índice na tabela FUNCIONARIO com base na coluna **Unome** (representada em maiúsculo):

```
CREATE INDEX idx_maiusc
ON Funcionario (UPPER(Unome))
```

Cria um índice com base na função UPPER(Unome), que retorna o sobrenome em letras maiúsculas.

UPPER('Silva') retornará '**Silva**'

Outros tipos de índices

Indexação baseada em função - Exemplo

- A consulta abaixo usará o índice:

```
SELECT Primeiro_nome, Unome
FROM Funcionario
WHERE UPPER(Unome) = 'Silva'
```

- Sem o índice baseado em função, um Oracle database poderia realizar uma varredura completa da tabela, pois um índice de B^+-tree só é pesquisado pelo uso direto do valor da coluna.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

QUESTÕES GERAIS SOBRE INDEXAÇÃO

Questões gerais sobre indexação

Índices Lógicos X Físico

- Éndice Físico
  - O ponteiro precisa ser mudado se o registro for movimentado para outro local no disco;
  - Ex: Se a organização de arquivo primária for baseada em hashing linear ou extensível, toda vez que um bucket for dividido, alguns registros serão alocados para novos buckets e terão novos endereços físicos;
  - Se houvesse um índice secundário no arquivo, os ponteiros para esses registros devem ser localizados e atualizados.

Questões gerais sobre indexação

Índices Lógicos X Físico

- índice Lógico
  - As entradas de índice têm a forma **<K, Kp>**;
  - Cada entrada tem um valor **K** para algum valor de campo de índice secundário combinado com o valor de **Kp** (do campo usado para a organização do arquivo primário);
  - Ao pesquisar o índice secundário no valor de **K**, um programa pode localizar o valor correspondente e acessar o registro pela organização do arquivo primário.

Questões gerais sobre indexação

Índices Lógicos X Físico

- Índice Lógico
  - Introduzem um nível adicional de indireção entre a estrutura de acesso e os dados;
  - São usados quando se espera que os endereços de registro físicos mudem com frequência.

Estrutura de acesso

- O índice não faz parte integral do arquivo de dados;
- Pode ser criado e descartado dinamicamente.

Questões gerais sobre indexação

Índice Secundário

- Evitar a ordenação física dos registros no arquivo de dados em disco;
- Podem ser criados junto com qualquer organização de registro primária;
- Criação em B^+-tree:
  - Percorre todos os registros no arquivo para criar as entradas no nível de folha;
  - As entradas são classificadas e preenchidas de acordo com o fator de preenchimento especificado;
  - Outros níveis de índice são criados, simultâneamente.

Questões gerais sobre indexação

Índice para restrição de chave em atributo

- Enquanto se pesquisa o índice para inserir um novo registro, é fácil verificar ao mesmo tempo se outro registro no arquivo tem o mesmo valor de atributo de chave que o novo registro;
- Se o índice for criado em um campo não chave, ocorrem duplicatas.

Questões gerais sobre indexação

Armazenamento de relações baseado em coluna

- Alternativo ao modo tradicional de armazenar relações linha por linha;
- Permite a liberdade adicional na criaçao de índices;
- Vantagens para aplicações que exigem somente consultas de leitura;
- A mesma coluna pode estar presente em várias projeções de um tabela e os índices podem ser criados em cada projeção.

Questões gerais sobre indexação

Armazenamento de relações baseado em coluna

- Vantagens de desempenho para as seguintes áreas:
  - Particionamento vertical da tabela coluna por coluna - somente as colunas necessárias são acessadas;
  - Uso de índices por colunas e índices de junção em várias tabelas para responder as consultas sem acesso as tabelas de dados;
  - Uso de visões materializadas para suporte a consultas em múltiplas colunas.

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan