Câmpus
Anápolis de Ciências
Exatas e Tecnológicas
Henrique Santillo

Universidade
Estadual de Goiás

Curso: Sistemas de Informação
Disciplina : Banco de Dados II

Índices

Prof. M.e. Guiliano Rangel Alves

Capítulo 12: Indexação e hashing

- Conceitos básicos
- Índices ordenados
- Arquivos de índice de árvore B+
- Arquivos de índice de árvore binária
- Definição de índice em SQL
- Acesso por chave múltipla

Métrica de avaliação de índice

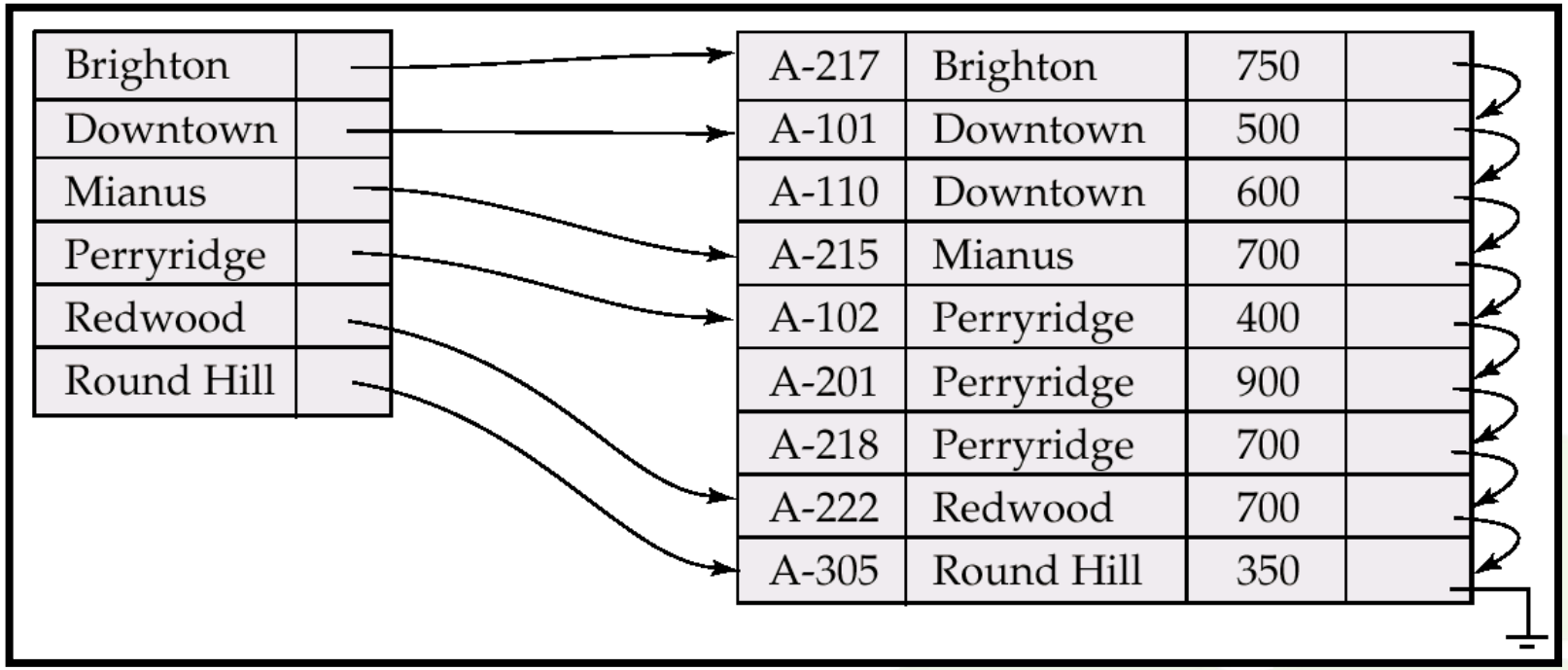
- Tipos de acesso admitidos com eficiência. Por exemplo,
  - registros com um valor especificado no atributo
  - ou registros com um valor de atributo caindo em um intervalo especificado de valores.
- Tempo de acesso
- Tempo de inserção
- Tempo de exclusão
- Sobrecarga de espaço

Índices ordenados

- Em um **índice ordenado**, as entradas de índice são **armazenadas classificadas pelo valor da chave de busca**. Por exemplo, catálogo de autor na biblioteca.
- **Índice primário**: em um arquivo **ordenado sequencialmente**, o índice cuja **chave de busca** especifica a **ordem sequencial do arquivo**.
  - Também chamado índice de agrupamento (ou clasterizado)
  - A chave de busca de um índice primário normalmente, mas não necessariamente, é a chave primária.
- **Índice secundário**: um índice cuja chave de busca especifica uma ordem diferente da ordem sequencial do arquivo. Também chamado índice não de agrupamento.
- **Arquivo sequencial indexado**: arquivo sequencial ordenado com um índice primário.

Arquivos de índice denso

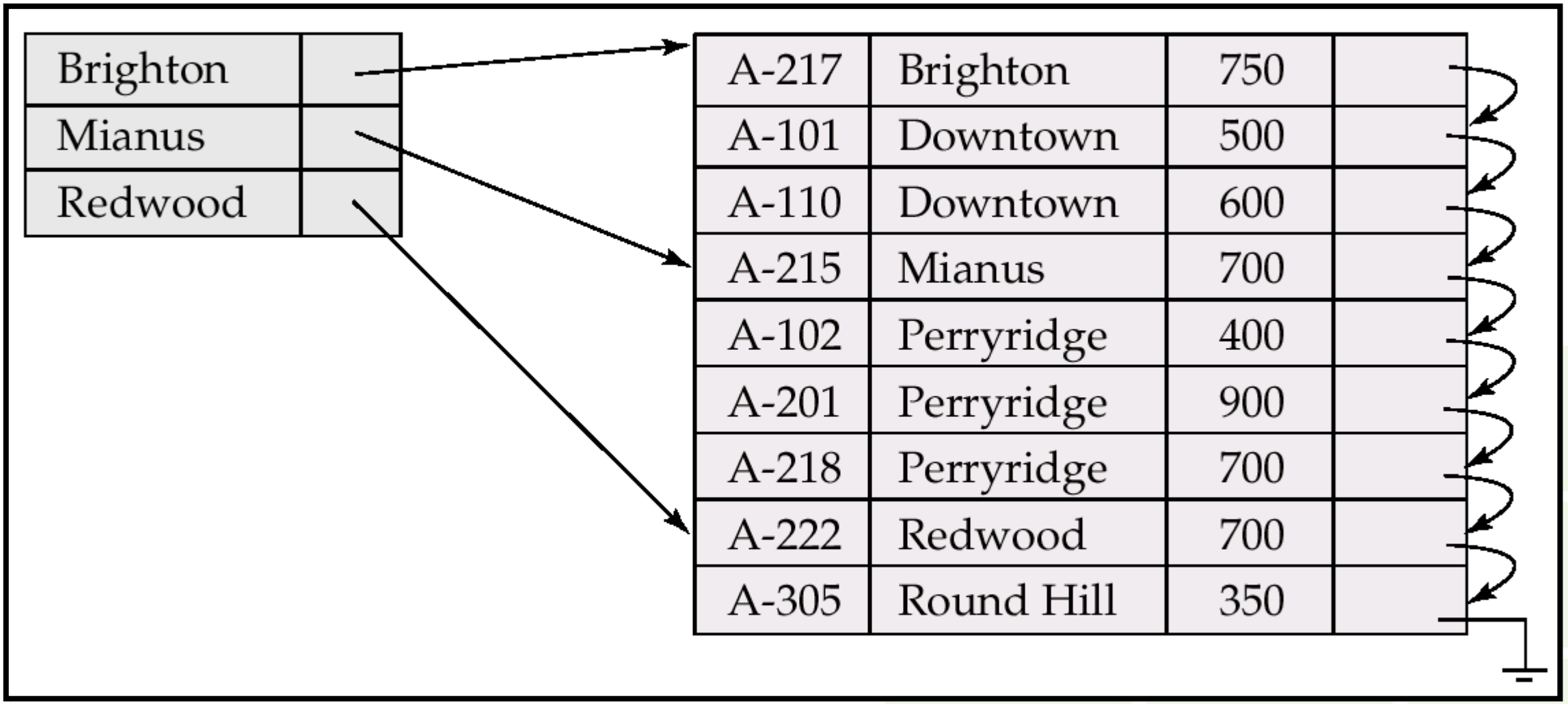
- **Índice denso —** O registro de índice aparece para cada valor de chave de busca no arquivo.



Arquivos de índice esparso

- **Índice esparso:** contém registros de índice para somente **alguns valores** de chave de busca.
  - Aplica-se quando os registros são **ordenados sequencialmente** por chave de busca
- Para localizar um registro com o valor de chave de busca K:
  - Encontramos o registro de índice com o maior valor de chave de busca < K
  - Pesquisamos o arquivo sequencialmente, começando no registro para o qual o registro de índice aponta
- Menos espaço e menos sobrecarga de manutenção para inserções e exclusões.
- Geralmente mais lento do que o índice denso para localizar registros.

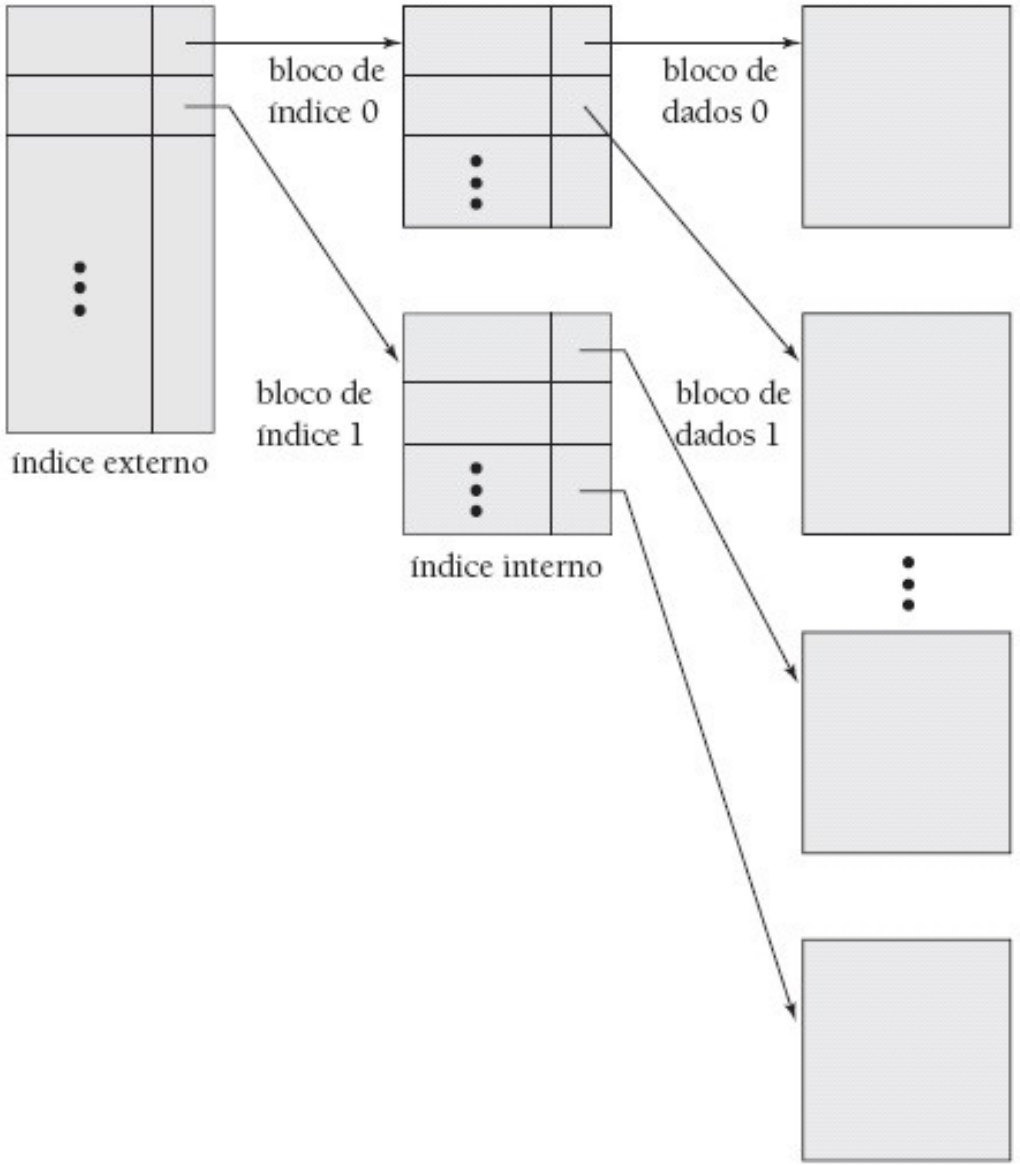
Exemplo de arquivos de índice esparso



Índice multinível

- Se o índice **primário** não **couber na memória**, o acesso se torna dispendioso.
- Para reduzir o número de acessos de disco aos registros de índice, trate o índice primário mantido em disco como um arquivo sequencial e construa um índice esparso sobre ele.
  - índice externo – um índice esparso do índice primário
  - índice interno – o arquivo de índice primário
- Se até mesmo o índice externo for muito grande para caber na memória principal, outro nível de índice pode ser criado, e assim por diante.
- Os índices em todos os níveis precisam ser atualizados na inserção ou exclusão no arquivo.

Índice multinível (cont.)



Atualização de índice: exclusão

- Se o registro excluído foi o único registro no arquivo com seu valor de chave de busca específico, a chave de busca também é excluída do índice.
- Exclusão de índice de único nível:
  - Índices densos - a exclusão da chave de busca é semelhante à exclusão de registro do arquivo.
  - Índices esparsos - se houver uma entrada para a chave de busca no índice, ela é excluída substituindo a entrada no índice pelo próximo valor de chave de busca no arquivo (em ordem de chave de busca). Se o próximo valor de chave de busca já tiver uma entrada de índice, a entrada é excluída em vez de ser substituída.

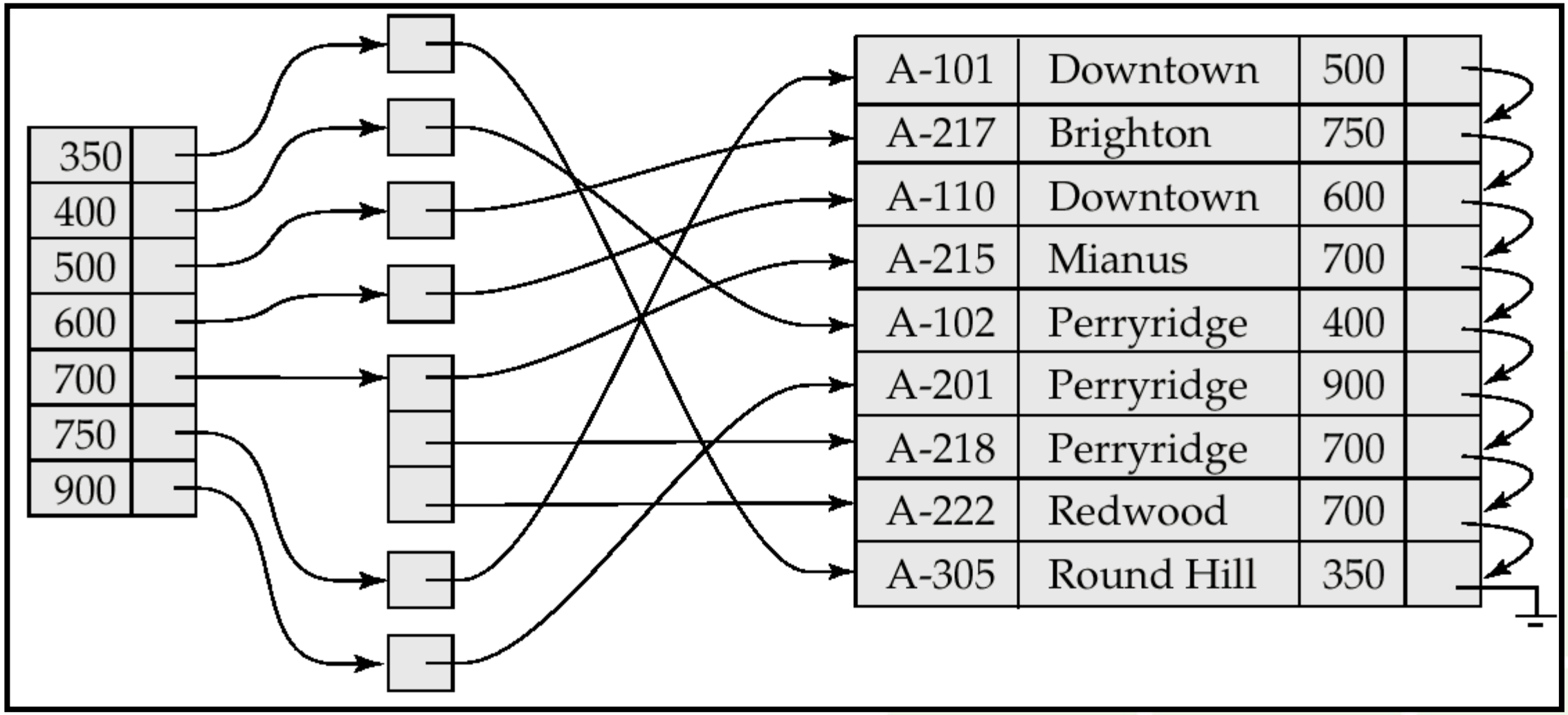
Atualização de índice: inserção

- Inserção de índice de único nível:
  - Realize uma pesquisa usando o valor de chave de busca que aparece no registro a ser inserido.
  - Índices densos - se o valor da chave de busca não aparecer no índice, insira-o.
  - Índices esparsos - se o índice armazena uma entrada para cada bloco do arquivo, nenhuma mudança precisa ser feita no índice, a menos que um novo bloco seja criado. Nesse caso, o primeiro valor de chave de busca que aparece no novo bloco é inserido no índice.
- Algoritmos de inserção multinível (além de exclusão) são simples extensões dos algoritmos de único nível

Índices secundários

- Frequentemente, alguém deseja encontrar todos os registros cujos valores em um certo campo (que não seja a chave de busca do índice primário) satisfazem alguma condição.
  - Exemplo 1: No banco de dados conta armazenado sequencialmente por número de conta, podemos querer encontrar todas as contas em determinada agência
  - Exemplo 2: como antes, mas onde queremos encontrar todas as contas com um saldo especificado ou intervalo de saldos
- Podemos ter um índice secundário com um registro de índice para cada valor de chave de busca; o registro de índice aponta para um balde que contém ponteiros para todos os registros reais com esse valor de chave de busca específico.

Índice secundário sobre o campo *saldo* de *conta*



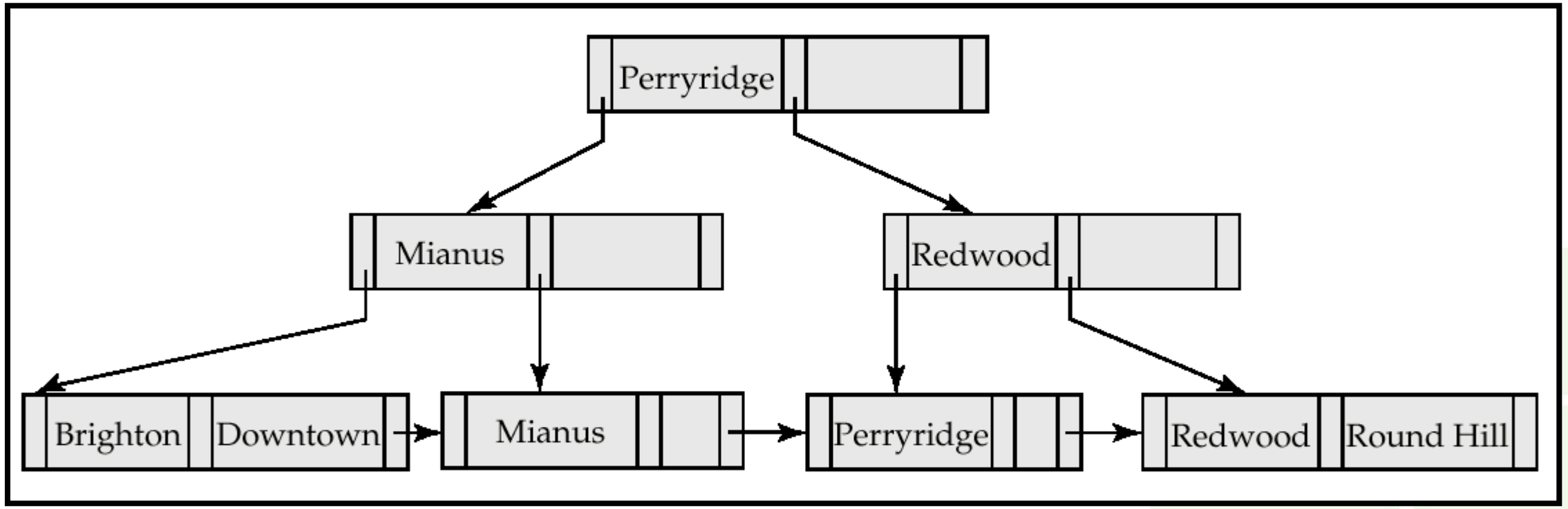
Índices primários e secundários

- Índices secundários precisam ser densos.
- Índices oferecem benefícios substanciais quando procuram registros.
- Quando um arquivo é modificado, cada índice no arquivo precisa ser atualizado. A atualização de índices impõe sobrecarga na modificação do banco de dados.
- A varredura sequencial usando índice primário é eficiente, mas uma varredura sequencial usando um índice secundário é dispendiosa.
  - cada acesso a registro pode apanhar um novo bloco do

Arquivos de índice de árvore B+

- Desvantagem dos arquivos sequenciais indexados: o desempenho diminui quando o arquivo aumenta, pois muitos blocos de estouro são criados. É preciso reorganizar periodicamente o arquivo inteiro.
- Vantagem dos arquivos de índice B+: reorganiza-se automaticamente com pequenas mudanças locais, em face a inserções e exclusões. A reorganização do arquivo inteiro não é necessária para manter o desempenho.
- Desvantagem das árvores B+: trabalho extra de inserção e exclusão, sobrecarga de espaço.
- Vantagens das árvores B+ superiores às desvantagens, e por isso software muito usadas.

Exemplo de uma árvore B+



Árvore B+ para arquivo *conta* (n = 3)

Definição de índice em SQL

- **Crie um índice**

  create index **<nome-índice>** on **<nome-relação> (<lista-atributos>)**

Por exemplo:

create index ***indice-b*** on ***agencia(nome-agencia)***

- Use *create unique index* para especificar indiretamente e impor a condição em que a chave de busca é uma chave candidata.
  - Não é realmente necessário se a restrição de integridade *unique* da SQL for admitida
- **Para remover um índice**

  drop index **<nome-índice>**

Acesso de chave múltipla

- **Use índices múltiplos para certos tipos de consultas.**
  - **Exemplo:**

select ***número-conta***
from ***conta***
where ***nome-agência*** **= "Perryridge"** and ***saldo*** **= 1000**

- Estratégias possíveis para processar consulta usando índices sobre atributos isolados**:**

1. Use índice sobre *nome-agência* para encontrar contas com saldos de $1000; teste *nome-agência* = "Perryridge".
2. Use índice sobre *saldo* para encontrar contas com saldos de $1000; teste *nome-agência* = "Perryridge".
3. Use índice *nome-agência* para encontrar ponteiros para todos os registros pertencentes à agência Perryridge. De modo semelhante, use índice sobre *saldo*. Apanhe a interseção dos dois conjuntos de ponteiros obtidos.

Índices sobre atributos múltiplos

- Suponha que temos um índice sobre a chave combinada (nome-agência, saldo).
- Com a cláusula where
  `where nome-agência = “Perryridge” and saldo = 1000`
  o índice sobre a chave de busca combinada apanhará apenas registros que satisfazem as duas condições.
  O uso de índices separados é menos eficiente - podemos apanhar muitos registros (ou ponteiros) que satisfazem apenas uma das condições.
  Também pode tratar de modo eficiente
  `where nome-agência = “Perryridge” and saldo < 1000`
- Mas não pode tratar de modo eficiente
  `where nome-agência < “Perryridge” and saldo = 1000`
  Pode apanhar muitos registros que satisfazem a primeira, mas não a segunda condição.

FIM