# Índices*

Professora Rosane Minghim

* Baseado no material de Leandro C. Cintra e
M. C. F. de Oliveira. Fonte: Folk & Zoelick, File Structures.

# Índice

- Em geral, um índice fornece mecanismos para localizar informações.

- No caso de arquivos, permite localizar registros rapidamente, com a vantagem de que não é necessário reorganizar o arquivo de dados quando novas informações são inseridas no mesmo.

# Índice simples

- Exemplo: Uma enorme coleção de CDs.
- Registros de tamanho variável com os campos:
  - ID Number: Número de identificação
  - Title: Título
  - Composer: Compositor(es)
  - Artist: Artista(s)
  - Label: Rótulo (código da gravadora)

- Chave primária: combinação de Label e ID Number.

# Arquivo de Dados

| Record address | Label | ID number | Title | Composer(s) | Artist(s) |
|---|---|---|---|---|---|
| 17 | LON | 2312 | Romeo and Juliet | Prokofiev | Maazel |
| 62 | RCA | 2626 | Quartet in C Sharp Minor | Beethoven | Julliard |
| 117 | WAR | 23699 | Touchstone | Corea | Corea |
| 152 | ANG | 3795 | Symphony No. 9 | Beethoven | Giulini |
| 196 | COL | 38358 | Nebraska | Springsteen | Springsteen |
| 241 | DG | 18807 | Symphony No. 9 | Beethoven | Karajan |
| 285 | MER | 75016 | Coq d'Or Suite | Rimsky-Korsakov | Leinsdorf |
| 338 | COL | 31809 | Symphony No. 9 | Dvorak | Bernstein |
| 382 | DG | 139201 | Violin Concerto | Beethoven | Ferras |
| 427 | FF | 245 | Good News | Sweet Honey in the Rock | Sweet Honey in the Rock |

**Figure 7.2** Contents of sample recording file.

# Índice Simples

**Index**

| *Key* | *Reference field* |
|---|---|
| ANG3795 | 152 |
| COL31809 | 338 |
| COL38358 | 196 |
| DG139201 | 382 |
| DG18807 | 241 |
| FF245 | 427 |
| LON2312 | 17 |
| MER75016 | 285 |
| RCA2626 | 62 |
| WAR23699 | 117 |

**Recording file**

| *Address of record* | *Actual data record* |
|---|---|
| 17 | LON \| 2312 \| Romeo and Juliet \| Prokofiev \| ... |
| 62 | RCA \| 2626\| Quartet in C Sharp Minor \| Beethoven \| ... |
| 117 | WAR \| 23699 \| Touchstone \| Corea \| ... |
| 152 | ANG \| 3795 \| Symphony No. 9 \| Beethoven \| ... |
| 196 | COL \| 38358 \| Nebraska \| Springsteen \| ... |
| 241 | DG \| 18807 \| Symphony No. 9 \| Beethoven \| ... |
| 285 | MER \| 75016 \| Coq d'Or Suite \| Rimsky-Korsakov \| ... |
| 338 | COL \| 31809 \| Symphony No. 9 \| Dvorak \| ... |
| 382 | DG \| 139201 \| Violin Concerto \| Beethoven \| ... |
| 427 | FF \| 245 \| Good News \|Sweet Honey in the Rock \| ... |

**Figure 7.3** Index of the sample recording file.

# Índice Simples

- O índice é ele próprio um arquivo com registros de tamanho fixo.
- Cada registro tem 2 campos de tamanho fixo:
  - um campo contém a chave;
  - outro informa a posição inicial (byte offset) do registro no arquivo de dados.
- A cada registro do arquivo de dados corresponde um registro no arquivo índice.
- O índice está ordenado, apesar do arquivo de dados não estar
  - Em geral, o arquivo de dados está organizado segundo a ordem de entrada dos registros - *entry sequenced file*.

# Índice Simples

- Usa-se dois arquivos:
  - o arquivo índice (*index file*);
  - e o arquivo de dados (*data file*).
- O arquivo índice é:
  - mais fácil de trabalhar, pois usa registros de tamanho fixo;
  - pode ser pesquisado com P.B. (em memória principal);
  - é muito menor do que o arquivo de dados.
- Registros de tamanho fixo no arquivo índice impõem um limite ao tamanho da chave primária.
- Os registros do índice poderiam conter outros campos além da chave/*offset* (por exemplo, o tamanho do registro).

# Índice Simples

- A inclusão de registros será muito mais rápida se o índice pode ser mantido (manipulado) em memória e o arquivo de dados é *entry sequenced*.
- Dados a chave e o *offset*, um único *seek* (i.e., um único acesso a disco) é necessário no arquivo de dados para recuperar o registro correspondente.

# Operações básicas no índice

- Para índices que cabem em memória:
  - criar arquivos índice e de dados;
  - carregar índice para memória;
  - inserir registro
    - inserção deve ser feita no arquivo de dados.
    - e também no índice, que eventualmente será reorganizado.
  - eliminar registro
    - remove do arquivo de dados, usando algum dos mecanismos de remoção.
    - remove também do índice. A remoção do registro do índice pode exigir a sua reorganização, ou pode-se simplesmente marcar os registros como removidos.

# Operações básicas no índice

- Para índices que cabem em memória:
  - atualizar registro - duas categorias:
    - Muda o valor da chave.
    - Muda o conteúdo do registro.
  - atualizar índice no disco: caso sua cópia em memória tenha sido alterada
    - É imperativo que o programa se proteja contra índices desatualizados.

# Como evitar índices desatualizados

- Deve haver um mecanismo que permita saber se o índice está atualizado em relação ao arquivo de dados.
- Possibilidade: um status *flag* é "setado" no arquivo índice mantido em disco assim que a sua cópia na memória é alterada.
- Esse *flag* pode ser mantido no registro *header* do arquivo índice, e atualizado sempre que o índice é reescrito no disco.
- Se um programa detecta que o índice está desatualizado, uma função é ativada que reconstrói o índice a partir do arquivo de dados.

# Índices muito grandes

- Se o índice não cabe inteiro na memória, o acesso e manutenção precisam ser feitos em memória secundária.
- Não é mais aconselhável usar índices simples, uma vez que:
  - a busca binária pode exigir vários acessos a disco;
  - a necessidade de deslocar registros nas inserções e remoções de registros tornaria a manutenção do índice excessivamente cara.

# Índices muito grandes

- Utiliza-se outras organizações
  - *Hashing*, caso a velocidade de acesso seja a prioridade máxima
    - Acesso direto apenas.
  - Árvores-B, caso se deseje combinar acesso por chaves e acesso sequencial eficientemente.

# Acesso por múltiplas chaves

- Como saber qual é a chave primária do registro que se quer acessar?
- Normalmente, o acesso a registros não se faz por chave primária, e sim por chaves secundárias.
- Solução: cria-se um índice que relaciona uma chave secundária à chave primária (e não diretamente ao registro).

# Acesso por múltiplas chaves

- Índices permitem muito mais do que simplesmente melhorar o tempo de busca por um registro.
- Múltiplos índices secundários
  - permitem manter diferente visões dos registros em um arquivo de dados.
  - permitem combinar chaves associadas e, deste modo, fazer buscas que combinam visões particulares.

# Acesso por múltiplas chaves

Composer index

| *Secondary key* | *Primary key* |
|---|---|
| BEETHOVEN | ANG3795 |
| BEETHOVEN | DG139201 |
| BEETHOVEN | DG18807 |
| BEETHOVEN | RCA2626 |
| COREA | WAR23699 |
| DVORAK | COL31809 |
| PROKOFIEV | LON2312 |
| RIMSKY-KORSAKOV | MER75016 |
| SPRINGSTEEN | COL38358 |
| SWEET HONEY IN THE R | FF245 |

**FIGURE 6.6** *Secondary key index organized by composer*.

Title index

| *Secondary key* | *Primary key* |
|---|---|
| COQ D'OR SUITE | MER75016 |
| GOOD NEWS | FF245 |
| NEBRASKA | COL38358 |
| QUARTET IN C SHARP M | RCA2626 |
| ROMEO AND JULIET | LON2312 |
| SYMPHONY NO. 9 | ANG3795 |
| SYMPHONY NO. 9 | COL31809 |
| SYMPHONY NO. 9 | DG18807 |
| TOUCHSTONE | WAR23699 |
| VIOLIN CONCERTO | DG139201 |

**FIGURE 6.8** *Secondary key index organized by recording title*.

# Alterações nas operações básicas

- **Inserir registro**:
  - Quando um novo registro é inserido no arquivo, devem ser inseridas as entradas correspondentes no índice primário e nos índices secundários.
  - Campo **chave** deve armazenado em sua forma canônica no índice secundário. O valor pode ser truncado, porque o tamanho da chave deve ser mantido fixo.
  - Diferença importante entre os índices primário e os secundários: nesses últimos pode ocorrer duplicação de chaves. Chaves duplicadas devem ser mantidas agrupadas e ordenadas segundo a chave primária.

# Alterações nas operações básicas

- **Eliminar registro**:
  - Implica em remover o registro do arquivo de dados e de todos os índices.
  - Índices primário e secundários são mantidos ordenados segundo a chave. Consequentemente, a remoção requer o rearranjo dos registros remanescentes para não deixar "espaços vagos".
  - **Alternativa**: atualizar apenas o índice primário, sem eliminar a entrada correspondente ao registro no índice secundário.

# Alterações nas operações básicas

- **Vantagem**: economia de tempo substancial quando vários índices secundários estão associados ao arquivo, principalmente se esses índices são mantidos em disco.
- **Custo**: espaço ocupado por registros inválidos. Poder-se-ia fazer "coletas de lixo" periódicas nos índices secundários.
- Ainda será um problema se o arquivo é muito volátil
  - outra solução: índice em árvore-B.

# Alterações nas operações básicas

- **Atualizar registro** - 3 situações:
  - alterou uma chave secundária: o índice secundário para esta chave precisa ser reordenado.
  - alterou a chave primária: atualizar (reordenar) o índice primário e corrigir os os campos de referência índices secundários.
    - Vantagem: atualização dos índices secundários não requer reorganização!
  - alterou outros campos: não afeta nenhum dos índices.

# Busca usando múltiplas chaves

- Uma das aplicações mais importantes das chaves secundárias é localizar conjuntos de registros do arquivo de dados usando uma ou mais chaves
- Pode-se fazer uma busca em vários índices e combinar (AND,OR,NOT) os resultados
- Ex: encontre todos os registros de dados tal que
  - composer = "BEETHOVEN" AND title = “SYMPHONY NO. 9”

# Melhoria de índices secundários

- Dois problemas nas estruturas de índices vistas até agora:
  - repetição das chaves secundárias;
  - necessidade de reordenar os índices sempre que um novo registro é inserido no arquivo, mesmo que esse registro tenha um valor de chave secundária já existente no arquivo.

# Melhoria de índices secundários

- **Solução 1**: Associar um vetor de tamanho fixo a cada chave secundária
  - Não é necessário reordenar o índice a cada inserção de registro;
  - Limitado a um número fixo de repetições;
  - Ocorre fragmentação interna enorme no índice - que talvez não compense a eliminação da duplicação de chaves.

# Solução 1

Revised composer index

| *Secondary key* | *Set of primary key references* | | | |
|---|---|---|---|---|
| BEETHOVEN | ANG3795 | DG139201 | DG18807 | RCA2626 |
| COREA | WAR23699 | | | |
| DVORAK | COL31809 | | | |
| PROKOFIEV | LON2312 | | | |
| RIMSKY-KORSAKOV | MER75016 | | | |
| SPRINGSTEEN | COL38358 | | | |
| SWEET HONEY IN THE R | FF245 | | | |

**FIGURE 6.9** Secondary key index containing space for multiple references for each secondary key.

# Melhoria de índices secundários

- **Solução 2**: Manter uma lista de referências - Listas invertidas
  - Já que tem-se uma lista de chaves primárias, pode-se associar cada chave secundária a uma "lista encadeada" (lista invertida) das chaves primárias referenciadas.
  - Índice secundário passa a ser composto por registros com 2 campos: campo chave, e campo com o RRN do primeiro registro com essa chave na lista invertida.
  - Referências às chaves primárias associadas a cada chave secundária são mantidas em um arquivo sequencial separado, organizado segundo a entrada dos registros.

# Listas invertidas: visão conceitual

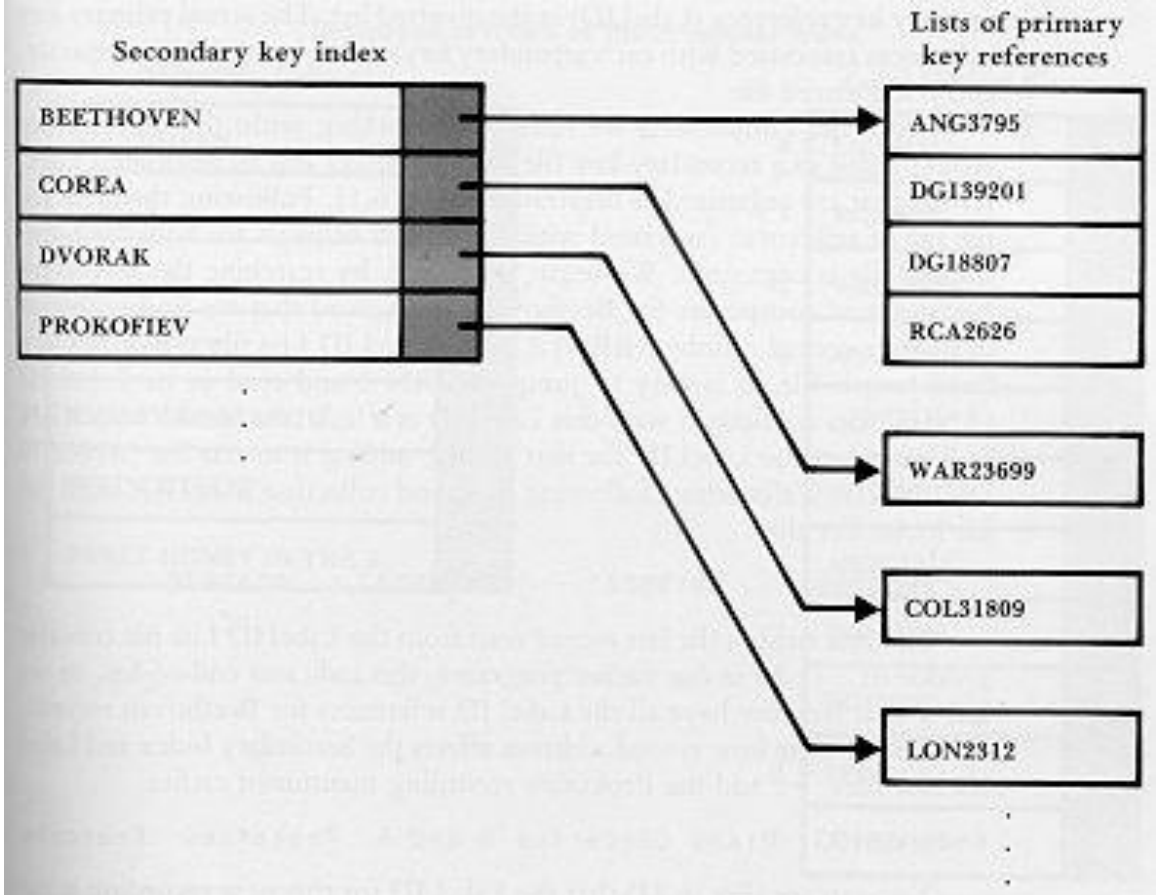
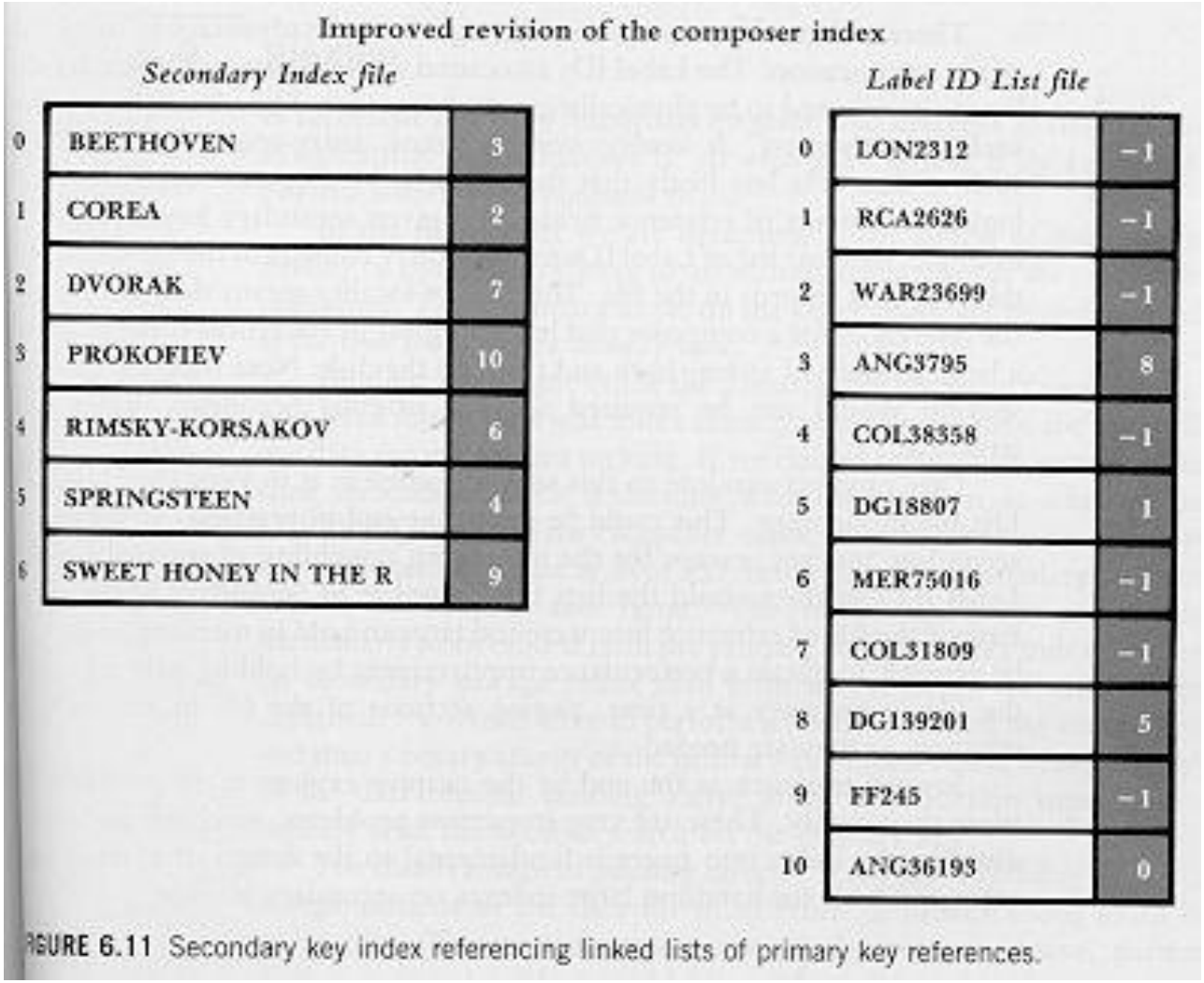



FIGURE 6.10 Conceptual view of the primary key reference fields as a series of lists.

# Listas invertidas



Improved revision of the composer index

*Secondary Index file*

| | | |
|---|---|---|
| 0 | BEETHOVEN | 3 |
| 1 | COREA | 2 |
| 2 | DVORAK | 7 |
| 3 | PROKOFIEV | 10 |
| 4 | RIMSKY-KORSAKOV | 6 |
| 5 | SPRINGSTEEN | 4 |
| 6 | SWEET HONEY IN THE R | 9 |

*Label ID List file*

| | | |
|---|---|---|
| 0 | LON2312 | −1 |
| 1 | RCA2626 | −1 |
| 2 | WAR23699 | −1 |
| 3 | ANG3795 | 8 |
| 4 | COL38358 | −1 |
| 5 | DG18807 | 1 |
| 6 | MER75016 | −1 |
| 7 | COL31809 | −1 |
| 8 | DG139201 | 5 |
| 9 | FF245 | −1 |
| 10 | ANG36193 | 0 |

GURE 6.11 Secondary key index referencing linked lists of primary key references.

# Listas invertidas

- **Vantagens**:
  - índice secundário só é alterado quando é inserido um registro com chave inexistente, ou quando é alterada uma chave já existente;
  - operações de eliminação, inserção ou alteração de registros já existentes implicam apenas em alterar arquivo da lista invertida
  - ordenação do arquivo de índice secundário é mais rápida: menos registros - e registros menores.
  - arquivo com listas de chaves nunca precisa ser ordenado, pois é "*entry-sequenced*";
  - é fácil reutilizar o espaço liberado pelos registros eliminados do arquivo de listas.

# Listas invertidas

- **Problema**
  - registros associados não estão adjacentes no disco: podem ser necessários vários *seek*s para recuperar a lista.
  - O ideal seria manter o índice e a lista na memória.

# Binding

- Nos índices primários vistos a associação (*binding*) entre a chave primária e o endereço físico do registro a que ela se refere ocorre no momento em que o registro é criado.
- Índice simples fornece acesso direto e, portanto, mais rápido, a um registro, dada a sua chave.
- Já as chaves secundárias são associadas a um endereço apenas no momento em que são de fato usadas(*late binding*). Isso implica em um acesso mais lento.
- O late *binding* trouxe vantagens: manutenção mais flexível, mais eficiente e confiável.
- Ressalta-se: é sempre desejável manter as modificações localizadas, o que é possível com o *late-binding*. O *early binding* só é aconselhável se o arquivo de dados é (quase) estático, e o acesso rápido a registros é a maior prioridade.

# FIM