



Índices

SCE-203 – Algoritmos e Estruturas de Dados II



Índice

- Em geral, um índice fornece **mecanismos para localizar informações**
 - Índice de um livro ou catálogo de uma biblioteca
 - Facilitam muito o trabalho de busca!
- Em arquivos
 - Permite localizar registros rapidamente
 - **Não é necessário ordenar** arquivo de dados, nem quando novos registros são adicionados



Índice

- **Índices Simples**

- Representados em *arrays*, cuja estrutura contém as *chaves* e os *campos de referência*

- **Outros Esquemas de Indexação**

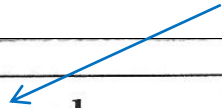
- Usam estruturas de dados mais complexas (árvores)



Índice simples

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

#bytes do registro



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.

Chave primária: combinação de **Label + ID Number**

Poderia ser qualquer outro campo ou combinação de campos que fosse único para cada registro

Index		Recording file	
Key	Reference field	Address of record	Actual data record
ANG3795	152	17	LON 2312 Romeo and Juliet Prokofiev ...
COL31809	338	62	RCA 2626 Quartet in C Sharp Minor Beethoven ...
COL38358	196	117	WAR 23699 Touchstone Corea ...
DG139201	382	152	ANG 3795 Symphony No. 9 Beethoven ...
DG18807	241	196	COL 38358 Nebraska Springsteen ...
FF245	427	241	DG 18807 Symphony No. 9 Beethoven ...
LON2312	17	285	MER 75016 Coq d'Or Suite Rimsky-Korsakov ...
MER75016	285	338	COL 31809 Symphony No. 9 Dvorak ...
RCA2626	62	382	DG 139201 Violin Concerto Beethoven ...
WAR23699	117	427	FF 245 Good News Sweet Honey in the Rock ...

Figure 7.3 Index of the sample recording file.



Índice simples

- O índice consiste, em geral, em um **outro arquivo** com registros de tamanho fixo
 - Mesmo que o arquivo principal com os dados não tenha registros de tamanho fixo
- Cada registro do índice contém pelo menos **2 campos de tamanho fixo**
 - Chave
 - Posição inicial (*byte offset*) ou RRN do registro no arquivo de dados



Índice simples

- A cada registro do arquivo de dados corresponde um registro no 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

- **Vantagens** do arquivo de índice sobre o de dados
 - Mais fácil de trabalhar, pois usa registros de tamanho fixo
 - Pode ser pesquisado com busca binária (em memória principal, inclusive, se valer a pena carregá-lo)
 - 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

- Como são feitas as operações básicas num arquivo **indexado** e “**entry-sequenced**”?
 - Inserção de registros
 - Remoção de registros
 - Atualização de registros
 - Busca de registros



Índice simples

- Como são feitas as operações básicas num arquivo **indexado** e “**entry-sequenced**”?
 - Inserção de registros
 - No arquivo de dados: no final;
 - Inserção de novo registro de chave no índice
 - Remoção de registros
 - Busca do registro usando o índice → 1 seek + remoção
 - Remoção de registro de chave no índice
 - Atualização de registros
 - Busca no índice + seek
 - Se mudar chave, alterar tb o índice
 - Busca de registros
 - Busca no índice + seek



Índice simples

- A inclusão de registros será muito mais rápida se o **índice pode ser mantido em memória interna** e o **arquivo de dados é *entry sequenced***
- Dados a chave e o *offset*, um **único *seek* 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 (aula anterior)
 - Remove também do índice
 - Índice pode ser reorganizado ou pode-se apenas marcar os registros excluídos



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
 - Possíveis estratégias?



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 deve ser ativada para reconstruir o índice a partir do arquivo de dados



Exercício

- Implementar em C uma sub-rotina que construa um índice em arquivo a partir de um arquivo de dados de alunos *entry sequenced*, com registros de tamanho variável

```
struct aluno {  
    char *nome;  
    int nro_USP;  
}
```



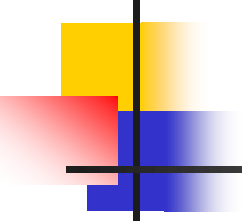

Índices muito grandes

- 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

- Utilizam-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

- 
-
- Ainda assim, o uso de índice simples que não cabe em RAM ainda é mais vantajoso do que manter um arquivo de dados ordenado e não indexado.
 - Tem registros de tamanho fixo, portanto, é possível fazer busca binária;
 - É bem menor do que arquivo de dados, facilitando a manutenção;
 - Índices fornecem múltiplas visões de um conjunto de dados



Acesso por múltiplas chaves

- Como decidimos qual será a chave primária dos registros?
 - Identifica unicamente, mas só tem valor “organizacional”
- Normalmente, o acesso a registros não se faz por chave primária, e sim por chaves secundárias
 - Quando se procura a busca por um livro em um biblioteca, começa-se pelo seu número ou pelo título/autor?
- Solução: cria-se um índice que relaciona uma chave secundária à chave primária (e não diretamente ao registro)
 - Índice secundário



Acesso por múltiplas chaves

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

Exemplo do Arquivo de Compositores

Index Primário		Recording file	
<i>Key</i>	<i>Reference field</i>	<i>Address of record</i>	Arquivo de Dados
			<i>Actual data record</i>
ANG3795	152	17	LON 2312 Romeo and Juliet Prokofiev ...
COL31809	338	62	RCA 2626 Quartet in C Sharp Minor Beethoven ...
COL38358	196	117	WAR 23699 Touchstone Corea ...
DG139201	382	152	ANG 3795 Symphony No. 9 Beethoven ...
DG18807	241	196	COL 38358 Nebraska Springsteen ...
FF245	427	241	DG 18807 Symphony No. 9 Beethoven ...
LON2312	17	285	MER 75016 Coq d'Or Suite Rimsky-Korsakov ...
MER75016	285	338	COL 31809 Symphony No. 9 Dvorak ...
RCA2626	62	382	DG 139201 Violin Concerto Beethoven ...
WAR23699	117	427	FF 245 Good News Sweet Honey in the Rock ...

Figure 7.3 Index of the sample recording file.

Acesso por múltiplas chaves

Ex.: busca por compositor ou por música

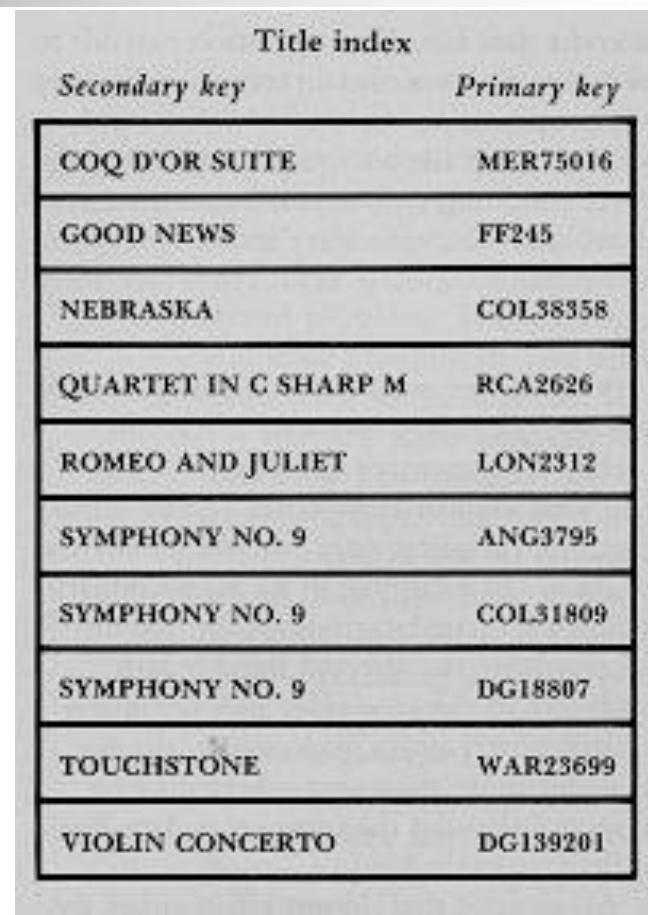
Índices Secundários



A table titled 'Composer index' with two columns: 'Secondary key' and 'Primary key'. It lists composers and their corresponding primary keys.

<i>Secondary key</i>	<i>Primary key</i>
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



A table titled 'Title index' with two columns: 'Secondary key' and 'Primary key'. It lists recording titles and their corresponding primary keys.

<i>Secondary key</i>	<i>Primary key</i>
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**
 - Tanto melhor se índices (primário e secundários) couberem na RAM
- Campos do índice secundário são de tamanho fixo, embora no arquivo de dados esses valores possam estar em campos de tamanho variável (ex. nome do compositor ou da música)
 - O valor então pode ser truncado
 - Levar isso em conta quando estabelecer o tamanho do campo
- Diferença importante entre os índices primário e os **secundários**: nesses últimos pode ocorrer **duplicação de chaves**
 - Chaves duplicadas devem ser mantidas agrupadas e ordenadas



Alterações nas operações básicas

- **Eliminar registro**

- Implica em **remover o registro do arquivo de dados e de todos os índices (primário e secundários)**
- Se índices mantidos ordenados, **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
 - Pode-se fazer "coletas de lixo" periódicas nos índices secundários
 - Ainda será um problema se o arquivo for muito volátil
 - Outra solução: índice em árvore-B
 - previsão no algoritmo de busca: embora presente no índice secundário, não haverá registro correspondente no primário



Alterações nas operações básicas

- **Atualizar registro - 3 situações**
 - **Alterou uma chave secundária:** o índice secundário para esta chave, se houver, **precisa ser reordenado**
 - **Alterou a chave primária:** **reordenar o índice primário** e corrigir os campos de referência dos índices secundários
 - Atualização dos índices secundários não requer reorganização
 - Alterou outros campos: não afeta nenhum dos índices
 - **E se o tamanho do registro mudar?**



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**
- Exemplo: encontre todos os registros de dados tal que
 - composer = "BEETHOVEN" **AND** title = "SYMPHONY NO. 9"

Busca usando múltiplas chaves

composer = "BEETHOVEN" **AND** title = "SYMPHONY NO. 9"

1

Composer index	
<i>Secondary key</i>	<i>Primary key</i>
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

2

Title index	
<i>Secondary key</i>	<i>Primary key</i>
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

3



Melhoria de índices secundários

- **Dois problemas** na estrutura de índices secundários 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 uma **lista de tamanho fixo a cada chave secundária**
 - Não seria mais necessário reordenar o índice a cada inserção de registro
 - Porém:
 - Limitado a um número fixo de repetições
 - Ocorre enorme fragmentação interna no índice - que talvez não compense a eliminação da duplicação de chaves

Solução 1

<i>Secondary key</i>	<i>Revised composer index</i>			
	<i>Set of primary key references</i>			
BEETHOVEN	ANG3795	DG139201	DG18807	RCA2626
COREA	WAR23699			
DVORAK	COL31809			
PROKOFIEV	LON2312			
RIMSKY-KORSAKOV	MER75016			
SPRINGSTEEN	COL38358			
SWEET HONEY IN THE R	FF245			

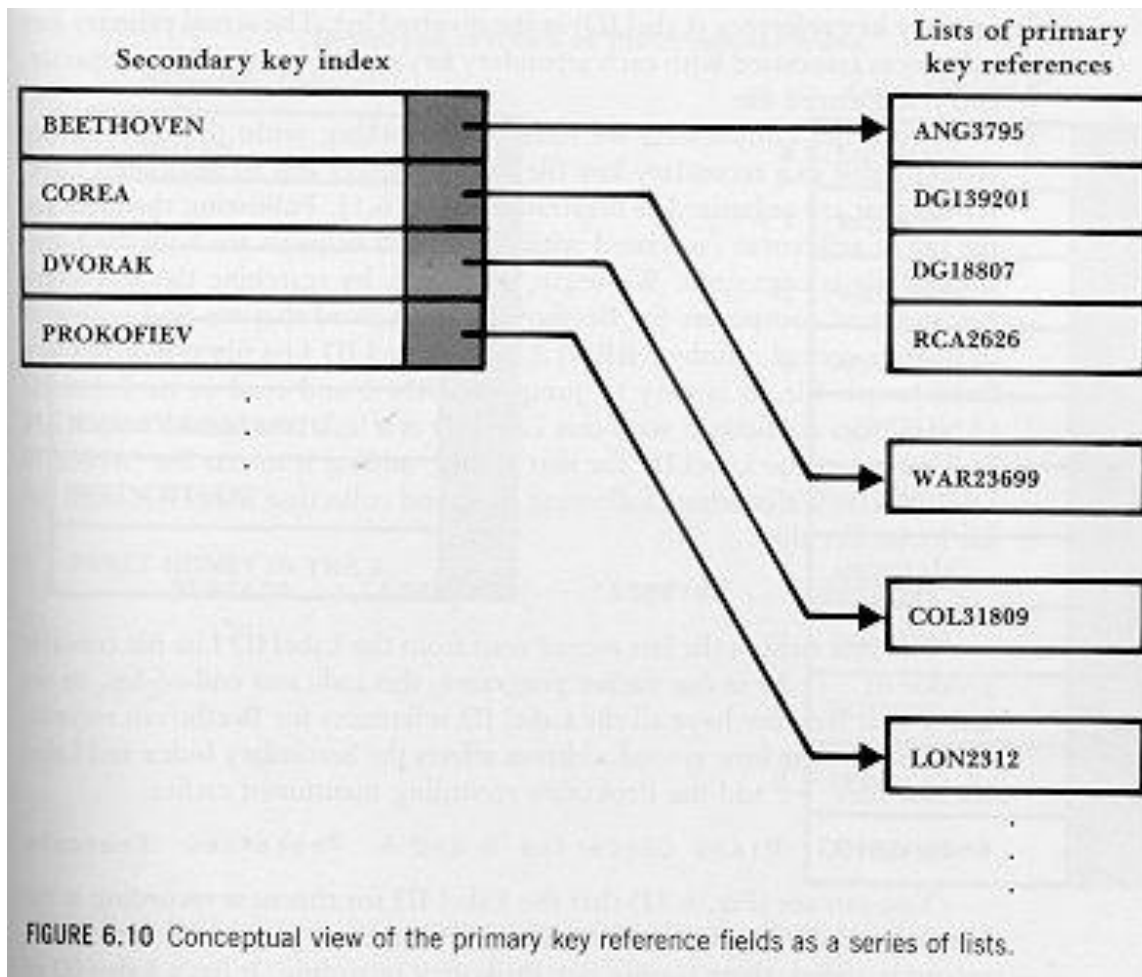
FIGURE 6.9 Secondary key index containing space for multiple references for each secondary key.



Melhoria de índices secundários

- **Solução 2:** manter uma lista de referências - **listas invertidas**
 - Pode-se associar cada **chave secundária a uma lista encadeada das chaves primárias** referenciadas
 - Índice secundário passa a ser composto por registros com 2 campos: **campo chave** e **campo com o RRN/byte offset do primeiro registro** com essa chave na lista invertida

Listas invertidas: visão conceitual



Por que se chama lista invertida?

Listas invertidas encadeadas

Referências às chaves primárias associadas a cada chave secundária podem ser mantidas em um **arquivo sequencial separado**, organizado segundo a entrada dos registros

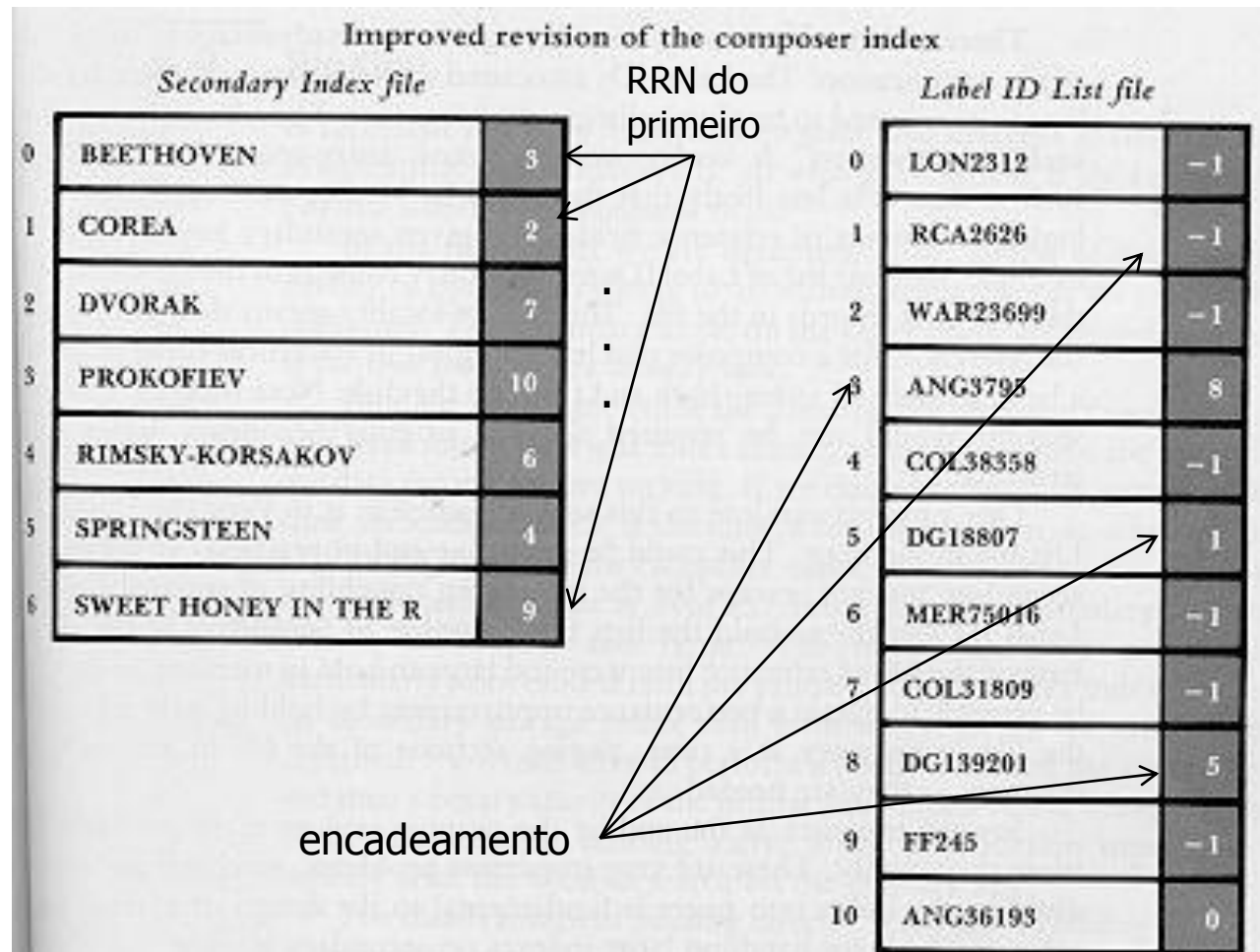


FIGURE 6.11 Secondary key index referencing linked lists of primary key references.



Listas invertidas

- Vantagens desta estratégia?



Listas invertidas

- Vantagens desta estratégia?
 - Índice secundário só é alterado quando é inserido um registro com chave secundária inexistente
 - Operações de eliminação, inserção ou alteração de registros já existentes implicam apenas em alterar arquivo das listas encadeadas
 - Ordenação do arquivo de índice secundário é mais rápida: menos registros - e registros menores
 - Arquivo com listas encadeadas nunca precisa ser ordenado, pois é *entry sequenced*
 - É fácil reutilizar o espaço liberado pelos registros eliminados do arquivo de listas encadeadas, já que os registros são de tamanho fixo.



Listas invertidas

- Problemas desta estratégia?



Listas invertidas

- Problemas desta estratégia?
 - Registros associados a um chave secundária não estão necessariamente adjacentes no disco: podem ser necessários vários *seeks* para recuperar os registros de uma lista encadeada
 - Buscar “todas as composições de Beethoven”
 - O ideal seria manter o índice e a lista encadeada na memória



Índices seletivos

- Um índice não precisa cobrir todo o arquivo de dados
 - Índice de músicas clássicas
 - Índice de músicas lançadas depois de 1980
- Dependente da aplicação e uso dos dados



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



Binding

- 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* traz 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