

Organização arquivos: indexação

Prof. Tiago A. Almeida

`talmeida@ufscar.br`

- ✓ Índices simples
- ✓ Acesso por múltiplas chaves
- ✓ Listas invertidas



✓ 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



✓ Exemplo: uma enorme coleção de Cds

✓ **Registros**

- **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

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

Arquivo de dados - exemplo

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

Chave Primária**RRN**

Index	Reference field	Address of record	Recording file
	<i>Key</i>		<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 ...

- ✓ 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
- ✓ Cada registro do arquivo de dados possui um registro correspondente no arquivo de 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**

✓ Usa-se dois arquivos:

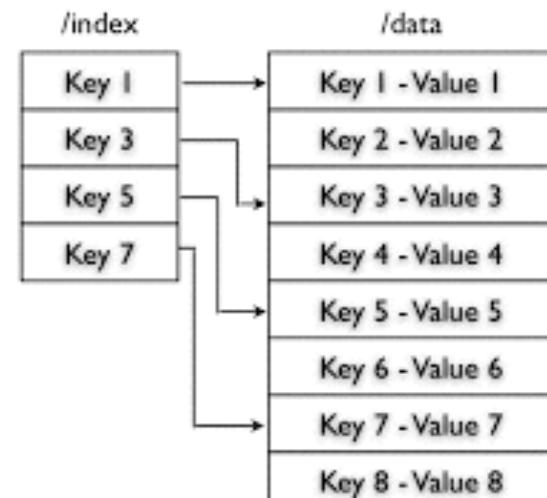
- o arquivo de índice (*index file*)
- o arquivo de dados (*data file*)

✓ O arquivo de índice é:

- mais fácil de trabalhar, pois usa registros de **tamanho fixo**
- pode ser pesquisado com **pesquisa binária** (em memória principal)
- é muito menor do que o arquivo de dados (idealmente **cabe na RAM**)

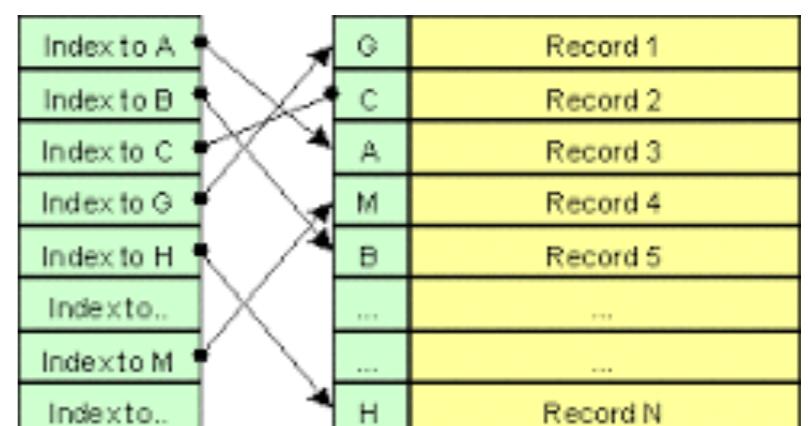
✓ Registros de tamanho fixo no arquivo de índice impõem um **limite** ao tamanho da chave primária

✓ Os registros do índice poderiam conter outros campos além da chave/*offset* (por exemplo, o tamanho do registro)



Índices simples

- ✓ A inclusão de registros é **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 *byte offset*, **um único seek** é necessário no arquivo de dados para recuperar o registro correspondente



Indexed File Organisation

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 deverá ser reordenado.
- **eliminar registro**
 - ★ remove do arquivo de dados, usando algum mecanismo 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**

- ▶ Índice deve ser atualizado e reordenado

- ★ 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:** uma *status flag* é “setada” no arquivo de índice mantido em disco assim que a sua cópia na memória é alterada 
- ✓ Essa *flag* pode ser mantida no registro *header* do arquivo de índice, e atualizada 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**



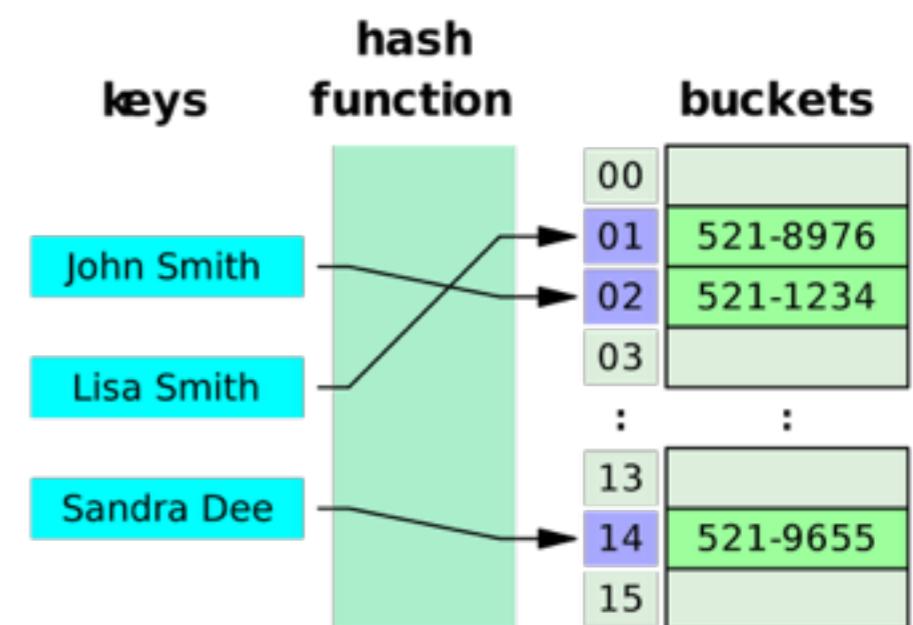
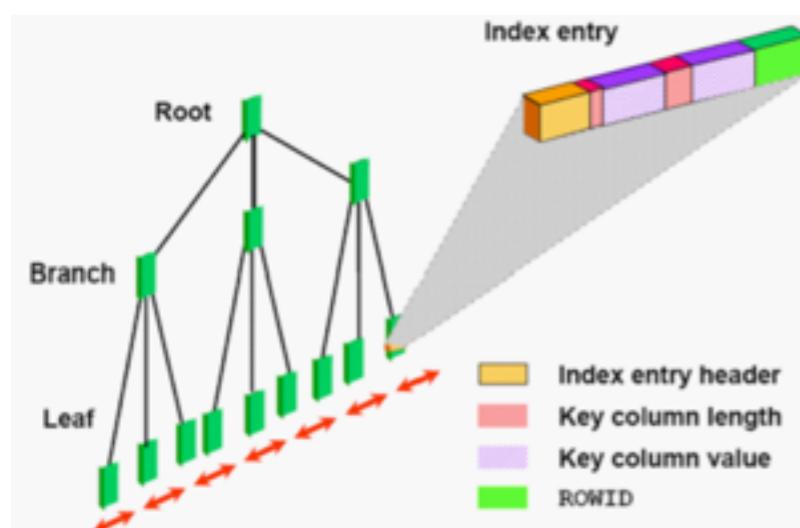
✓ **Não é mais aconselhável usar índices simples**, uma vez que:

- 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



✓ Utiliza-se outras estruturas de dados

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



Acesso por múltiplas chaves

- ✓ Como saber qual é a chave primária do registro que se quer consultar?

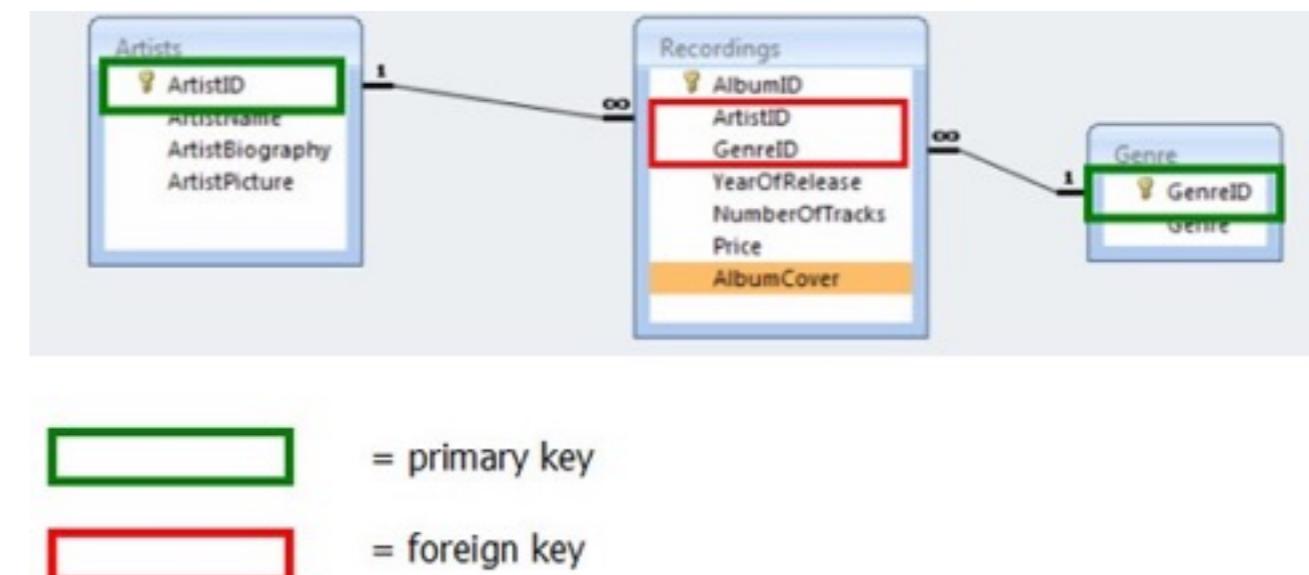


- ✓ Normalmente, o acesso a registros não se faz por chave primária, e sim por chaves secundárias

- Quando se procura por um livro em uma biblioteca, consulta-se pelo seu código ou pelo título/autor?

- ✓ **Solução:** criar um índice que relaciona uma chave secundária à chave primária (e não diretamente ao registro)

- **Índice secundário**



= primary key



= foreign key

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



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

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

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 ser 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

Remover registro

- ✓ Implica em **excluir** (ou marcar) o registro do arquivo de dados e de todos os **índices**
- ✓ Índices primário e secundários são mantidos ordenados segundo a chave. Consequentemente, a remoção poderá requerer o **rearranjo dos registros remanescentes** para não deixar “espaços vazios”
- ✓ **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
- Fazer “coletas de lixo” periódicas nos índices secundários
- Ainda será um problema se o arquivo for muito volátil



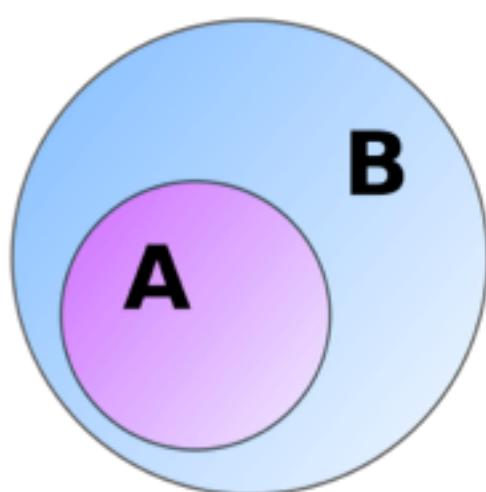
Atualizar registro: três situações

- ✓ Alterou um campo que é chave secundária: atualizar e reordenar o índice secundário
- ✓ Alterou o campo da chave primária: atualizar e reordenar o índice primário e corrigir os campos de referência nos índices secundários
- ✓ 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 os resultados usando operadores (AND ,OR, NOT)
- ✓ Ex: encontrar todos os registros de dados, tal que
 - **composer** = “BEETHOVEN” **AND**
 - **title** = “SYMPHONY NO. 9”



Busca usando 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

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

Busca usando múltiplas chaves

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

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

Busca usando múltiplas chaves

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

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

Busca usando múltiplas chaves

Composer index	
<i>Secondary key</i>	<i>Primary key</i>
BEETHOVEN	ANG3795
BEETHOVEN	DG139201
BEETHOVEN	DG18807
BEETHOVEN	
COREA	
DVORAK	
PROKOFIEV	
RIMSKY-KORSAKOV	MER75016
SPRINGSTEEN	COL38358
SWEET HONEY IN THE R	FF245

Title index	
<i>Secondary key</i>	<i>Primary key</i>
COQ D'OR SUITE	MER75016
GOOD NEWS	FF245
NEBRASKA	COL38358
	RCA2626
	LON2312
	ANG3795
	COL31809
SYMPHONY NO. 9	DG18807
TOUCHSTONE	WAR23699
VIOLIN CONCERTO	DG139201

Matching: tira vantagem da ordenação das chaves associadas a uma chave secundária
→ algoritmos

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

A
Z
↓

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

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

Melhoria de índices secundários

Solução 1: associar um vetor de tamanho fixo a cada chave secundária

<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			

Solução 1: associar um vetor de tamanho fixo a cada chave secundária

✓ Vantagem

- Não é necessário reordenar o índice a cada inserção de registro



✓ Desvantagem

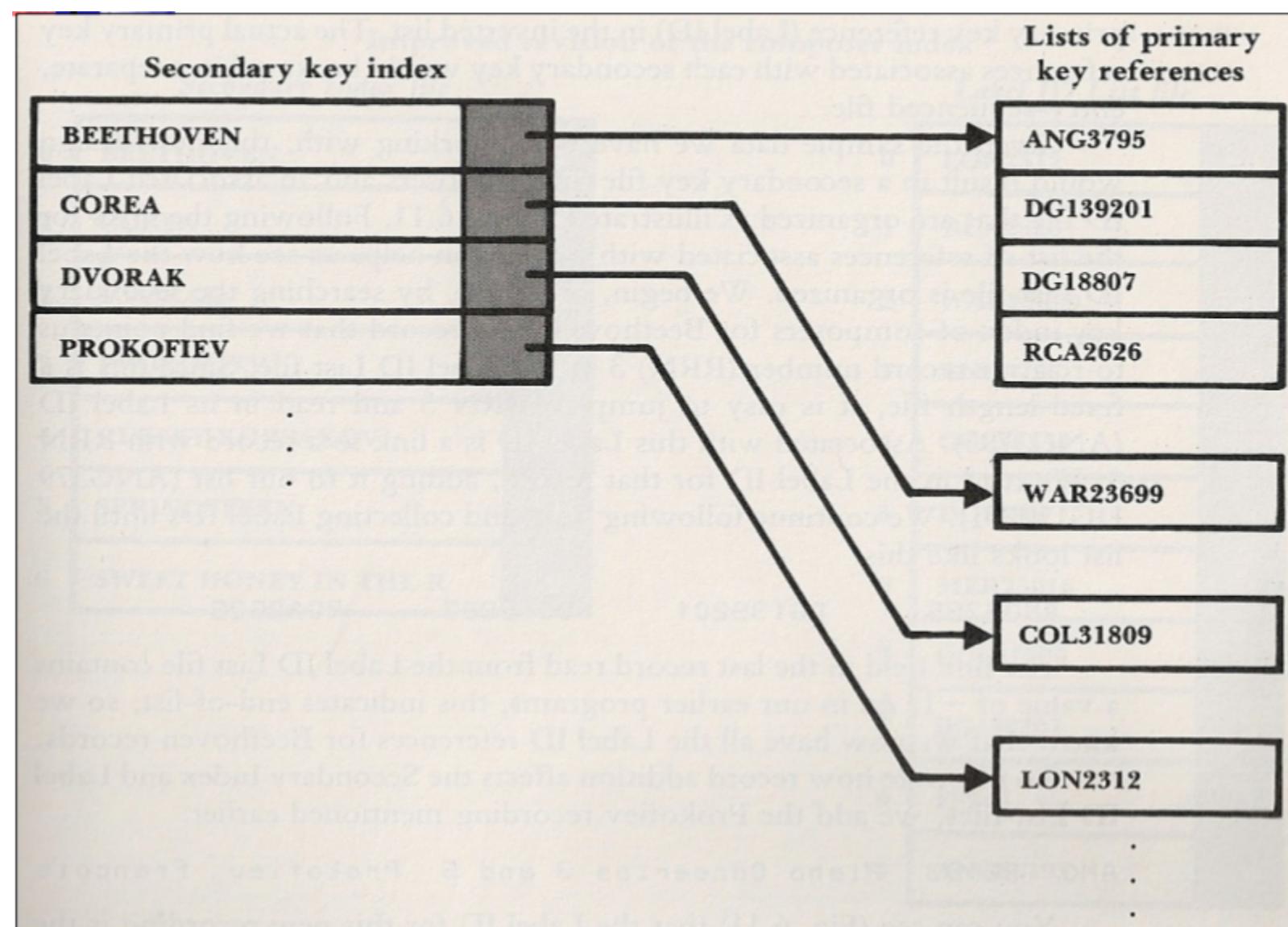
- Limitado a um número fixo de chaves primárias
- Ocorre fragmentação interna enorme no índice - que talvez não compense a eliminação da duplicação das chaves secundárias



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



Solução 2: listas invertidas

- ✓ Índice secundário passa a ser composto por registros com dois 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

Improved revision of the composer index		
	Secondary Index file	Label ID List 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

Vantagens

- ✓ Índice secundário só é alterado quando é inserido um registro com chave inexistente ou quando é alterada uma chave já existente
- ✓ Operações de remoção, inserção ou alteração de registros implicam apenas em alterar o arquivo da lista invertida
- ✓ Ordenação do arquivo de índice secundário é mais rápida: menos registros (e registros menores)
- ✓ O arquivo com a lista de chaves de referência não precisa ser ordenado, pois é “entry-sequenced”
- ✓ É fácil reutilizar o espaço liberado pelos registros eliminados do arquivo da lista de referência



Desvantagem

- ✓ Registros associados na lista de referência podem não estar adjacentes no disco: podem ser necessários várias operações de *seek* para recuperar a lista
- ✓ O ideal seria manter o índice secundário e a lista de referência na memória principal

