



Índices

SCC-503 Algoritmos e Estruturas de Dados II

Thiago A. S. Pardo

Leandro C. Cintra

M.C.F. de Oliveira



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 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)
- **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

Rec. addr.	Label	ID number	Title	Composer(s)	Artist(s)
32†	LON	2312	Romeo and Juliet	Prokofiev	Maazel
77	RCA	2626	Quartet in C Sharp Minor	Beethoven	Julliard
132	WAR	23699	Touchstone	Corea	Corea
167	ANG	3795	Symphony No. 9	Beethoven	Giulini
211	COL	38358	Nebraska	Springsteen	Springsteen
256	DG	18807	Symphony No. 9	Beethoven	Karajan
300	MER	75016	Coq d'or Suite	Rimsky-Korsakov	Leinsdorf
353	COL	31809	Symphony No. 9	Dvorak	Bernstein
396	DG	139201	Violin Concerto	Beethoven	Ferras
442	FF	245	Good News	Sweet Honey in the Rock	Sweet Honey in the Rock

†Assume there is a header record that uses the first 32 bytes.

Índice Simples

Indexfile			Datafile
Key	Reference field	Address of record	Actual data record
ANG3795	167	32	LON 2312 Romeo and Juliet Prokofiev . . .
COL31809	353	77	RCA 2626 Quartet in C Sharp Minor . . .
COL38358	211	132	WAR 23699 Touchstone Corea . . .
DG139201	396	167	ANG 3795 Symphony No. 9 Beethoven . . .
DG18807	256	211	COL 38358 Nebraska Springsteen . . .
FF245	442	256	DG 18807 Symphony No. 9 Beethoven . . .
LON2312	32	300	MER 75016 Coq d'or Suite Rimsky . . .
MER75016	300	353	COL 31809 Symphony No. 9 Dvorak . . .
RCA2626	77	396	DG 139201 Violin Concerto Beethoven . . .
WAR23699	132	442	FF 245 Good News Sweet Honey In The . . .



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**



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 de índice impõem um **limite ao tamanho da chave primária**
 - O que acontece se a chave primária extrapolar o limite imposto?
- Os registros do **índice** poderiam **conter outros campos** além da chave/*offset* (por exemplo, o tamanho do registro a que se referem)



Índice simples

- Como são feitas as operações básicas?
 - Inserção de registros
 - Remoção de registros
 - Atualização de registros
 - Busca de registros

Operações básicas no índice



- Para índices que cabem em memória
 - Criar arquivos de 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 (pode fazer compactação?)
 - Remove também do índice
 - Índice pode ser reorganizado ou se pode apenas marcar os registros excluídos



Operações básicas no índice

- Para índices que cabem em memória
 - Atualizar registro - duas categorias
 - Muda o valor da chave
 - Índice deve ser atualizado
 - Muda o conteúdo do registro
 - Pode ou não alterar o tamanho do registro, podendo exigir realocação 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 de 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 em duplas

- Implementar em C uma sub-rotina que construa um índice a partir de um arquivo de dados de alunos e que, depois, percorra esse índice para encontrar um aluno (pelo número)

```
struct aluno {  
    char nome[50];  
    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 no índice 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 maior prioridade
 - Acesso direto apenas
 - *Árvores-B*, caso se deseje combinar acesso por chaves e acesso seqüencial eficientemente



Acesso por múltiplas chaves

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



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
 - Quando se procura por um livro em uma 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



Acesso por 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 G SHARP M	RCA2626
ROMEO AND JULIET	LON2312
SYMPHONY NO. 9	ANG3795
SYMPHONY NO. 9	COL31809
SYMPHONY NO. 9	DG18807
TOUGHSTONE	WAR23699
VIOLIN CONCERTO	DG139201



Alterações nas operações básicas

■ Inserir registro

- Quando um **novο 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



Alterações nas operações básicas

- **Eliminar registro**

- Implica em **remover** o registro do arquivo de **dados** e de todos os **índices**
- 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 é muito volátil
 - Outra solução: 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**: **reordenar o índice primário** e corrigir os campos de referência dos índices secundários
 - Vantagem: 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 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 G SHARP M	RCA2626
ROMEO AND JULIET	LON2312
SYMPHONY NO. 9	ANG3795
SYMPHONY NO. 9	COL31809
SYMPHONY NO. 9	DG18807
TOUGHSTONE	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 G SHARP M	RCA2626
ROMEO AND JULIET	LON2312
SYMPHONY NO. 9	ANG3795
SYMPHONY NO. 9	COL31809
SYMPHONY NO. 9	DG18807
TOUGHSTONE	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 G SHARP M	RCA2626
ROMEO AND JULIET	LON2312
SYMPHONY NO. 9	ANG3795
SYMPHONY NO. 9	COL31809
SYMPHONY NO. 9	DG18807
TOUGHSTONE	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
TOUGHSTONE	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



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

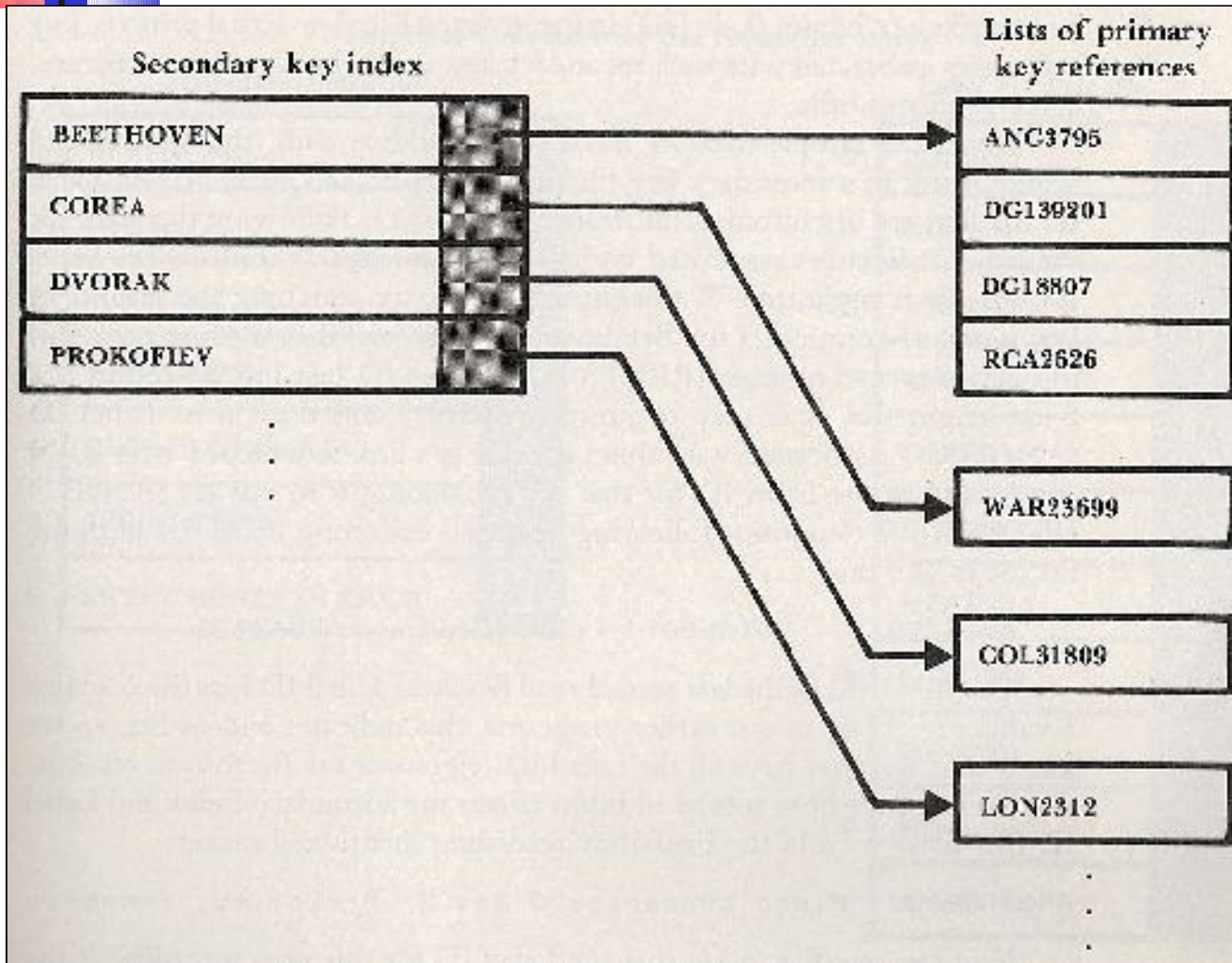
BEETHOVEN	ANG3795	DG139201	DG18807	RCA2626
COREA	WAR23699			
DVORAK	COL31809			
PROKOFIEV	LON2312			
RIMSKY-KORSAKOV	MER75016			
SPRINGSTEEN	COL38358			
SWEET HONEY IN THE R	FF245			



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
 - **Referências às chaves primárias** associadas a cada chave secundária são mantidas em um **arquivo seqüencial separado**, organizado segundo a entrada dos registros

Listas invertidas: visão conceitual



Por que se chama lista invertida?

Listas invertidas

Improved revision of the composer index

Secondary Index file

0	BEETHOVEN	8
1	COREA	2
2	DVORAK	7
3	PROKOFIEV	10
4	RIMSKY-KORSAKOV	5
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	5
6	MER75016	-1
7	COL31809	-1
8	DG139201	9
9	FF245	-1
10	ANG36193	0



Listas invertidas

- Vantagens desta estratégia?



Listas invertidas

- **Vantagens**

- Índice secundário só é alterado quando é inserido um registro com chave inexistente, ou quando é alterada uma chave já existente
 - Eliminação, inserção ou alteração de registros já existentes implicam apenas em alterar 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 a ordem de entrada é mantida
- É fácil reutilizar o espaço liberado pelos registros eliminados do arquivo de listas



Listas invertidas

- Problemas desta estratégia?



Listas invertidas

- **Problemas**

- Registros associados não estão adjacentes no disco: podem ser necessários vários *seeks* para recuperar a lista
- O ideal seria manter o índice e a lista na memória



Índices seletivos

- O 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
 - Caso contrário, o que acontece?



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
 - Caso contrário, o que acontece?
 - Muita atualização de índice secundário



Estudo de caso

- *Como ferramentas de busca na Internet funcionam?*
 - *Criação e atualização de índices de palavras-chave e onde encontrá-las*
 - *Definição de uma ordenação de páginas baseada na “importância” ou “relevância”*
- *“Como o Google funciona”*
 - *<http://www.google.com/howgoogleworks/>*