Algoritmos e Estruturas de Dados II

Prof. Debora Medeiros

Indexação de Arquivos II:

Índices Simples Grandes & Indexação Secundária

Adaptado e Estendido dos Originais de:

Ricardo J. G. B. Campello
Leandro C. Cintra
Maria Cristina F. de Oliveira

Arquivo de Índice (Revisão)

- Exemplo Prático (Arquivo de Músicas)
  - Registros de tamanho variável com:
    - **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 Índice (Revisão)

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?

Arquivo de Índice (Revisão)

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.

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
  - Nada muda para o arquivo principal, que é manipulado em memória secundária sempre
- **Busca?**

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
  - Nada muda para o arquivo principal, que é manipulado em memória secundária sempre
- **Busca**
  - Busca seqüencial é O(n) acessos, mesmo com blocagem
  - BB é O(log n) acessos, mas não se beneficia de blocagem
    - pode demandar um acesso para cada registro verificado
    - Não há melhorias em relação à busca no arquivo de dados

Arquivos de Índice Grandes

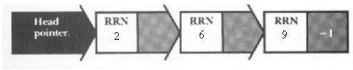
- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
- **Remoção?**

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
- **Remoção**
  - Alternativa 1: Deslocar todos os registros subseqüentes no arquivo de índice para preencher espaço do registro removido
    - otimiza espaço, mas a um custo computacional altíssimo...
  - Alternativa 2: Colocar um marcador e encadear o registro removido em uma lista de registros de índice disponíveis
    - análogo ao que é feito para o arquivo principal

Arquivos de Índice Grandes

- **Remoção**
  - Alternativa 2 (Exemplo):



  - Limitação?

head.first_avail = 2

RRN	Key	Reference field
0	ANG3795	152
1	COL31809	338
2	*|6|COL38358	196
3	DG139201	382
4	DG18807	241
5	FF245	427
6	*|9|LON2312	17
7	MER75016	285
8	RCA2626	62
9	*|-1|WAR23699	117

Index

Arquivos de Índice Grandes

- **Remoção**
  - Alternativa 2 (limitação):
    - inserção deverá respeitar ordem da chave para permitir BB ...
    - pode não valer a pena manter e percorrer a lista de disponíveis com baixa possibilidade de sucesso ...
    - Solução?

head.first_avail = 2

RRN	Key	Reference field
0	ANG3795	152
1	COL31809	338
2	*|6|COL38358	196
3	DG139201	382
4	DG18807	241
5	FF245	427
6	*|9|LON2312	17
7	MER75016	285
8	RCA2626	62
9	*|-1|WAR23699	117

Index

Arquivos de Índice Grandes

- **Remoção**
  - Alternativa 3:
    - Apenas marcar os registros como disponíveis (sem lista)
    - Como funciona inserção nesse caso?

RRN	Key	Reference field
0	ANG3795	152
1	COL31809	338
2	* COL38358	196
3	DG139201	382
4	DG18807	241
5	FF245	427
6	* LON2312	17
7	MER75016	285
8	RCA2626	62
9	* WAR23699	117

Index

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
- **Inserção** (alternativa 3 de remoção)
  - Para permitir BB, chave inserida deve respeitar ordem do índice
    - Busca-se pela localização onde a chave deveria ser inserida (BB)
    - Se localização corresponde a um slot disponível, tudo resolvido
    - Caso contrário, é necessário deslocar todos os registros de índice subseqüentes até o próximo slot vago ou EOF

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
- **Atualização?**

Arquivos de Índice Grandes

- Se o índice não cabe na memória primária, o acesso e manutenção precisam ser feitos em memória secundária
- **Atualização**
  - Se atualização muda o valor da chave:
    - trata-se como uma remoção do reg. de índice antigo seguida de uma inserção do reg. de índice atualizado
  - Se atualização não muda o valor da chave:
    - se tamanho do registro não aumenta, nada muda no índice
    - caso contrário, muda-se apenas o byte offset no índice

Arquivos de Índice Grandes

- Desempenho das operações em arquivos de índices simples que não cabem em RAM só pode ser melhorado com abordagens de indexação mais sofisticadas:
  - Hashing Externo
    - Máximo desempenho para acesso direto
  - Árvores
    - Bom compromisso entre desempenho, manutenibilidade e possibilidade de acesso seqüencial ordenado por chaves

Arquivos de Índice Grandes

- Mesmo não cabendo na RAM, índices simples têm vantagens
  - Possibilitar BB para registros de tamanho variável
  - Mesmo não cabendo na RAM, é menor que o arquivo de dados
    - Menos dados a serem tranferidos
  - *Pinned records*
    - Rearranjar chaves sem mover registros
  - Índices múltiplos

Indexação Secundária

- O que fazer quando a chave primária não é o alvo da consulta?
  - Por exemplo, enquanto CPF é uma chave muito usual, o que dizer do código do nosso arquivo de músicas?
    - Como saber que se deve procurar por **COL38358** quando se deseja a ficha musical de "Nebraska", de Bruce Springsteen ???
    - Exemplo de livros em uma biblioteca

Indexação Secundária

- Muitas vezes, o acesso a registros não se faz por chave primária, mas por chaves secundárias
- Como localizar o registro, se nosso índice é construído em função da chave primária?
- **Solução?**

Indexação Secundária

- Muitas vezes, o acesso a registros não se faz por chave primária, mas por chaves secundárias
- Como localizar o registro, se nosso índice é construído em função da chave primária?
- **Solução**:
  - cria-se um outro índice que relaciona uma chave secundária à chave primária (late binding)
  - usa-se então o índice da chave primária para localizar o registro

Indexação Secundária

- Exemplo Prático (Arquivo de Músicas):

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

Indexação Secundária

- Por que não associar o índice secundário à localização dos registros?
- Forma canônica
- Tamanho fixo
  - Truncamento se necessário
  - Incluir na regra da forma canônica

Indexação Secundária

- Índices permitem muito mais que melhorar o tempo de localização de um registro
- Múltiplos índices secundários:
  - permitem manter diferentes visões dos registros em um mesmo arquivo de dados
  - permitem combinar chaves associadas e fazer consultas que combinam visões particulares

Indexação Secundária

- Diferença importante entre os índices dos tipos primário e secundário?

Indexação Secundária

- Diferença importante entre os índices dos tipos primário e secundário:
  - Nos secundários, podem ocorrer múltiplos registros com chaves iguais
  - Chaves duplicadas devem ser mantidas agrupadas e ordenadas internamente ao grupo segundo a chave primária
    - Permite consultas eficientes envolvendo combinações de chaves secundárias...

Operações Básicas

- **Remoção:**
  - Implica em remover o registro do arquivo de dados e de todas as referências a ele nos arquivos de índices
  - Buscar o registro e eventualmente gerenciar os espaços vagos em múltiplos índices
    - pode ser custoso se não couberem em RAM
  - Alternativa...

Operações Básicas

- **Remoção:**
  - **Alternativa:** atualizar apenas o índice primário, sem eliminar as entradas correspondentes nos índices secundários
  - Se um índice secundário referenciasse a localização do registro?
    - Problema de *pinned record*
    - Ex: registro apagado e seu espaço reaproveitado

Operações Básicas

- **Remoção** (alternativa)**:**
  - Atualizar apenas o índice primário, sem eliminar as entradas correspondentes nos índices secundários
    - É mais simples e menos sujeito a inconsistências
    - A busca irá apenas ser mal sucedida ao procurar, a partir de uma referência não atualizada no arquivo de índice secundário, por uma chave primária que não mais existe
      - Nesse momento, é possível eliminar o registro do índice secundário
      - Porém, existe um custo computacional extra associado
        - Busca por chave inexistente no índice primário
    - Remoção periódica

Operações Básicas

- **Inserção:**
  - Quando um novo registro é inserido no arquivo, devem ser inseridas as entradas correspondentes no índice primário e nos índices secundários
    - entradas devem ser inseridas respeitando a ordenação
    - se os arquivos de índices não couberem em RAM, pode ser muito custoso
      - especialmente quando a remoção alternativa é adotada, o que implica necessariamente o deslocamento de todos os registros subseqüentes à posição de inserção até o final do arquivo.

Operações Básicas

- **Atualização** (3 situações):
  - Situação 1: Alterou uma chave secundária
    - ...

Operações Básicas

- **Atualização** (3 situações):
  - Situação 1: Alterou uma chave secundária
    - índice secundário para esta chave precisa ser reordenado
    - operação relativamente custosa

Operações Básicas

- **Atualização** (3 situações):
  - Situação 2: Alterou a chave primária
    - ...

Operações Básicas

- **Atualização** (3 situações):
  - Situação 2: Alterou a chave primária
    - índice primário precisa ser reordenado
    - índices secundários precisam ser varridos e as entradas contendo a chave primária alterada devem ser atualizadas
      - se houver chaves secundárias duplicadas, pode ser necessário reordená-las localmente pela chave primária

Operações Básicas

- **Atualização** (3 situações):
  - Situação 3: Alterou apenas outros campos
    - ...

Operações Básicas

- **Atualização** (3 situações):
  - Situação 3: Alterou apenas outros campos
    - não afeta nenhum dos índices secundários
    - no máximo é preciso atualizar o valor do byte offset no respectivo registro do índice primário
      - Porque...?

Busca com Múltiplas Chaves

Arquivo de Índice (Revisão)

- Exemplo Prático (Arquivo de Músicas)
  - Registros de tamanho variável com:
    - **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 Índice (Revisão)

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.

Arquivo de Índice (Revisão)

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.

Indexação Secundária (Revisão)

- Exemplo Prático (Arquivo de Músicas):

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 com 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 (consulta) em vários índices e combinar (AND, OR, NOT) os resultados individuais
- Exemplo: Encontre todos os registros tal que
  - salário > R$3000 **OR** tempo_serviço > 10 anos
- Imagine fazer isso sem índice secundário...

Busca com Múltiplas Chaves

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

Busca com Múltiplas Chaves

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

ANG3795
DG139201
DG18807
RCA2626

AND

ANG3795
COL31809
DG18807

  - E se fossem muitos registros?
    - Algoritmo?

Busca com Múltiplas Chaves

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

ANG3795
DG139201
DG18807
RCA2626

AND

ANG3795
COL31809
DG18807

  - Co-processamento seqüencial dos arquivos (Cap. Seguinte...) !
    - beneficia-se da ordenação local pelas chaves primárias !

resultado ➡

```
ANG|3795|Symphony No. 9|Beethoven|Giulini
DG|18807|Symphony No. 9|Beethoven|Karajan
```


Índices Secundários Melhorados

- Problemas nas estruturas de índices vistas até agora...
  - Espaço?

Índices Secundários Melhorados

- Problemas nas estruturas de índices vistas até agora:
  - repetição de chaves secundárias
    - arquivos de índices secundários maiores que o necessário
  - Inserção de um novo registro?

Índices Secundários Melhorados

- Dois problemas nas estruturas de índices vistas até agora:
  - repetição de chaves secundárias
    - arquivos de índices secundários maiores que o necessário
  - necessidade de rearranjar os índices mesmo quando um novo registro que tenha um valor de chave secundária já existente no arquivo seja inserido
    - P. ex. se uma nova gravação da sinfonia no. 9 de Beethoven for inserida no nosso arquivo de música
- Como melhorar?

Índices Secundários Melhorados

- **Solução 1:** Associar um conjunto (vetor) de chaves primárias (tamanho fixo) a cada chave secundária
  - No mesmo registro

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			

Índices Secundários Melhorados

- **Solução 1:**
  - Elimina entradas com chaves secundárias duplicadas;
  - No caso de inserção de registro com chave secundária existente, não é necessário:
    - inserir um novo registro e
    - reordenar o índice
  - Modificações são feitas no registro correspondente no índice secundário

Índices Secundários Melhorados

- **Solução 1:**
  - Porém ...

Índices Secundários Melhorados

- **Solução 1:**
  - Porém ...
    - É limitado a um número fixo de repetições da chave
    - Quanto maior esse número, maior a fragmentação interna do arquivo de índice !
      - talvez não compense a eliminação das chaves duplicadas
  - Como resolver?

Listas Invertidas (visão conceitual)

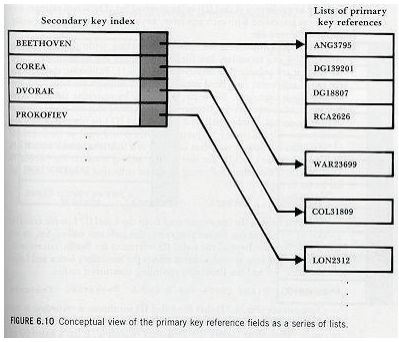


FIGURE 6.10 Conceptual view of the primary key reference fields as a series of lists.

Índices Secundários Melhorados

- **Solução 2:** Listas invertidas
  - Associar cada chave secundária a uma lista encadeada (denominada invertida) dessas chaves primárias
  - Substitui-se a referência à chave primária nos registros do arquivo de índice secundário por uma referência ao RRN do primeiro registro com essa chave na lista invertida
  - Listas invertidas são mantidas em um arquivo seqüencial separado, organizado segundo a entrada dos registros
    - *entry sequenced file*

Índices Secundários Melhorados

- Exemplo:

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

Lista Invertida

Lista Invertida

Improved revision of the composer index

	Secondary Index file			*Label ID List file*	
0	BEETHOVEN	3	0	LON2312	-1
1	COREA	2	1	RCA2626	-1
2	DVORAK	7	2	WAR23699	-1
3	PROKOFIEV	0	3	ANG3795	8
4	RIMSKY-KORSAKOV	6	4	COL38358	-1
5	SPRINGSTEEN	4	5	DG18807	1
6	SWEET HONEY IN THE R	9	6	MER75016	-1
			7	COL31809	-1
			8	DG139201	5
			9	FF245	-1

Índice Secundário

Lista Invertida

- **Vantagens:**
  - Índice secundário só precisa ser alterado quando:
    - ...

Lista Invertida

- **Vantagens:**
  - Índice secundário só precisa ser alterado quando:
    - inserido um registro com chave secundária ainda não existente
    - removido registro cabeça de lista invertida (talvez o único...)
      - -1 no campo de referência
    - alterada uma chave (primária ou secundária) já existente
  - Quando necessário, rearranjar o índice é mais simples:
    - contém menos registros; e
    - não existe duplicidade de chaves secundárias
  - Pode ser feito com as técnicas de manutenção de arquivos de índice ordenados discutidas anteriormente

Lista Invertida

- **Vantagens:**
  - Em muitos casos, as operações de remoção, inserção ou alteração de registros no arquivo de dados implicam apenas em alterar o arquivo de listas invertidas
  - Arquivo de listas invertidas nunca precisa ser ordenado, pois é *entry sequenced*
    - única preocupação é encadear cada lista de forma ordenada segundo a chave primária
  - Logo, é trivial reutilizar o espaço liberado por registros eliminados do arquivo de listas invertidas

Lista Invertida

- **Problema**
  - Registros associados a cada valor de chave secundária, encadeados em uma mesma lista invertida não estão adjacentes no arquivo lógico e no disco:
    - podem ser necessários vários *seeks* para recuperar uma lista
    - E o índice secundário original?

Lista Invertida

- **Problema**
  - Registros associados a cada valor de chave secundária, encadeados em uma mesma lista invertida não estão adjacentes no arquivo lógico e no disco:
    - podem ser necessários vários *seeks* para recuperar uma lista
    - o que podia ser evitado com o índice secundário original
  - O ideal seria manter o índice e as listas em RAM
  - Quando não é possível, é recomendável pensar em estruturas de indexação mais sofisticadas

Binding

- Nos índices primários, a associação (*binding*) entre a chave primária e a localização do registro a que ela se refere ocorre no momento em que o registro é criado e introduzido no arquivo de índices
  - Fornece acesso direto rápido a um registro, dada a sua chave
  - Recaptulando:
    - por que isso não pode ser feito com chaves secundárias?
    - Problema de manter índices atualizados.

Binding

- Já as chaves secundárias são associadas às localizações apenas no momento em que são de fato usadas (*late binding*)
  - Dada a chave secundária busca-se pela(s) primária(s) e, só então, associa-se a primeira ao endereço de um ou mais registros
  - Isso implica um acesso mais lento
  - Mas também implica manutenção mais eficiente e confiável (localizada)

Binding

- Ressalta-se: é sempre desejável manter as modificações localizadas, o que é possível com o *late binding*
  - Caso contrário: muita atualização de índice secundário
- Em arquivos estáticos (e.g. CD, DVD), no entanto, pode ser mais interessante associar diretamente cada índice secundário à(s) localização(ões) dos registros correspondentes (*early binding*)
  - Não existe manutenção...

Exercícios

- Insira pelo menos 3 novos registros no arquivo de músicas utilizado como exemplo em aula e remova pelo menos 2 e mostre, a cada inserção e remoção, como fica o arquivo de índice secundário com chave "compositor" e o arquivo correspondente de listas invertidas
  - Insira no mínimo 1 registro com chave secundária ainda não existente e no mínimo 2 com chaves já existentes
  - Remova no mínimo 1 registro que seja o único de sua chave secundária e no mínimo 1 cuja chave secundária contenha outros registros.

Bibliografia

- **M. J. Folk and B. Zoellick, *File Structures: A Conceptual Toolkit*, Addison Wesley, 1987.**