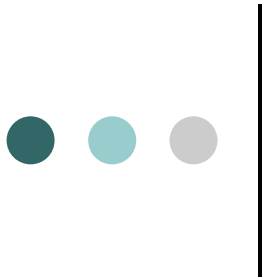




# Capítulo 6

Indexação (Continuação)



# Continuação do Capítulo

1. Índices muito grandes na memória.
2. Acesso por múltiplas chaves.
3. Recuperação por combinação de chaves secundárias.
4. Listas invertidas.
5. Índices seletivos.

# ● ● ● | Índices Muito Grandes na Memória

- O arquivo de índices não cabe na memória
  - ALTERNATIVA: acessar o arquivo de índices no disco
    - A busca binária requer várias operações de *seek* ao disco
      - o que aumenta o tempo gasto na pesquisa.
      - Obs: a busca binária no arquivo de índices não é substancialmente mais rápida do que no arquivo de dados.
  - Use algoritmos especializados que requerem número menor de seeks:
    - B-tree (Capítulo 8) e B+tree (Capítulo 9)
      - usa estruturas não lineares (árvores) para organizar os índices.
    - Hash (Capítulo 10)
      - para os casos onde velocidade de acesso é a maior prioridade.

Péssima  
Idéia!!!

# ● ● ● | Índices Muito Grandes na Memória

- Vantagens do uso de índices, mesmo em memória secundária, sobre o uso do arquivo de dados ordenado por chave:
  - O uso de índices permite o uso de pesquisa binária para a obter acesso via chave num arquivo de registros de tamanho variável.
  - Se os registros do arquivo de índices forem substancialmente menores do que os do arquivo de dados, então ordenação e manutenção dos índices serão menos custosas do que se feitas no arquivo de dados.
  - Se existirem registros (arquivo de dados) excluídos numa LED, o uso de índices nos permite rearranjar as chaves sem mover os registros de dados.

# ● ● ● | Acesso por Múltiplas Chaves

- Buscas via chave primária são raras:
  - “Encontre o registro com a chave DG18807.”
- Rotineiramente se faz acesso a registros por chave secundária:
  - “O que eu quero é a Sinfonia No. 9 do Beethoven.”
- Podemos criar um catálogo para a coleção de gravações, por ex., consistindo de entradas para título do álbum, compositor e artista.



Campos de chave secundária.



# Acesso por Múltiplas Chaves

- O catálogo de uma biblioteca proporciona visões múltiplas da coleção da biblioteca, mesmo que exista apenas um único conjunto de livros organizados em uma única ordem.
- DA MESMA MANEIRA PODEMOS USAR ÍNDICES MÚLTIPLOS PARA PROPORCIONAR VISÕES MÚLTIPLAS DO ARQUIVO DE DADOS.
- Analogia à biblioteca: é muito raro buscar um livro pelo seu número de catálogo. Geralmente se busca por um assunto, título ou autor. Dado o título, autor ou assunto, se procura no catálogo a chave primária.



# Acesso por Múltiplas Chaves

- Pode-se criar tantos índices quantas sejam as chaves de acesso.
- É possível implementar vários índices secundários.
- Cada índice dá uma visão diferente de um arquivo de dados.
  - é possível combinar essas visões:
    - fazendo buscas que sejam a união das respostas de índices secundários.
    - fazendo buscas que sejam a intersecção das respostas de índices secundários.

# ● ● ● | Acesso por Múltiplas Chaves

- Exemplo: um arquivo de índices (secundário) que relaciona Compositor com o LABEL ID.
  - Chave primária: LABEL ID
  - Chave secundária: Compositor (Beethoven)

Secondary key	Primary key
Beethoven	ANG3795
Beethoven	DG139201
Beethoven	DG18807
Beethoven	RCA2626
Corea	WAR23699
Dvorak	COL31809
Prokofiev	LON2312

A referência é para a chave primária, e não para o endereço do registro no arquivo (byte\_offset). Isto significa que o arquivo de índices das chaves primárias deve ser consultado para encontrar o endereço do registro.

OBS: Existem diversas vantagens em não amarrar a chave secundária a um endereço específico.





# Acesso por Múltiplas Chaves

- Observação:
  - No exemplo da biblioteca, uma vez que se obtém o número do catálogo, se pode ir direto às prateleiras pois os livros estão ordenados pelo número de catálogo (que é a chave primária).
  - Já no nosso arquivo os registros estão na ordem de entrada. Por isso, após obter o LABEL ID deve-se ainda consultar o arquivo de índices das chaves primárias.



# Acesso por Múltiplas Chaves

## ○ INCLUSÃO DE REGISTROS:

- As entradas devem ser inseridas no arquivo de índices primário e no de índices secundário.
- Índice secundário:
  - Custo: similar ao de adicionar um registro ao arquivo de índices primário. Decresce muito se o arquivo de índices pode ser mantido e modificado em memória.
  - A chave é armazenada na forma canônica, e em registro de tamanho fixo.
  - Há uma natural duplicação de chaves secundárias.
    - Estas chaves estão agrupadas, e devem estar ordenadas pelo campo de referência (ou seja, pela chave primária).

# ● ● ● | Acesso por Múltiplas Chaves

## ○ ELIMINAÇÃO DE REGISTROS:

- Requer remover todas as referências nos arquivos de índices primário e secundário.
  - implica em reorganizá-los: muito custoso se não cabem na memória.
- Alternativa:
  - Eliminar o registro do arquivo de dados e apenas a referência no arquivo de índices primário (**não modificar o arquivo de índice secundário**).
    - a busca pela chave secundária pode retornar um valor inválido quando retornar uma chave primária inexistente.
      - Como o índice secundário aponta para o índice primário a condição pode ser testada sem causar erro.



# Acesso por Múltiplas Chaves

## ○ ELIMINAÇÃO DE REGISTROS:

### ● Vantagem:

- Se os índices secundários estão associados ao índice primário
  - os índices secundários não precisam ser atualizados.

### ● Custo:

- Espaço ocupado por registros inválidos.
  - pode-se remover esses registros periodicamente.
- Problema se o arquivo for muito volátil.
  - outra solução (por exemplo, uso de *B-Tree*) seria mais aconselhável, pois permite a eliminação sem ter que rearranjar muitos registros.

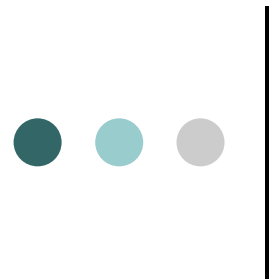
Obs: O registro excluído ainda ocupa espaço no arquivo de índices secundário. Caso muitas exclusões sejam feitas, pode-se periodicamente, limpar o arquivo de índices secundário.



# Acesso por Múltiplas Chaves

## ○ ATUALIZAÇÃO DE REGISTROS:

- Se mudar a chave secundária:
  - Reordene o arquivo de índices secundário
- Se mudar a chave primária:
  - Reordene o arquivo de índices primário e atualize o arquivo de índices secundário (que contém a chave primária)
- Se mudar outro campo:
  - não afeta se o registro tem tamanho fixo.
  - não afeta se o campo alterado couber no atual espaço do registro de tamanho variável.
  - se o campo não couber no atual espaço do registro variável então
    - trate como uma exclusão seguida de inclusão no arquivo de dados.
    - mude a PRR no arquivo de índices primário.



## Recuperação por Combinação de Chaves Secundárias

- Uma das mais importantes aplicações de chaves secundárias:
  - A utilização de duas ou mais chaves combinadas para recuperar um conjunto especial de registros de um arquivo de dados.
    - Encontre todas as gravações com compositor “BEETHOVEN” (chave secundária = compositor)
    - Encontre todas as gravações com o título “Violin Concerto”

# ● ● ● | Recuperação por Combinação de Chaves Secundárias

## ○ Exemplo:

- Encontre todas as gravações com compositor “BEETHOVEN” e (**and**) título “Symphony No.9”

Matches from composer index	Matches from title index	Matched list (logical “and”)
ANG3795	ANG3795	ANG3795
DG139201	COL31809	DG18807
DG18807	DG18807	
RCA2626		

Com o uso de índices secundários, este tipo de busca é simples e rápida, pois as operações lógicas são realizadas nos arquivos de índices, reduzindo assim os seeks.

## ● ● ● | Recuperação por Combinação de Chaves Secundárias

- Com esta facilidade de combinar listas ordenadas rapidamente, podemos recuperar:
  - Intersecções (Beethoven AND Symphony No. 9)
  - Uniões (Beethoven OR Prokofiev OR Symphony No. 9)

O arquivo de dados está na ordem de entrada (inserção temporal), e toda essa combinação pode ser feita sem precisar ordená-lo. A ordenação é feita somente nos pequenos arquivos de índices que geralmente podem ser mantidos em memória.



# ● ● ● | Problemas!!!

## ○ A estrutura de índices secundários/primários possuem duas dificuldades:

1. Requer reorganizar o arquivo de índices todas as vezes que é incluído um novo registro no arquivo.
  - Mesmo que seja para uma chave secundária já existente.
2. Repetição das chaves secundárias
  - Perda de espaço; arquivos maiores sem necessidade.

Secondary key	Primary key
Beethoven	ANG3795
Beethoven	DG139201
Beethoven	DG18807
Beethoven	RCA2626
Corea	WAR23699
Dvorak	COL31809
Prokofiev	LON2312



# Listas Invertidas

- 1a SOLUÇÃO:

- Fazer cada registro do arquivo de índice secundário, consistir da:
  - chave secundária + um array dos campos de referência para aquela chave

BEETHOVEN	ANG3795	DG139201	DG18807	RCA2626
-----------	---------	----------	---------	---------

- Problemas:
  - O array terá um limite de tamanho e podemos ter mais registros para incluir.
  - Podemos ter muitos espaços não utilizados em alguns arrays  
**(fragmentação interna)**

# ● ● ● | Listas Invertidas

É uma lista de chaves primárias associada a uma chave secundária.

- 2a SOLUÇÃO: Listas Invertidas
  - Organizar o arquivo de índices secundário de maneira que cada registro contém a chave secundária e um ponteiro para uma lista encadeada de referências.

**Secondary Key Index File    LABEL ID List File**

0	Beethoven	3
1	Corea	2
2	Dvorak	5
3	Prokofiev	7

0	LON2312	-1
1	RCA2626	-1
2	WAR23699	-1
3	ANG3795	6
4	DG18807	1
5	COL31809	-1
6	DG139201	4
7	ANG36193	0



# Listas Invertidas

- Vantagens:
  - O arquivo de índices secundário só é rearranjado quando um novo compositor é adicionado ou o nome de um compositor é modificado.
    - Adicionar ou excluir gravações para um compositor somente afeta o arquivo contendo a lista LABEL ID.
    - Excluir todas as gravações para um compositor pode ser feito colocando “-1” no campo de referência do arquivo de índices secundário.



# Listas Invertidas

- Vantagens: (cont.)
  - Re-organizar o arquivo de índices secundário é mais rápido, pois ele está menor.
  - O arquivo da lista LABEL ID não precisa estar ordenado, pois está na ordem de entrada (inserção temporal).
  - Podemos facilmente reutilizar o espaço dos registros excluídos do arquivo contendo a lista LABEL ID, pois eles tem tamanho fixo.



# Listas Invertidas

- Desvantagens:
  - Perda de “localidade”
    - Os labels das gravações que tem a mesma chave secundária não são contíguos no arquivo da lista LABEL ID (= muito seeking)
    - Para melhorar esta situação, uma alternativa seria manter o arquivo da lista LABEL ID em memória, ou se for muito grande, utilizar mecanismos de paginação, mantendo somente uma parte do arquivo em memória, conforme vão sendo necessárias.

# ● ● ● | Listas Invertidas

- É possível colocar o arquivo de lista invertida no arquivo de chave principal criando campos para fazer a lista encadeada

Registro	Ch. Principal	Endereço
0	1002	4
1	1002	3
2	1003	1
3	1004	2
4	1005	0

Registro do Ademar: administrativo

Registro do Afonso: técnico

Registro da Lara: comercial

Registro do Carlos: administrativo

Registro da Sônia: administrativo

Ex: arquivo de chave principal



# Listas Invertidas

Registro	Ch. Principal	Endereço (NRR)	Ch. Secun.	Próx.	Ant.
0	0001		Administrativo	3	7
1	0002		Técnico	4	4
2	0003		Comercial	5	5
3	1001	4	Administrativo	6	0
4	1002	3	Técnico	1	1
5	1003	2	Comercial	2	2
6	1004	1	Administrativo	7	3
7	1005	0	Administrativo	0	6

Exemplo de lista invertida juntamente com o arquivo de chave principal (inserida como cabeçalho)





# Arquivos Multilistas

- Os arquivos de índice multilistas são a generalização das listas invertidas. Exemplo:
  - Consideremos um arquivo com dados de alunos
    - Chave principal (CP) a matrícula
    - Chaves secundárias (CS): o estado e IRA
    - Campos incrementais (CI) não são chave principal ou secundária (não podem ser utilizados em busca) porém são frequentemente acessados e incluídos no arq. Índice (ex: nome)
    - Outras informações adicionais (ex: sobrenome, endereço, etc.) são gravados somente no arq. principal.

# ● ● ● | Arquivos Multilistas

- Geramos uma lista com todas as informações que farão parte do arquivo INDEX ordenado pela chave principal

CP	CI	CS1	CS2
1001111	João	MG	4
1002222	José	SP	5
1003333	Maria	RJ	5
1004444	Gérson	MG	5
1100000	Sandra	SP	4
1101111	Miguel	MG	4
1102222	Ana	SP	4



# Listas Invertidas

- Atribuimos a informação de endereço (NRR ou PRR) gerando a lista “A”

CP	ED	CI	CS1	CS2
1001111	1	João	MG	4
1002222	2	José	SP	5
1003333	3	Maria	RJ	5
1004444	4	Gérson	MG	5
1100000	5	Sandra	SP	4
1101111	6	Miguel	MG	4
1102222	7	Ana	SP	4

# ● ● ● | Listas Invertidas

- Ordenamos as triplas por chave secundária e inserimos os ponteiros

CS1	ED	CP	P1
MG	1	1001111	4
MG	4	1004444	6
MG	6	1101111	-1
RJ	3	1003333	-1
SP	2	1002222	5
SP	5	1100000	7
SP	7	1102222	-1

CS2	ED	CP	P1
4	1	1001111	5
4	5	1102222	6
4	6	1101111	7
4	7	1102222	-1
5	2	1002222	3
5	3	1003333	4
5	4	1001111	-1

# ● ● ● | Listas Invertidas

- Geramos arquivo “index” de lista invertidas para cada índice secundário (podendo ou não incluir quantidade de registros de cada valor)

CS1	ED	Quantid.
MG	1	3
RJ	3	1
SP	2	3

CS2	ED	Quantid.
4	1	4
5	2	3

# ● ● ● | Listas Invertidas

- Reordenamos as listas anteriores pela chave principal

CS1	ED	CP	P1
MG	1	1001111	4
SP	2	1002222	5
RJ	3	1003333	-1
MG	4	1004444	6
SP	5	1100000	7
MG	6	1101111	-1
SP	7	1102222	-1

CS2	ED	CP	P1
4	1	1001111	5
5	2	1002222	3
5	3	1003333	4
5	4	1004444	-1
4	5	1100000	6
4	6	1101111	7
4	7	1102222	-1

# ● ● ● | Listas Invertidas

- Incluir os ponteiros na lista “A”. A nova lista será o arquivo “index” de chave principal

CP	ED	CI	CS1	CS2	P1	P2
1001111	1	João	MG	4	4	5
1002222	2	José	SP	5	5	3
1003333	3	Maria	RJ	5	-1	4
1004444	4	Gérson	MG	5	6	-1
1100000	5	Sandra	SP	4	7	6
1101111	6	Miguel	MG	4	-1	7
1102222	7	Ana	SP	4	-1	-1

# ● ● ● | Índices Seletivos

- Podemos construir índices seletivos, tais como:
  - “Gravações lançadas antes de 1970.”
  - “Gravações lançadas desde 1970.”
  - Também podemos utilizar operações booleanas:
    - “Recupere todas as gravações de Beethoven desde 1970”
  - Verifiquem a metodologia de indexação do iTunes(Apple) x Windows Media Player!