Chave de Acesso

Helton Hideraldo Bíscaro

Apresentação

Obtenção de informações usando chaves

- A tarefa de localizar uma informação em um arquivo.
  - Isso implica em fazer uma busca por um registro
  - O registro pode ser facilmente encontrado apenas se o seu RRN.
  - Se o RRN do registro não é conhecido, o acesso a um registro implica em fazer uma pesquisa pelo mesmo, através de uma (ou mais) de suas chaves
- Se pesquisa pela chave requer uma Busca Seqüencial no arquivo $\rightarrow$ Muito caro.
  - a busca por um registro em 1000 requer, em média, 500 comparações e, no pior caso, 1000 comparações.
  - Para um arquivo em disco, cada comparação requer (no pior caso), um acesso a disco para recuperar o registro o custo de um acesso a disco é muito alto quando comparado ao custo de um acesso à RAM

Chaves de Acesso

Obtenção de informações usando chaves

- Acesso através de Pesquisa Binária
  1. complexidade: O(log n)
  2. pior caso: (maior inteiro imediatamente menor que (log n)) + 1 comparações
  3. em média: (maior inteiro imediatamente menor que (log n)) + 1/2 comparações
  4. busca por um registro entre 1000: no máximo, 11 comparações, ou seja, 11 acessos a disco.
- Pesquisa Sequencial X Pesquisa Binária
  1. E se o arquivo a ser pesquisado aumentar? Por exemplo, em vez de 1000, 2000 registros?
  2. sequencial.: pior caso = 2000 acessos, 1000 em media.
  3. binária: pior caso = 11 acessos.
  4. Portanto: a pesquisa binária é obviamente muito mais atrativa, mas só funciona com arquivo ordenado

Chaves de Acesso

Obtenção de informações usando chaves

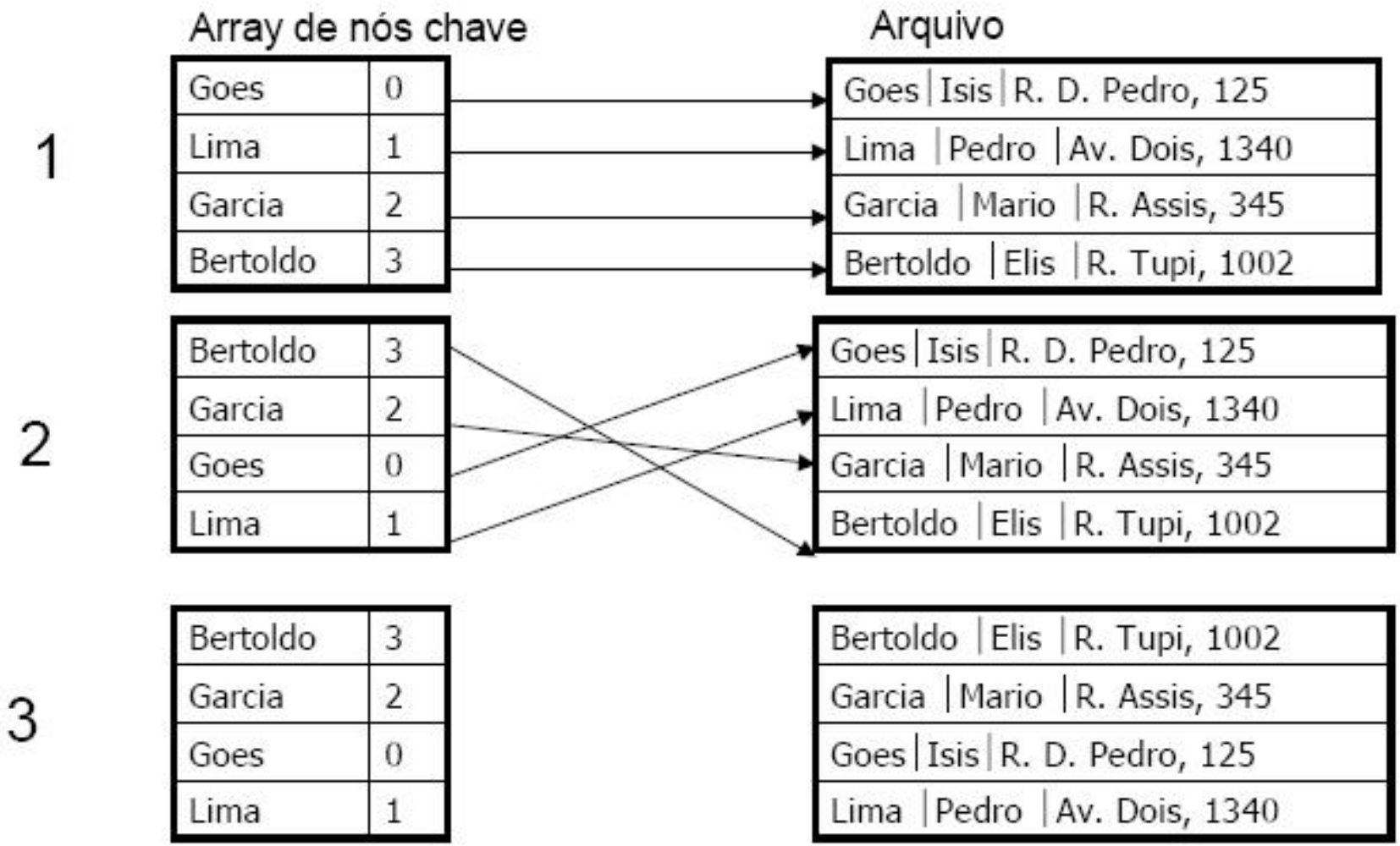
- A ordenação de um arquivo em disco torna-se complicada justamente porque os arquivos estão em disco e não em RAM
- Se o arquivo cabe na RAM, pode-se carregá-lo na memória para ordenar: acessa-se o arquivo duas vezes (para leitura, antes de ordenar, e para escrita, depois de ordenado)
- Se o arquivo não cabe na memória, e o trabalho deve ser feito em disco, temos alguns problemas:
  1. Pesquisa binária representa uma melhora em relação à pesquisa sequencial, mas ainda requer 1, ou 2, ou 8, ou 16,... acessos a disco.
  2. O custo de manter o arquivo ordenado no disco pode ser muito alto....
  3. limita o uso de uma busca binária simples
  4. uma solução pode ser o **keysort**.

Ordenação Key- Sort

Descrição do Método

- Funciona de modo parecido com a ordenação em RAM, mas não utiliza o arquivo todo, apenas as chaves:
  1. lê-se as chaves de cada registro do arquivo para RAM (tem-se pares chave-identificador do registro - RRN).
  2. ordena-se as chaves.
  3. utiliza-se as chaves ordenadas para gerar o novo arquivo, agora ordenado.
- Esforço requerido:
  1. Ler o arquivo seqüencialmente um vez $\longrightarrow$ copiar as chaves
  2. Depois de ordenar as chaves $\longrightarrow$ Acessar cada registro diretamente pela ordem e copiar o registro (seek)
  3. Escrever o registro uma vez, seqüencialmente.

Ordenação Key- Sort - Exemplo



Ordenação Key- Sort

Melhoria do Método

- em vez de escrever um novo arquivo ordenado, escreve-se a lista de chaves ordenadas... a lista de chaves e respectivos RRNs dos registros acaba se tornando um índice para os registros do arquivo.
- Deste modo, a pesquisa por um registro em particular pode ser feita por pesquisa binária, na memória RAM (o índice cabe).

Índices

- Manter arquivos ordenados para permitir pesquisa binária é muito caro.
- Vantagens de se usar um índice simples para um arquivo de dados $\rightarrow$ adicionar registros muito mais rapidamente do que em um arquivo ordenado, desde que o índice seja suficientemente pequeno para ser mantido em memória principal.
- Se o tamanho dos registros no índice é pequeno, esta não é uma condição difícil de ser atendida para arquivos de dados com alguns milhares de registros.
- Por enquanto, vamos assumir que estas condições são válidas, e que o índice é carregado inteiro do disco (onde é mantido) para um vetor de registros denominado INDEX[].

Índices

Índices simples

- consiste simplesmente de vetores com chaves e campos de referência.
- Exemplo: Suponha que temos uma enorme coleção de CD's, e desejamos acessá-los através de um arquivo. Para cada registro, manter as seguintes informações:
  1. Número:→ identificação
  2. Título
  3. Artista
  4. Rótulo → nome da gravadora

54	5
143	4
210	3
323	1
329	2
400	0

INDEX ⇧ RRN

0	400 | Minas | Milton Nascimento | Emi-Odeon | 1975
1	323 | Falso brilhante | Elis Regina | Philips | 1976
2	329 | A Arte de | Chico Buarque | PolyGram | 1982
3	210 | Chico Canta | Chico Buarque | Philips | 1985
4	143 | A Arte de | Milton nascimento | Universal | 1988
5	54 | Geraes | Milton Nascimento | Emi-Odeon | 1976

Manutenção de Índices

1. Criar dois arquivos: de índice e de dados(em disco) $\rightarrow$ inicialmente vazios, apenas com os registros header.
2. Carregar o arquivo de índice para memória no vetor INDEX[]
   - leitura seqüencial do arquivo de índices
   - carregar em um vetor
3. Buscar um registro
   - fazer busca (binária) no vetor de índices
   - obter referência para arquivo de dados
   - ler registro no arquivo de dados

Manutenção de Índices

4. Operações:
   1. adição de registro
      - a inserção deve ser feita no arquivo de dados e no índice.
      - o índice deve ser mantido ordenado pode ser necessário reorganizá-lo.
   2. atualizar índice no disco
      - Quando a cópia em memória foi alterada (inserções e/ou remoções).
      - O que acontece se esta atualização não é feita, ou é feita apenas parcialmente em disco? (Ex: o programa não terminou adequadamente).

Manutenção de Índices

Deve haver um mecanismo que permita saber se o índice está atualizado. Exemplo: Utilizando um flag *out-of-date*

- O 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 deve setado assim que este é carregado na memória, e atualizado (resetado) sempre que o índice é reescrito no disco.
- Todo programa, antes de usar o índice, verifica o flag: se está setado, indica que o arquivo está desatualizado.
- se o programa detecta que o índice está desatualizado, deve existir uma função a ser ativada que reconstrua o índice a partir do arquivo de dados.

Índices Muito Grandes

Se o índice não cabe na memória, o seu acesso e manutenção precisa ser feito em memória **secundária.**

- Não é mais aconselhável usar índices simples:
  - a busca binária pode exigir vários acessos (seek) a disco;
  - a necessidade de deslocar registros nas inserções e remoções de registros tornaria a manutenção do índice excessivamente cara.
- Utilizar:
  - uma organização em hashing para o índice (caso a velocidade de acesso seja a prioridade máxima); ou
  - árvores-B, caso se deseje combinar acesso por chaves e acesso seqüencial eficientemente.

Índices para acesso por múltiplas chaves

- Normalmente, o acesso a registros não se faz por chave primária, e sim por chaves secundárias.
- Criamos um índice que relaciona uma chave secundária à chave primária (e não diretamente ao registro).

Índices permitem muito mais do que simplesmente melhorar o tempo de acesso para a busca por uma chave:

- permitem trabalhar com múltiplos índices secundários.
- arquivos índice permitem manter diferente visões dos registros em um arquivo de dados $\longrightarrow$ assim como um catálogo de biblioteca permite pesquisar livros por autor, título ou assunto
- também permitem combinar chaves associadas e, deste modo, combinar visões particulares.

Melhoria de índices secundários: Listas invertidas

Exemplo: Considerar o Index do exemplo anterior e criar índice para Artista.

54	5
143	4
210	3
323	1
329	2
400	0

INDEX

0	400 | Minas | Milton Nascimento | Emi-Odeon | 1975
1	323 | Falso brilhante | Elis Regina | Philips | 1976
2	329 | A Arte de | Chico Buarque | PolyGram | 1982
3	210 | Chico Canta | Chico Buarque | Philips | 1985
4	143 | A Arte de | Milton nascimento | Universal | 1988
5	54 | Geraes | Milton Nascimento | Emi-Odeon | 1976

Index para Artista

CHICO BUARQUE	210
CHICO BUARQUE	329
ELIS REGINA	323
MILTON NASCIMEN	54
MILTON NASCIMEN	143

Índice secundário

- Campos chave na forma canônica → no formato que será usado na busca.
  Neste caso:
  - letras maiúsculas
  - Tamanho max=15 catracteres
- Para exibir o nome no formato normal

Alterações nas operações básicas

Adição de registro de dados:

- inserir as entradas correspondentes nos índices primário e secundário.
  - Nenhum problema se os índices estão na RAM.
  - pode exigir que registros sejam deslocados, para criar uma posição de
- inserção, ou que um vetor de ponteiros para as estruturas seja rearranjado.
  - o campo chave no índice secundário armazenado na forma canônica
  - o valor pode ser truncado porque o tamanho da chave deve ser mantido fixo
  - a forma canônica deve levar em consideração esta restrição de tamanho para que a busca no índice funcione corretamente.
- Obervações
  - Diferença entre os índices primário e secundário $\longrightarrow$ no secundário pode ocorrer duplicação de chaves.
  - Chaves duplicadas devem ser mantidas agrupadas e ordenadas segundo a chave primária.

Alterações nas operações básicas

Eliminação de registro de dados:

- Remoção do registro do arquivo de dados e de todos os índices
- Rearranjo dos registros remanescentes nos índices primário e secundários $\longrightarrow$ para manter a ordenação pela chave.
- Alternativa para reduzir o rearranjo:
  - Atualizar apenas o índice primário - não eliminar a entrada correspondente ao registro do índice secundário $\longrightarrow$ a busca retorna um valor inválido.
  - Como o índice secundário referencia o índice primário (e não o registro físico no arquivo de dados), se for feita uma busca por um registro já removido essa condição será acusada na busca pela chave primária feita no índice primário, e a não remoção da entrada do índice secundário.

Alterações nas operações básicas

- Vantagem na redução do rearranjo:
  - economia de tempo substancial quando vários índices secundários estão associados ao índice primário.
  - Ainda mais se estes índices estiverem sendo mantidos em disco, e o usuário está no terminal esperando o resultado da operação de remoção
- Custo na redução do rearranjo:
  - O espaço ocupado por registros inválidos. Para reduzir o custo:
  - Fazer *coletas de lixo* periódicas nos índices secundários.
  - Se o arquivo de dados é muito volátil $\longrightarrow$ utilizar árvore-B para a estrutura de índice secundário $\longrightarrow$ permite remoção sem que seja necessário rearranjar muitos registros.

Alterações nas operações básicas

Atualização de registro de dados:

- Índices secundários não fazem referências diretas ao arquivo de dados
- a atualização de registros afeta os índices secundários apenas se as chaves secundárias ou primárias forem alteradas.
- Existem 3 situações possíveis:
  - a atualização alterou uma chave secundária $\longrightarrow$ reordenar o índice secundário para esta chave.
  - a atualização alterou a chave primária $\longrightarrow$ reordenar o índice primário e corrigir os índices secudários (os campos de referência) $\longrightarrow$ a atualização dos índices secundários não requer reorganização
  - alteração em outros campos $\longrightarrow$ não afeta nenhum dos índices.

Busca Usando Múltiplas chaves

uso de uma ou mais chaves para localizar conjuntos de registros do arquivo de dados, fazendo uma busca em vários índices e uma combinação (AND) dos resultados.

Exemplo: arquivo de CDs, e dois arquivos secundários para ele, com chaves Artista e Rótulo.

Algumas consultas possíveis:

- encontre CD com número= 54 (acesso por chave primária);
- encontre CDs com Artista= **Chico Buarque** (chave secundária: Artista);
- encontre todos os CD's com Rótulo= **Philips** (chave secundária: Rótulo).
- consultas que combinem várias chaves: CDs com Artista=**Milton Nascimento**e Rótulo=**Emi-Odeon**
- Para obter a resposta basta fazer um AND das chaves primárias retornadas por cada busca (por Artista e Rótulo), e buscar no índice os endereços dos registros resultantes no arquivo de dados.
- As buscas poderiam ser combinadas de outras formas (por exemplo, uniões - OR).

Busca Usando Múltiplas chaves

Exemplo: arquivo de CDs, e dois arquivos secundários para ele, com chaves Artista e Rótulo

0	400	Minas	Milton Nascimento	Emi-Odeon	1975
1	323	Falso brilhante	Elis Regina	Philips	1976
2	329	A Arte de	Chico Buarque	PolyGram	1982
3	210	Chico Canta	Chico Buarque	Philips	1985
4	143	A Arte de	Milton nascimento	Universal	1988
5	54	Geraes	Milton Nascimento	Emi-Odeon	1976

54	5
143	4
210	3
323	1
329	2
400	0

Índice primário

CHICO BUARQUE	210
CHICO BUARQUE	329
ELIS REGINA	323
MILTON NASCIMEN	54
MILTON NASCIMEN	143
MILTON NASCIMEN	400

Emi-Odeon	54
Emi-Odeon	400
Philips	210
Philips	323
PolyGram	329
Universal	143

Índices secundários

Busca Usando Múltiplas chaves

0	400 | Minas | Milton Nascimento | Emi-Odeon | 1975
1	323 | Falso brilhante | Elis Regina | Philips | 1976
2	329 | A Arte de | Chico Buarque | PolyGram | 1982
3	210 | Chico Canta | Chico Buarque | Philips | 1985
4	143 | A Arte de | Milton nascimento | Universal | 1988
5	54 | Geraes | Milton Nascimento | Emi-Odeon | 1976

54	5
143	4
210	3
323	1
329	2
400	0

Resultado das consultas:

1. encontre CD com número= 54 (acesso pela chave primária);

54 | Geraes | Milton Nascimento | Emi-Odeon | 1976

Busca Usando Múltiplas chaves

0	400	Minas	Milton Nascimento	Emi-Odeon	1975
1	323	Falso brilhante	Elis Regina	Philips	1976
2	329	A Arte de	Chico Buarque	PolyGram	1982
3	210	Chico Canta	Chico Buarque	Philips	1985
4	143	A Arte de	Milton nascimento	Universal	1988
5	54	Geraes	Milton Nascimento	Emi-Odeon	1976

54	5
143	4
210	3
323	1
329	2
400	0

CHICO BUARQUE	210
CHICO BUARQUE	329
ELIS REGINA	323
MILTON NASCIMEN	54
MILTON NASCIMEN	143
MILTON NASCIMEN	400

Resultado das consultas:

2. encontre CD's com Artista= "Chico Buarque"

210	Chico Canta	Chico Buarque	Philips	1985
329	A Arte de	Chico Buarque	PolyGram	1982

Busca Usando Múltiplas chaves

0	400	Minas	Milton Nascimento	Emi-Odeon	1975
1	323	Falso brilhante	Elis Regina	Philips	1976
2	329	A Arte de	Chico Buarque	PolyGram	1982
3	210	Chico Canta	Chico Buarque	Philips	1985
4	143	A Arte de	Milton nascimento	Universal	1988
5	54	Geraes	Milton Nascimento	Emi-Odeon	1976

54	5
143	4
210	3
323	1
329	2
400	0

Emi-Odeon	54
Emi-Odeon	400
Philips	210
Philips	323
PolyGram	329
Universal	143

Resultado das consultas:

3. encontre todos os CD's com Rótulo= "Philips"

210	Chico Canta	Chico Buarque	Philips	1985
323	Falso brilhante	Elis Regina	Philips	1976

Busca Usando Múltiplas chaves

Resultado da consulta: **Encontre CD's com Artista="Chico Buarque"e Rótulo="Philips"**

→ Combinação do resultado de 2 e 3

2. encontre CD's com Artista= "Chico Buarque"

210	Chico Canta	Chico Buarque	Philips	1985
329	A Arte de	Chico Buarque	PolyGram	1982

3. encontre todos os CD's com Rótulo= "Philips"

210	Chico Canta	Chico Buarque	Philips	1985
323	Falso brilhante	Elis Regina	Philips	1976

4. CD's com Artista="Chico Buarque"e Rótulo="Philips"

210	Chico Canta	Chico Buarque	Philips	1985

→ Combinação do resultado de 2 e 3

Melhoria de índices secundários: Listas invertidas

Problemas com as estruturas de índices comuns:

- a repetição das chaves secundárias, que resulta em arquivos maiores (e, portanto, com menores chances de caber na memória);
- a necessidade de reordenação dos índices cada vez que um novo registro é inserido no arquivo, mesmo que esse registro tenha um valor de chave secundária já existente no arquivo.

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

- não é necessário reordenar o índice a cada inserção de registro.
- essa facilidade fica limitada a um número fixo de repetições.
- ocorre fragmentação interna enorme - que talvez não compense a eliminação das chaves duplicadas...

Melhoria de índices secundários: Listas invertidas

Queremos uma solução em que não seja necessário reordenar o índice a cada novo registro inserido; mas que não limite o número de chaves iguais que podem ocorrer no índice; e não cause perda de espaço com fragmentação interna.

Solução 2: Ligar à lista de referências - Listas invertidas

- cada chave secundária é relacionada a uma lista encadeada das chaves primárias referenciadas
- redefinir o índice secundário de forma que ele seja composto por registros com 2 campos: um campo chave, e um campo com o RRN do primeiro registro na lista invertida.
- As referências às chaves primárias associadas a cada chave secundária são mantidas em um arquivo seqüencial separado, organizado segundo a entrada dos registros.
- O índice de chaves secundárias funciona da seguinte forma:
  - cada chave secundária leva a uma ou mais chaves primárias (daí o termo "invertida)
  - a chave secundária leva à lista de chaves primárias, a qual por sua vez leva aos registros...

Busca Usando Múltiplas chaves

Solução 2: Ligar à lista de referências - Listas invertidas

- **Exemplo:Considerando o arquivo de dados de Cds.**

400	Minas	Milton Nascimento	Emi-Odeon	1975
323	Falso brilhante	Elis Regina	Philips	1976
329	A Arte de	Chico Buarque	PolyGram	1982
210	Chico Canta	Chico Buarque	Philips	1985
143	A Arte de	Milton nascimento	Universal	1988
54	Geraes	Milton Nascimento	Emi-Odeon	1976

54	5
143	4
210	3
323	1
329	2
400	0

CHICO BUARQUE	210
CHICO BUARQUE	329
ELIS REGINA	323
MILTON NASCIMEN	54
MILTON NASCIMEN	143
MILTON NASCIMEN	400

Índice secundário

≅

CHICO BUARQUE	3
ELIS REGINA	1
MILTON NASCIMEN	5

+

0	400	-1
1	323	-1
2	329	-1
3	210	2
4	143	0
5	54	4

Índice secundário com lista invertida

Melhoria de índices secundários: Listas invertidas

Vantagens:

- o índice secundário só é alterado quando um registro com uma chave não existente é inserido, ou quando uma chave existente é alterada (nome trocado); operações de eliminação, inserção ou alteração de registros já existentes implicam apenas na mudançaa do arquivo da lista invertida.
- a ordenação do arquivo de índice secundário é mais rápida (menos registros - e registros menores)
- o arquivo com listas de chaves nunca precisa ser ordenado, pois é sequencial
- é fácil reutilizar o espaço liberado pelos registros eliminados do arquivo de listas $\longrightarrow$ registros de tamanho fixo.

Problema:

- Registros associados não estão adjacentes - podem ser necessários vários seeks para recuperar a lista. O ideal seria poder manter o índice e a lista na memória

Índices seletivos

- Uma das vantagens de índices secundários é que eles podem ser utilizados para *dividir* um arquivo em partes, de modo a fornecer visões distintas de um mesmo arquivo.
  - Por exemplo: CD's lançados depois de 1980.
  - Essa facilidade é útil quando o conteúdo de um arquivo pode ser dividido naturalmente e logicamente em categorias.

Associação do índice ao endereço físico do registro

- Em índices primários $\longrightarrow$ a associação ocorre no momento em que o arquivo é criado.
  - fornece acesso direto e, portanto, mais rápido, a um registro, dada a sua chave.
- Em índices secundários $\longrightarrow$ associadas a um endereço apenas no momento em que são de fato usadas
  - Isso é possível visto que os índices não se referem diretamente ao endereço físico dos registros, mas ao índice primário.

Associação do índice ao endereço físico do registro

- Vantagens da associação apenas no último momento (latebinding)
  - mínima quantidade de reorganização quando os registros são adicionados ou excluídos.
  - Alterações importantes são feitas em um lugar em vez de em muitos lugares.
- Desvantagens da associação apenas no último momento:
  - Isso implica em um acesso mais lento, principalmente se os índices estiverem em disco.
  - pesquisa binária no índice secundário + pesquisa binária no índice primário