Algoritmos e Estruturas de Dados II

Profa. Debora Medeiros

Ordenação e Busca em Arquivos
Indexação de Arquivos I: Índices Simples

Adaptado dos Originais de:

Ricardo Campello
Thiago Pardo
Leandro C. Cintra
Maria Cristina F. de Oliveira

Ordenação e busca em arquivos

- É relativamente fácil buscar elementos em conjuntos ordenados
- A ordenação pode ajudar a diminuir o número de acessos a disco

Busca em arquivos

- Já vimos busca seqüencial
  - O(n) → Muito ruim para acesso a disco!
- E a busca binária?
  - Modo de funcionamento?
  - Complexidade de tempo?

Busca binária

- Dificuldade: ?

Busca binária

- Dificuldade: ordenar os dados
  - Necessário para a busca binária
- Alternativa?

Busca binária

- Dificuldade: ordenar os dados
  - Necessário para a busca binária
- Alternativa: ordenar os dados em RAM
  - Ainda é necessário: ler todo o arquivo e ter memória interna disponível

Busca binária

- Limitações
  - Registros de tamanho fixo
  - Requer mais do que 1 ou 2 acessos
    - Por exemplo, em um arquivo com 1.000 registros, são necessários aproximadamente 10 acessos em média ainda é ruim!

Busca binária

- Limitações
  - Manter um arquivo ordenado é muito caro
    - Reordenação sempre que houver uma adição
      - Lista de registros novos
        - Merge
      - Índices, hashing

Busca binária

- Limitações
  - Manter um arquivo ordenado é muito caro
    - Reordenação sempre que houver uma adição
      - Lista de registros novos
        - Merge
      - Índices, hashing
    - Estruturas de dados que permitam rápida reorganização do arquivo
      - Estruturas de árvore
        - Árvores-B

Ordenação

- Alternativa para carregar registros na RAM e ordená-los?
  - Tem como fazer melhor?
  - O que é necessário para ordenar?

Ordenação

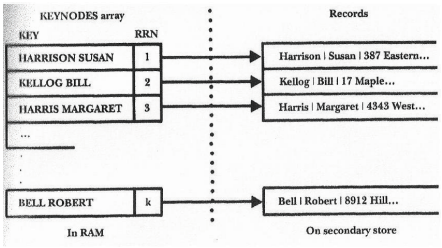
- Alternativa para carregar registros na RAM e ordená-los?
  - Carregar somente as chaves para ordenação
    - Pois elas são essenciais para a ordenação, não o registro todo
    - Possibilidade de ordenar arquivos maiores.
      - Keysorting

Keysorting

- Método
  1. Cria-se na memória interna um vetor, em que cada posição tem uma chave do arquivo e um ponteiro para o respectivo registro no arquivo (RRN ou byte inicial)
  2. Ordena-se o vetor na memória interna
  3. Cria-se um novo arquivo com os registros na ordem em que aparecem no vetor ordenado na memória principal

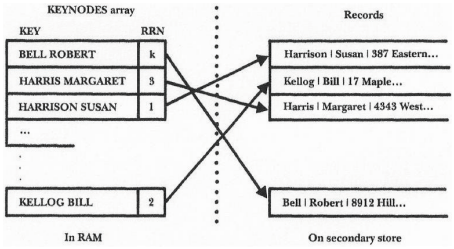
Keysorting

- Exemplo
  - Carregando dados na RAM



Keysorting

- Exemplo
  - Ordenando dados em RAM



Keysorting

- Limitações
  - Inicialmente, é necessário ler as chaves de todos os registros no arquivo
  - Depois, para se criar o novo arquivo, devem-se fazer vários seeks no arquivo para cada posição indicada no vetor ordenado
    - Mais uma leitura completa do arquivo
    - Não é uma leitura seqüencial
    - Alterna-se leitura no arquivo antigo e escrita no arquivo novo

Keysorting

- Questões
  - Por que criar um novo arquivo?
  - Não vale a pena usar o vetor ordenado como um índice?
    - Nesse caso, em um outro arquivo

Questão delicada

- O que fazer com os espaços vazios originados de registros eliminados?
  - Pinned records
    - Referência física
    - Não pode ser movido
    - Dificulta ordenação
    - Utilização de índices

Índice

- Mecanismo para localizar informações via chave
  - mapeamento chave → localização da informação
  - por exemplo, índice de um livro
- No caso de arquivos:
  - permite localizar registros rapidamente
  - evita ter que reorganizar o arquivo de dados conforme este for modificado
    - faça uma analogia com um texto...

Arquivo de Índice

- Um **Arquivo de Índice**:
  - Impõe ordem a um arquivo de dados sem precisar rearranjar o arquivo em si
  - Permite acesso a registros via chave sem precisar varrer o arquivo de dados
  - Permite várias visões diferentes de um mesmo arquivo de dados
    - acesso por múltiplas chaves

Arquivo de Índice

- Estudaremos inicialmente arquivos com **Índices Simples**:
  - Estrutura de dados linear
    - lista de pares (chave , localização)
- Posteriormente no curso veremos índices com EDs mais sofisticadas
  - Por exemplo, árvores

Arquivo de Índice

- 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

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

Index			Recording file
Key	*Reference field*	*Address of record*	*Actual data record*
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 | ...

Figure 7.3 Index of the sample recording file.

Arquivo de Índice

- Cada par (chave , localização) é um registro
  - implementação eficiente usa **registros de tamanho fixo**
    - chave e localização (byte offset) como campos de tamanho fixo
    - pode eventualmente conter outros campos
      - p. ex. tamanho do registro no arquivo de dados
- Em geral, mantido ordenado
  - com registros de tamanho fixo, permite busca binária (BB)
- Menor e mais simples que o arquivo de dados original
  - muitas vezes cabe todo em memória primária!

Arquivo de Índice

- O Arquivo de Dados, em contraste...
  - em geral, muito maior que o arquivo de índices
  - em geral, possui registros de tamanho variável
  - em geral, "organizado" segundo a ordem de entrada dos registros
    - ***entry sequenced file***

Arquivos de Índice Moderados

- A manutenção e busca de registros no arquivo de dados será muito mais eficiente se o arquivo de índice puder ser carregado e manipulado em RAM
  - Isso é possível em muitos casos, quando o arquivo de índice possui tamanho "moderado"

Arquivos de Índice Moderados

- Exemplo:
  - Arquivo de dados com 10^6 registros de ~1Kb em média
    - deve ser indexado até byte offset ~ $10^6 \times 1000$ = 1 Bilhão
    - 4 bytes são mais que suficientes para representar esse offset
  - Arquivo de índice com registros de 24 bytes
    - 4 bytes para o byte offset + 20 bytes para a chave
      - CPF, por exemplo, requer apenas 11 bytes na maior representação
    - Com 10^6 registros, arquivo de índice ocupa 24×10^6 = **24Mb**

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - Quais?

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - **Busca**
    - Depois de carregado o índice, qualquer registro é localizado e recuperado em RAM com **O(1) acessos** externos
      - qq. consulta será O(1), ao preço fixo dos acessos para ler todo o índice
    - Em RAM, localização da chave no índice é muito rápida
      - Se índice não estiver ordenado, busca é seqüencial
      - Mas normalmente mantém-se o índice ordenado, para permitir BB

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - **Inserção**
    - novo registro é inserido no final do arquivo de dados ou segundo uma política do tipo first-fit ou worst-fit
    - um registro associado é também inserido no índice
      - contém a chave e o byte offset do novo registro no arquivo de dados
      - se índice está em vetor ordenado, inserção demanda deslocamentos
        - mas em RAM, isso não demanda qualquer acesso

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - **Remoção**
    - registro é removido do arquivo de dados segundo alguma política de marcação de registros removidos (p. ex. first-fit)
    - o registro associado deve também ser removido do índice
      - deslocamentos ou marcação da célula correspondente do vetor
        - não demanda qualquer acesso

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - **Atualização**
    - **...**

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Carrega-se todo o índice em um vetor
  - **Atualização**
    - Altera-se o registro no arquivo de dados
    - Se atualização mudou o valor da chave (remoção + inserção):
      - altera-se o registro no vetor de índices em RAM
        - chave e, eventualmente, byte offset (porquê ???)
      - reordena-se o vetor de índices
    - Se atualização não mudou o valor da chave:
      - se tamanho do registro não aumenta, nada muda no índice
      - caso contrário, muda-se apenas o byte offset no índice

Operações Básicas

- Para arquivos de índice que cabem em RAM:
  - Ao final de uma seção de operações
    - deve-se atualizar o arquivo de índice no disco
      - caso sua cópia em memória tenha sido alterada
  - É imperativo que o programa se proteja contra índices desatualizados
    - queda de energia
    - crashs do sistema (software ou do hardware)
  - Idéias?

Prevenção de Índices Desatualizados

- Deve haver um mecanismo que permita saber se o índice está atualizado em relação ao arquivo de dados
- Possibilidade:
  - Um *flag* de status é setado no arquivo índice mantido em disco assim que a sua cópia na memória é alterada
  - Esse *flag* pode ser mantido no registro cabeçalho do arquivo índice, e atualizado 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

Bibliografia

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