Algoritmos e Estruturas de Dados II – SCC-203

Arquivos: Índices

Gustavo Batista

Índice

- Índices são estruturas utilizadas para facilitar a localização de informações.
- No caso de arquivos, as principais vantagens são:
  - Permite impor uma ordem no arquivo sem rearranjar o arquivo;
  - Possibilita múltiplas visões sobre o arquivo por meio da criação de mais de um índice;
  - Fornece acesso rápido a arquivos com registros de tamanho variável.

Índice simples

- Ex. Uma enorme coleção de CDs.
- Registros de tamanho variável com os campos:
  - **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.**

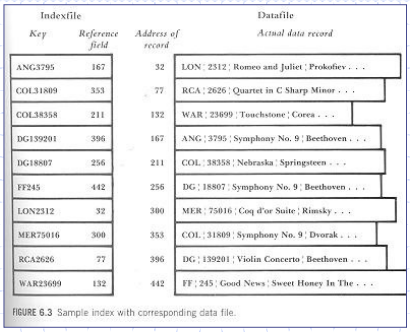
Índice simples

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.

FIGURE 6.2 Sample contents of *Datafile*.

Índice simples



Indexfile

Key	Reference field	Address of record
ANG3795	167	32
COL31809	353	77
COL38358	211	132
DG139201	396	167
DG18807	256	211
FF245	442	256
LON2312	32	300
MER75016	300	353
RCA2626	77	396
WAR23699	132	442

Datafile

Actual data record

LON | 2312 | Romeo and Juliet | Prokofiev . . .
RCA | 2626 | Quartet in C Sharp Minor . . .
WAR | 23699 | Touchstone | Corea . . .
ANG | 3795 | Symphony No. 9 | Beethoven . . .
COL | 38358 | Nebraska | Springsteen . . .
DG | 18807 | Symphony No. 9 | Beethoven . . .
MER | 75016 | Coq d'or Suite | Rimsky . . .
COL | 31809 | Symphony No. 9 | Dvorak . . .
DG | 139201 | Violin Concerto | Beethoven . . .
FF | 245 | Good News | Sweet Honey In The . . .

FIGURE 6.3 Sample index with corresponding data file.

Índice simples

- O índice é ele próprio um arquivo com registros de tamanho fixo.
- Cada registro tem 2 campos de tamanho fixo:
  - um campo contém a chave;
  - outro informa a posição inicial (RRN/*byte offset*) do registro no arquivo de dados.
- A cada registro do arquivo de dados corresponde um registro no arquivo 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 - ***entry sequenced file*** .

Índice simples

- Usa-se dois arquivos:
  - o arquivo índice (*index file*);
  - e o arquivo de dados (*data file*).
- O arquivo índice é:
  - mais fácil de trabalhar, pois usa registros de tamanho fixo;
  - pode ser pesquisado com busca binária (em memória principal);
  - é muito menor do que o arquivo de dados.
- Registros de tamanho fixo no arquivo índice impõem um limite ao tamanho da chave primária.
- Os registros do índice poderiam conter outros campos além da chave/*offset* (por exemplo, o tamanho do registro).

Índice simples em memória principal

- A inclusão de registros será muito mais rápida se o índice pode ser mantido (manipulado) em memória e o arquivo de dados é *entry sequenced* .
- Dados a chave e o *offset*, um único *seek* (i.e., um único acesso a disco) é necessário no arquivo de dados para recuperar o registro correspondente.

Operações básicas no índice

- Para índices que cabem em memória:
  - Criação dos arquivos índice e de dados;
  - Carregar índice para memória;
  - Adição de registro:
    - A inserção deve ser feita no arquivo de dados;
    - E também no índice, que provavelmente será reorganizado.
  - Eliminação de registro:
    - Remove do arquivo de dados, usando algum dos mecanismos de remoção;
    - Remove também do índice. A remoção do registro do índice pode exigir a sua reorganização, ou pode-se simplesmente marcar os registros como removidos.

Operações básicas no índice

- Para índices que cabem em memória:
  - **Atualização de registro**: a atualização cai em duas categorias:
    - Muda o valor da chave;
    - Muda o conteúdo dos demais campos do registro.
  - **Atualizar índice no disco**: caso sua cópia em memória tenha sido alterada:
    - É imperativo que o programa se proteja contra índices desatualizados.

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 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 é ativada que reconstrói o índice a partir do arquivo de dados.

Índices muito grandes

- Se o índice não cabe inteiro na memória, o seu acesso e manutenção precisa ser feito em memória secundária.
- Não é mais aconselhável usar índices simples, uma vez que:
  - a busca binária 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

- Utiliza-se outras organizações para o índice:
  - *Hashing,* caso a velocidade de acesso seja a prioridade máxima;
  - 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?
- Normalmente, o acesso a registros não se faz por chave primária, e sim por chaves secundárias.
- Solução: cria-se um índice que relaciona uma chave secundária à chave primária (e não diretamente ao registro).

Acesso por múltiplas chaves

- Índices permitem muito mais do que simplesmente melhorar o tempo de busca por um registro.
- Múltiplos índices secundários:
  - permitem manter diferente 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

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

FIGURE 6.6 Secondary key index organized by composer.

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

FIGURE 6.8 Secondary key index organized by recording title.

Busca usando múltiplas chaves

- Uma das aplicações mais importantes das chaves secundárias envolve o 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,OR,NOT) dos resultados .
- Ex: encontre todos os registros de dados com:
  - composer = “BEETHOVEN” AND title = “SYMPHONY NO. 9”.

Exercício

- Faça uma subrotina que crie um arquivo de índice a partir de um arquivo de registros de tamanho fixo.
- O arquivo principal contém os campos Codigo (int - chave), Nome (string 25), Idade (int), Telefone (string 15).

18

Agradecimentos

- Slides da profa. Cristina Ferreira.

19