GBC053 - Gerenciamento de Bancos de Dados

Aula 6
Organização de arquivos para desempenho
Introdução à ordenação interna e busca binária

Humberto Razente
humberto.razente@ufu.br

Busca de registros em arquivos

- Busca por posição
  - busca por posição em geral não atende aos requisitos das aplicações
  - desejado: busca por atributos chave
- Busca sequencial
  - caro para grandes arquivos

Busca baseada em chute

- Considere um arquivo com 1000 registros e a busca pela chave ”Maria José”
  - busca sequencial realiza 1000 comparações
  - caso médio, considera-se 500 comparações

- Se os registros estiverem ordenados, como em uma lista telefônica, pode-se aplicar uma busca binária
  - busca binária: 10 comparações

Busca binária

```
int low = 0, high = numRecords – 1;
while (low <= high) {
    int guess = (high – low) / 2;
    fseek(fd,guess,SEEK_SET);
    fread(key, sizeof(key), 1, fd);
    if (key == search)
            return true;
    else if (key < search)
            high = guess – 1;
    else
            low = guess + 1;
}
return false;
```

Busca binária – disco

```
int low = 0, high = (numRecords – 1) * sizeof(key);
while (low <= high) {
    int guess = (high – low) / 2;
    fseek(fd,guess,SEEK_SET);
    fread(key, sizeof(key), 1, fd);
    if (key == search)
            return true;
    else if (key < search)
            high = guess – sizeof(key);
    else
            low = guess + sizeof(key);
}
return false;
```

Busca binária

- Busca binária tem complexidade $O(\log_2 n)$
- Busca sequencial tem complexidade $O(n)$

- Mas para realizarmos busca binária, registros devem estar ordenados

Ordenação de um arquivo em memória

- Qualquer algoritmo de ordenação interna requer múltiplas passagens pelos dados
  - resultaria em muitos ***seeks*** para se ordenar diretamente no disco
- Se todos os registros couberem na memória primária
  - faz-se a leitura sequencial dos registros
  - aplica-se um algoritmo de ordenação interna

Ordenação interna

- Ordenação por troca
  - BubbleSort, QuickSort
- Ordenação por inserção
  - InsertionSort, ShellSort
- Ordenação por seleção
  - SelectionSort, HeapSort
- Outros métodos
  - MergeSort, BucketSort

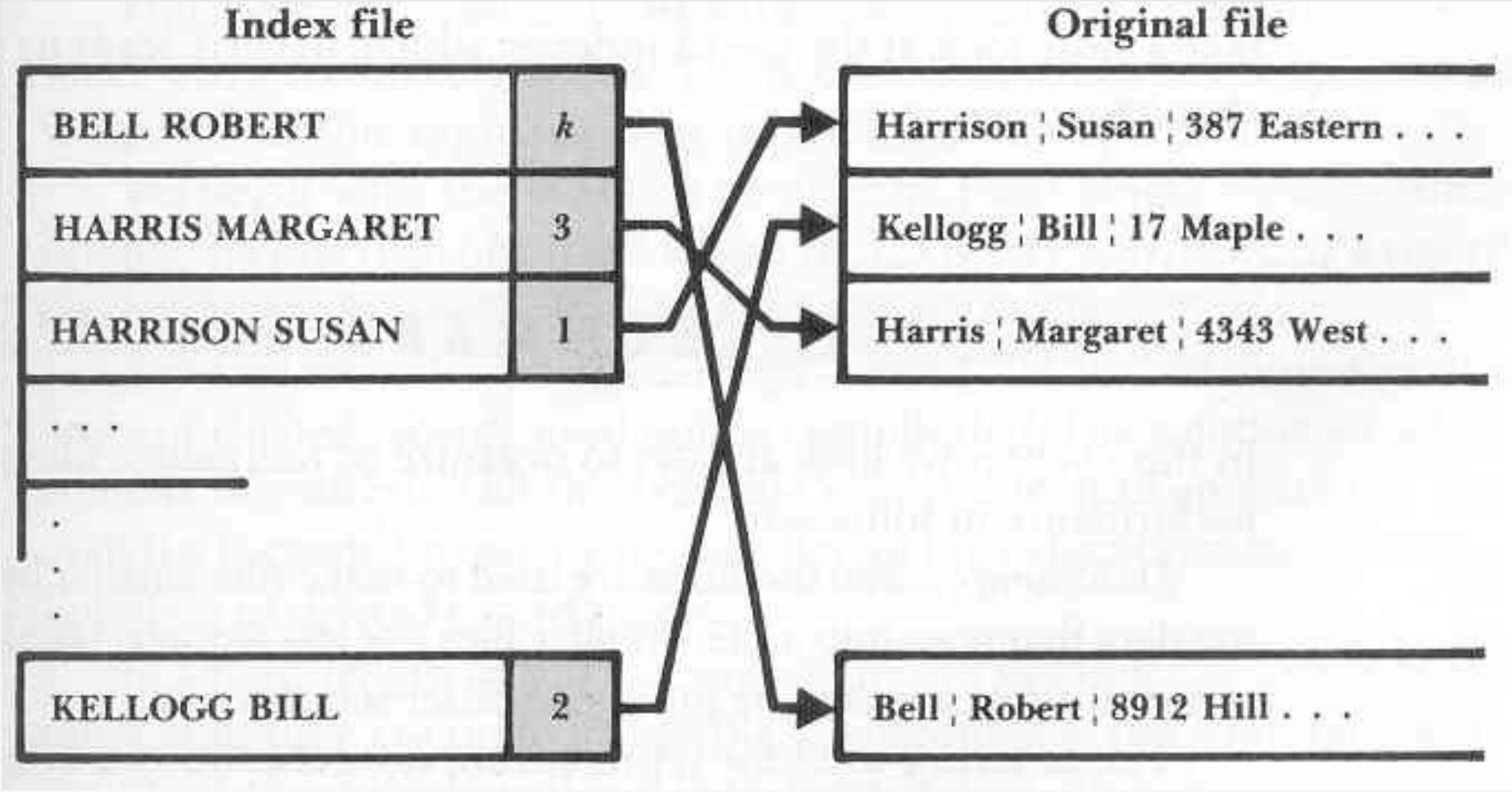
Ordenação de um arquivo em memória

- Limitações da busca binária e ordenação interna
  - Manter um arquivo ordenado é muito caro
    - cada inserção/atualização pode requerer nova ordenação
    - uso depende da frequência das inserções/atualizações

Keysorting

- Baseado na idéia de que para ordenar um arquivo em memória
  - os únicos dados que precisam estar na memória são as chaves
  - não é preciso ler todos os atributos, apenas as chaves
  - ordena-se as chaves e então reorganiza-se os registros no arquivo de acordo com a nova ordem
  - como não lê os registros inteiros para memória, é capaz de ordenar arquivos maiores que a ordenação em memória

Keysorting


Index file
Original file
BELL ROBERT
k
HARRIS MARGARET
3
HARRISON SUSAN
1
KELLOGG BILL
2
Harrison ¦ Susan ¦ 387 Eastern . . .
Kellogg ¦ Bill ¦ 17 Maple . . .
Harris ¦ Margaret ¦ 4343 West . . .
Bell ¦ Robert ¦ 8912 Hill . . .

Keysorting

- Após ter as chaves ordenadas em memória, pode-se reescrever os registros em ordem em um novo arquivo
  - 1 seek para cada registro
  - operação cara
- Estratégia é manter arquivo com registros
  - e escrever em disco o vetor ordenado de chaves
  - resulta em um índice

Leitura complementar

- Capítulo **”Organizing Files for Performance”** do livro
  - Folk et al. ”File Structures: An Object-Oriented Approach with C++”, Editora Pearson, 3ª edição, 1998