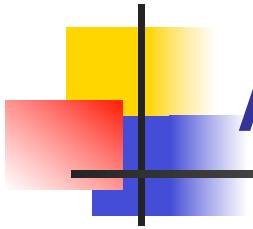


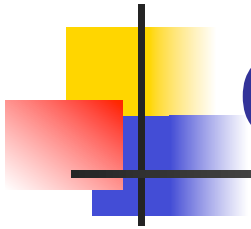
Ordenação e Busca em Arquivos

Thiago A. S. Pardo

Leandro C. Cintra

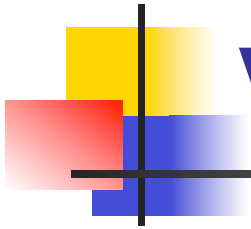
M.C.F. de Oliveira

Cristina D. A. Ciferri



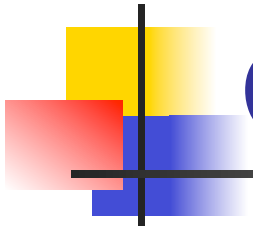
Observações

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



Varredura de Arquivos

- Busca linear
  - recupera cada registro do arquivo, verificando se os valores dos atributos satisfazem à condição de seleção
- Busca binária
  - recupera registros quando a condição de seleção envolve uma comparação de igualdade no atributo que determina a ordenação do arquivo



Custos: Comparações

$$C_{busca\_sequencial} = n$$

- n: número de registros que são comparados
- todos os registros são varridos (pior caso)
- complexidade: O(n)

$$C_{busca\_binária} = \log_2(n) + 1$$

- n: número de registros que são comparados
- complexidade: O(log n)

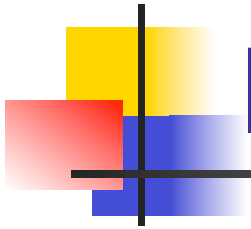
Custos: Acessos a Disco

$$C_{busca\_sequencial} = b$$

- b: número de blocos que contêm os registros
- todos os blocos são varridos

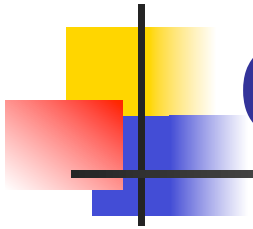
$$C_{busca\_binária} = \log_2(b) + \lceil s/bfr \rceil - 1$$

- log2(b): custo para localizar o primeiro registro
- ⌈s/bfr⌉: blocos ocupados pelos registros que satisfazem à condição de seleção
- 1: custo para recuperar o primeiro registro



Busca Binária

- Dificuldade
  - ordenação dos dados do arquivo
- Alternativa
  - ordenação dos dados em RAM
    - leitura de todo o arquivo em disco
    - *ordenação dos dados em memória primária*
    - escrita de todo o arquivo em disco

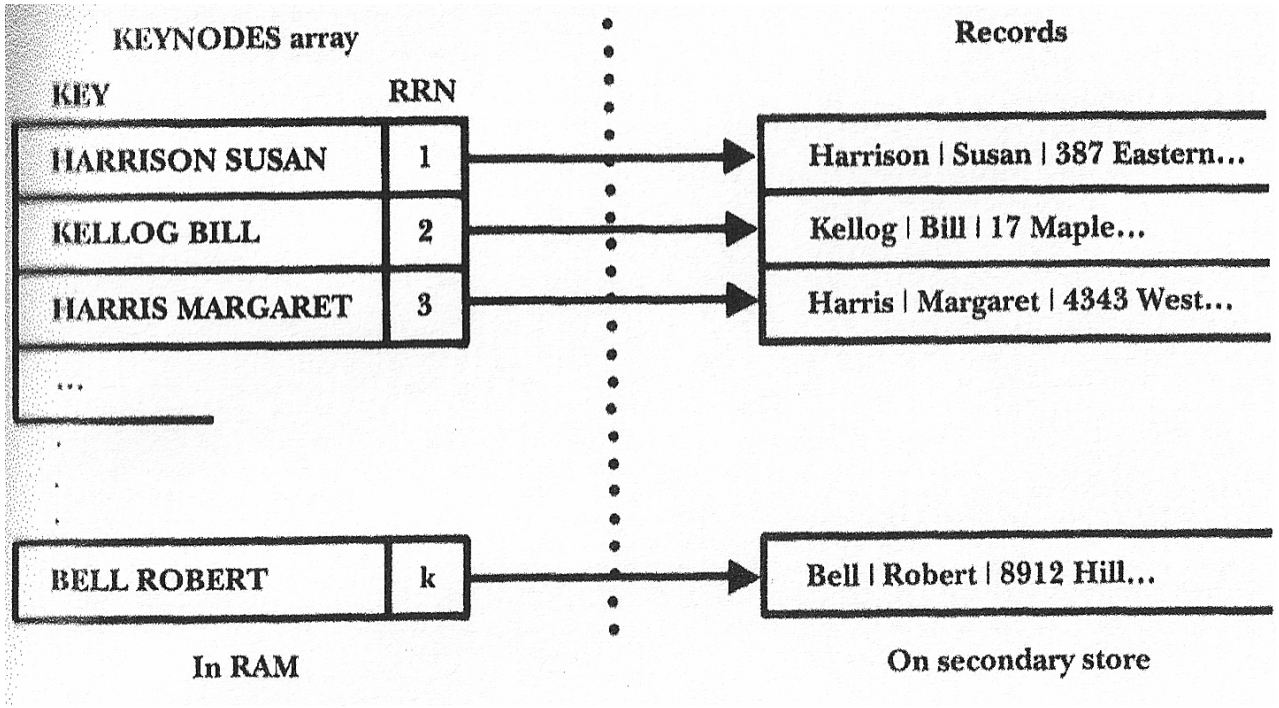


Ordenação dos Dados em RAM

- Como melhorar?
  - carregar somente as chaves para ordenação → *ordenação por chave*
- Método
  - cria-se em RAM um vetor, em que cada posição armazena uma chave e o RRN ou *byte offset* do registro
  - ordena-se o vetor em RAM

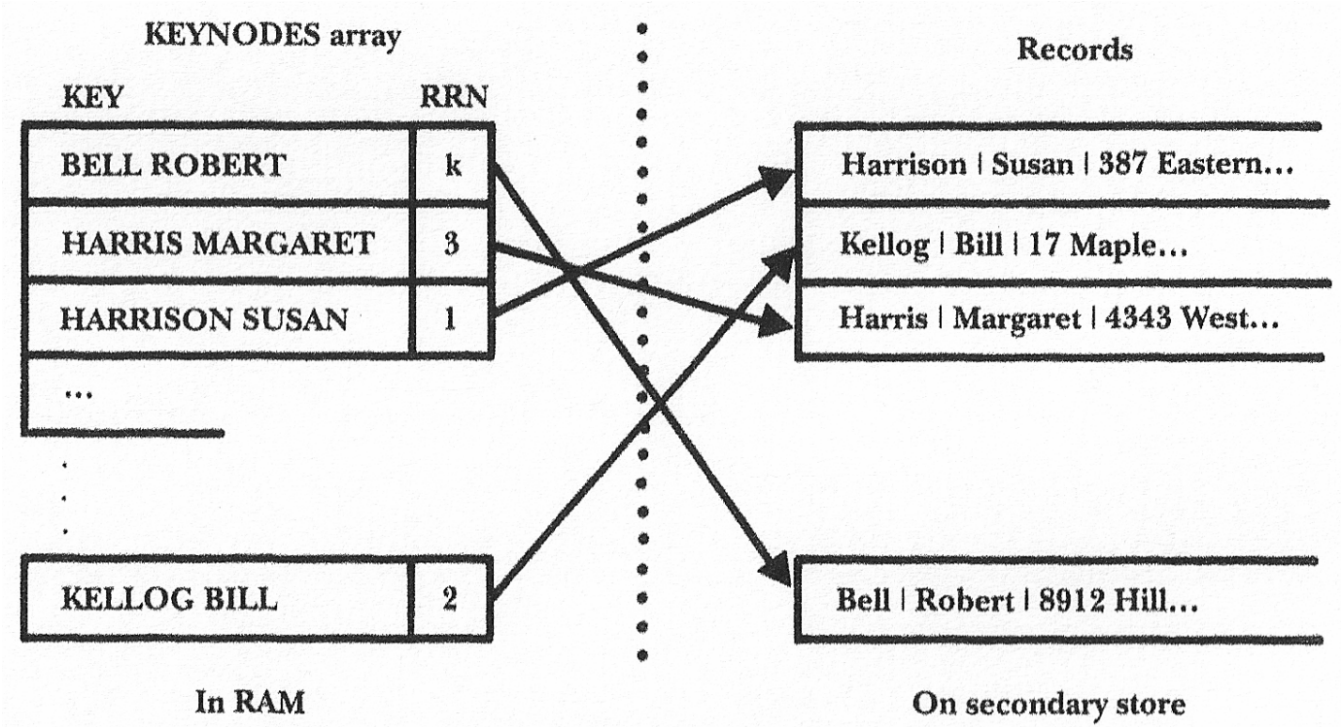
Keysorting

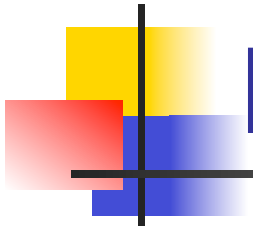
- Carregando dados na RAM



Keysorting

- Ordenando dados em RAM





Keysorting

- Limitações
  - necessário ler os registros uma segunda vez antes da escrita no arquivo de saída ordenado
  - read(arqEntrada), write(arqSaida)
    - os registros não são lidos sequencialmente do arquivo de entrada
    - a escrita não é sequencial devido à alternância entre o arquivo de entrada e o arquivo de saída

Pensando em Índices

- *Por que realizar a tarefa custosa de escrever em disco a versão ordenada do arquivo?*
- Solução melhor ....
  - grava-se a ordenação da chave em um novo arquivo (arquivo de índice)
  - realiza-se busca binária no arquivo de índice, e recupera-se o RRN ou *byte offset*
  - realiza-se acesso direto no arquivo original (arquivo de dados)