Acesso Sequencial e Direto

6897/9895 – Organização e Recuperação de Dados
Profa. Valéria D. Feltrim

UEM – CTC – DIN

Slides preparados com base no Cap. 4 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992, e nos slides disponibilizados pelo Prof. Pedro de Azevedo Berger (DCC/UnB)

Acesso a registros

- Arquivos que utilizam registros pressupõem que:
  - O registro é a unidade de informação
  - Deve ser possível recuperar um registro específico
- Como identificar um registro específico?
  - Pela sua posição no arquivo
    - 6º registro, 2º registro, 10º registro, ...
    - 👎Difícil tratamento na memória do usuário
  - Pelo um valor de chave (um campo)
    - Sobrenome = “Silva”, Nome = “Alan”, ...
    - 👍Mais conveniente!

Chave de registros

- Deve haver regras para mapear os campos do registro em chaves em uma forma padrão (canônica)
- Por exemplo:
  - Se a regra define que as chaves têm letras maiúsculas sem espaços em branco no final, qualquer entrada dada pelo usuário deve ser convertida para a forma canônica antes da inserção e da pesquisa
    - *SILVA → SILVA*
    - *Silva→SILVA*
    - *silva→SILVA*

Chave de registros

- Chave **primária**
  - Identifica **unicamente** cada registro
  - Em geral, não pode ser modificada
    - O ideal é que a chave primária seja artificial (“*dataless*”)
- Chave **secundária**
  - Chave que **pode se repetir** em dois ou mais registros
  - Não há garantia de unicidade
  - Pode ser utilizada para buscas simultâneas de várias chaves (p.e., todos os “Rodrigues” que estudam na “UEM”)
    - Exemplos: nome, cidade, UF, etc.

Voltaremos a falar de chaves primárias/secundárias no Cap. 6

Busca sequencial

- Como buscar por uma chave?
- A busca sequencial consiste em ler um arquivo, registro por registro, procurando por um certo valor de chave
  - Se a chave usada na busca for primária, apenas um registro será selecionado
- Desempenho da busca sequencial
  - O esforço de buscar um registro específico é diretamente proporcional ao número *n* de registros do arquivo
  - Para busca em arquivo, a medida utilizada é o número de leituras físicas
  - Se o acesso a cada registro implica em uma leitura física, então temos:

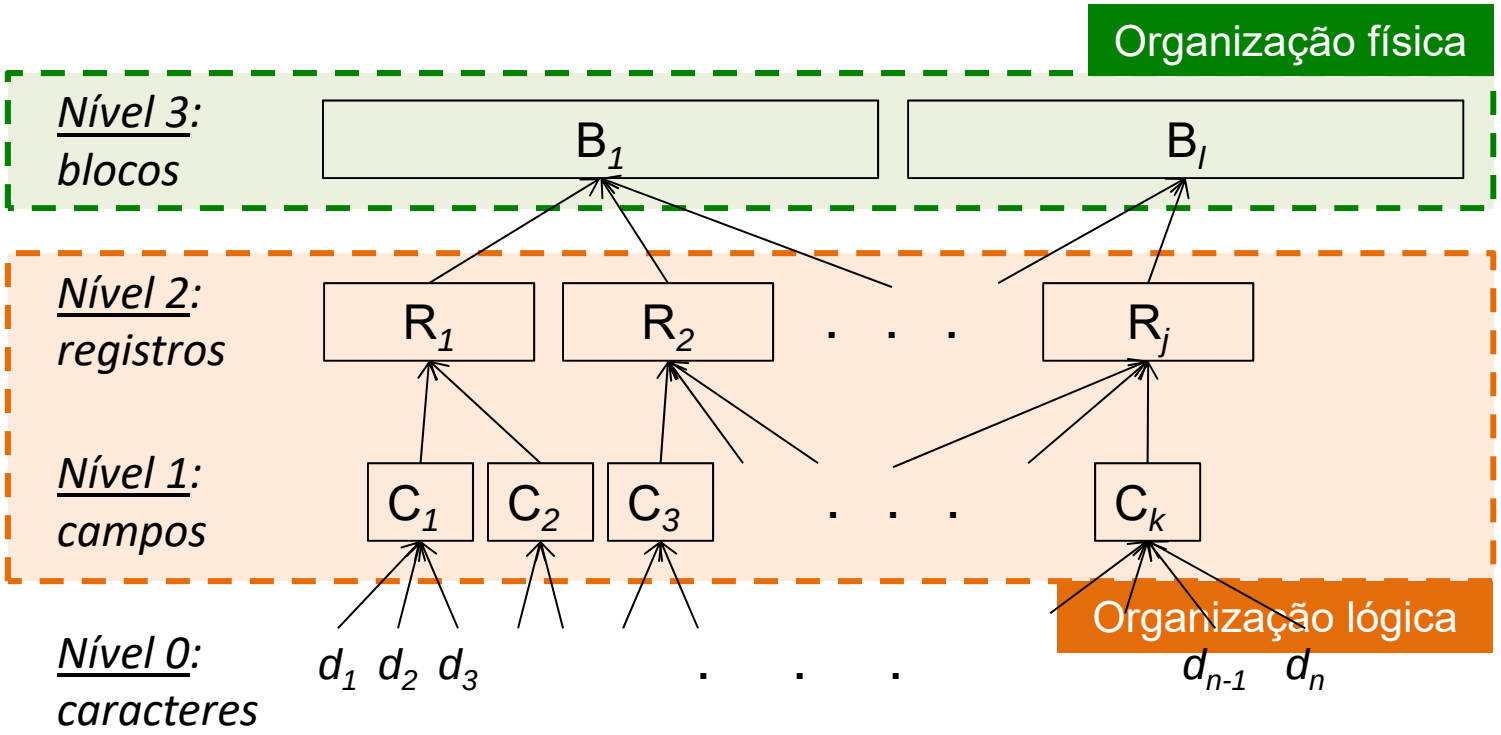
Melhor caso	1 leitura	O(1)
Pior caso	*n* leituras	O(n)
Caso médio	*n*/2 leituras	O(n)

Busca sequencial em blocos

- O desempenho da busca sequencial pode ser melhorado usando-se a recuperação em bloco de registros
  - Sabemos que a parte mais lenta de uma operação de acesso a disco é o ***seeking***, realizado para localizar a porção correta do disco
  - O custo de buscar e ler um registro e depois buscar e ler outro registro é maior do que o custo de buscar (e ler) dois registros sucessivos de uma só vez
  - Podemos melhorar o desempenho da busca sequencial lendo um **bloco com vários registros** por vez para então processar o bloco em RAM

Busca sequencial em blocos

- O agrupamento de registros em blocos introduz um novo nível de organização, porém diferente dos outros
  - Organização lógica (em campos e registros) → manter/dar significado
  - Organização física (em blocos) → melhorar desempenho



Busca sequencial em blocos

- Se cada bloco tem *k* registros (*k* = fator de bloco)*, então:*

Melhor caso	1 leitura	*O(1)*
Pior caso	*n*/*k* leituras	*O(n)*
Caso médio	*n*/2*k* leituras	*O(n)*

- Exemplo:
  - Suponha um arquivo com **4.000 registros**
  - Busca sequencial normal: teremos, em média, **2.000** leituras físicas para encontrar um registro específico
  - Busca sequencial por bloco: se agruparmos 16 registros por bloco, teremos, em média, **125** leituras físicas para encontrar um registro
    - As características do sistema operacional devem ser consideradas na hora de definir o tamanho do bloco, p.e., é recomendado usar um tamanho que seja múltipo do tamanho de cluster usado pelo S.O.

Busca sequencial em blocos

- Considerações
  - A recuperação de blocos de registros economiza tempo
    - Reduz o número de operações de *seek*
    - Mas não muda a ordem do esforço que é de **O(n)**
  - Pode aumentar a quantidade de transferências entre disco e RAM
    - Sempre é transferido o bloco todo, mesmo quando o registro procurado é o primeiro do bloco
    - Mesmo assim, dada a redução no número de *seeks*, compensa

Acesso direto

- O arquivo passa a ser visto como uma sequência de registros
- Na busca pelo k-ésimo registro no arquivo:
  - O acesso é feito diretamente ao k-ésimo registro
  - A leitura não passa pelos k – 1 registros anteriores
  - O custo da busca é constante → *O(1)*
    - Independe do tamanho do arquivo

Acesso direto

- Para o acesso direto, precisamos saber o endereço do registro a ser recuperado
  - Se os registros têm tamanho variável, precisamos conhecer seu ***byte-offset*** → endereço do primeiro byte do registro
  - Se os registros têm tamanho fixo, precisamos apenas do Número Relativo do Registro (**RRN**)
    - Podemos calcular o *byte-offset* usando o RRN
  - Sem um índice, o *byte-offset*/RRN não servem de muita coisa
    - Veremos como criar e manter arquivos de índices em aulas seguintes

Acesso direto

- Registro de tamanho fixo
  - Nesse caso, o *byte-offset* é calculado a partir do RRN
  - *Byte-offset* = RRN * tamanho do registro
    - P.e., se queremos o recuperar o registro de RRN = 546 e o tamanho dos registros é 128b, então o *byte-offset* do registrado procurado é 546 x 128 = 69.888 → i.e., o 1º byte do registro em questão está no byte 69.888 do arquivo
    - A quantidade de bytes do cabeçalho (se houver) deve ser considerada no cálculo do *byte-offset*
    - Podemos fazer um *fseek()* diretamente para esse byte

Nem toda LP trata o arquivo como uma sequência de bytes → Pascal, por ex. Nesse caso, o seek é feito por "elemento" no arquivo e não por um *offset.*

Registro de cabeçalho (*Header*)

- Podemos manter informações sobre o arquivo gravadas no 1º registro
  - **Registro de cabeçalho (*header*)**
- Exemplo de informações típicas de cabeçalho:
  - Número de registros, tamanho dos registros (se o tamanho for fixo), nº de campos nos registros, data e hora de criação e/ou última atualização, etc.
- Os dados do cabeçalho auxiliam o uso do arquivo por parte da aplicação
- O registro de cabeçalho difere dos demais registros do arquivo
  - O que pode ser um problema em algumas linguagens de programação

Acesso a arquivos e organização de arquivos

- Até aqui, nós vimos:
  - Registros de tamanho fixo
  - Registros de tamanho variável

  Organização de arquivos
  - Acesso sequencial
  - Acesso direto

  Acesso a arquivos
- Como **acesso** e **organização** interagem?

Acesso a arquivos e organização de arquivos

- A escolha de uma organização particular para um arquivo depende de várias coisas, entre elas:
  - Facilidades oferecidas pela linguagem de programação
  - **Como se quer acessar o arquivo**
    - Registros de tamanho fixo
      - Acesso tanto sequencial quanto direto é feito facilmente
    - Registros de tamanho variável
      - Acesso sequencial fácil
      - Acesso direto?
        - » Por RRN não é possível
        - » Para o acesso direto é preciso manter uma lista relacionando cada RRN com o respectivo *byte-offset*