

Acesso Sequencial e Direto

Organização e Recuperação de Dados


Profa. Valéria

UEM – CTC – DIN

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?
 - Pela sua **posição** no arquivo
 - P.e., 6º registro, 2º registro, 10º registro, ...
 - 🙅 Sem muita utilidade prática
 - Por um valor de **chave** (um campo)
 - P.e., Sobrenome = “Silva”, Nome = “Alan”, ...
 - 👍 Mais conveniente

25Silva | Alan | (23)3666-1111 | 23Flores | Andre | 3300-9874 | 30Santos | Cristina | (42)3568-4789 | ...



Chaves 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

Chaves 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 moram em “Maringá”)
 - Exemplos: nome, cidade, UF, etc.

Voltaremos a falar de chaves primárias/secundárias quando falarmos sobre Índices

Busca sequencial

- Como buscar um registro por 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
 - 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:

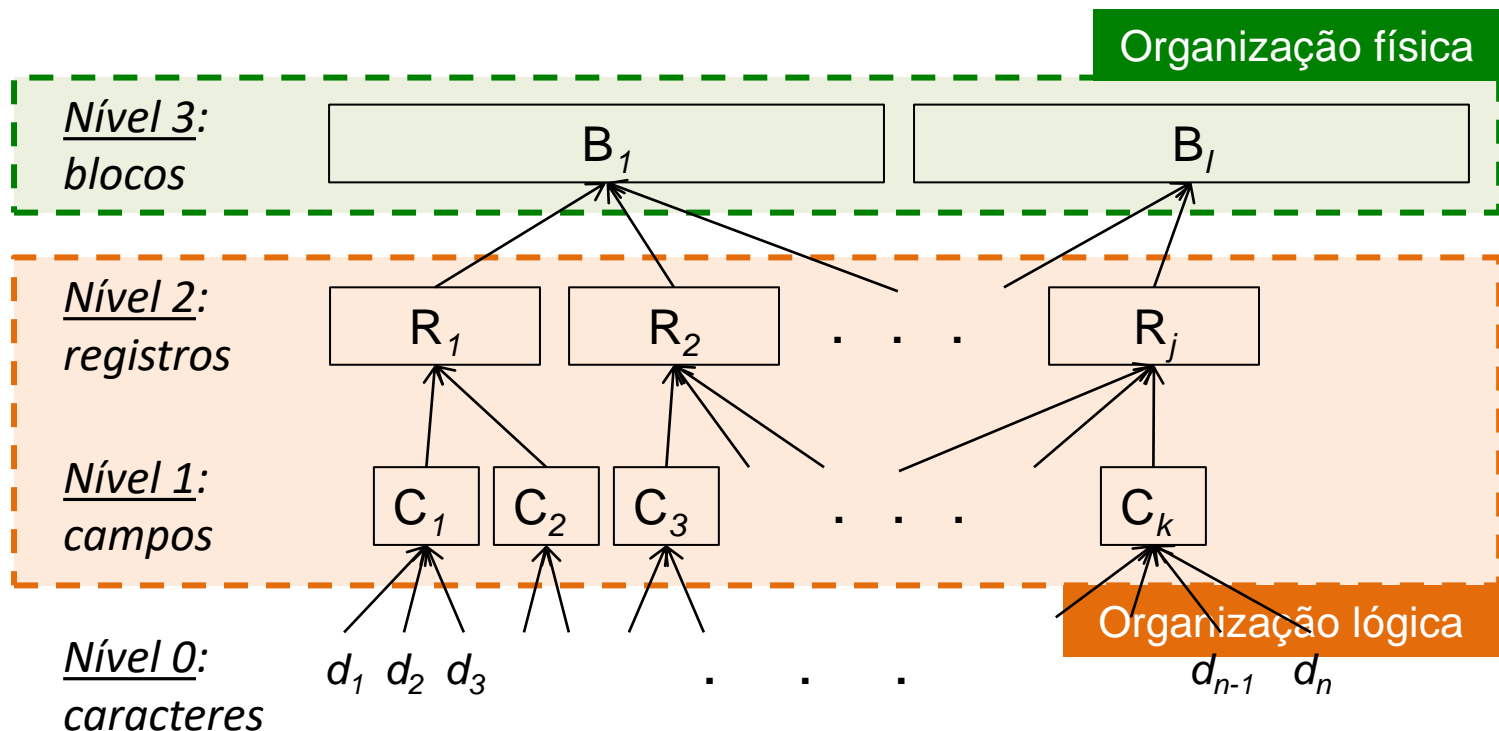
Melhor caso	1 leitura	$O(1)$
Pior caso	n leituras	$O(n)$
Caso médio	$n/2$ leituras	$O(n)$

Busca sequencial em blocos

- O desempenho da busca sequencial será melhor se a leitura do arquivo é feita em bloco de registros
 - Sabemos que a parte mais lenta do acesso ao disco é o **seek**, realizado para localizar o cilindro correto 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 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 ao arquivo, porém diferente dos vistos até aqui
 - Campos e registros → manter/dar significado → Organização **lógica**
 - Blocos → melhorar desempenho → Organização **física**



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	$n/2k$ leituras	$O(n)$

- Exemplo:

- Suponha um arquivo com **4.000 registros**
- Busca sequencial por registro: 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 múltiplo do tamanho de *cluster* usado pelo S.O.

Busca sequencial em blocos

■ Considerações:

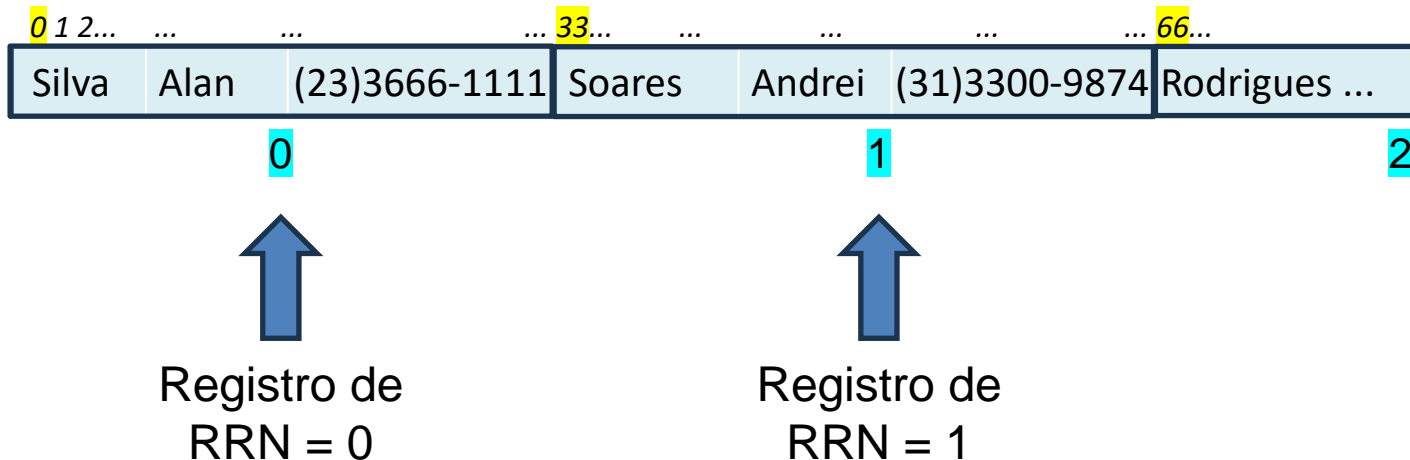
- A recuperação de blocos de registros economiza tempo
 - Reduz o número de operações de *seek*
 - Mas não muda a ordem do esforço, que é de $O(n)$
- Pode aumentar a quantidade de transferências entre disco e RAM
 - Sempre é transferido o bloco todo, mesmo quando o registro procurado for o primeiro do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

Acesso direto

- No acesso direto, um registro é acessado por meio do seu endereço
- Idealizado para registros de fixo
- Na busca pelo k-ésimo registro no arquivo:
 - O acesso é feito diretamente ao registro k
 - A leitura não passa pelos $k - 1$ registros anteriores
 - O custo da busca é constante $\rightarrow O(1)$
 - Independe do tamanho do arquivo

Acesso direto

- Quando os registros têm tamanho fixo, podemos usar a ordem em estão no arquivo como um endereço
 - Visão lógica do arquivo como uma sequência de registros em vez de uma sequência de bytes
- Chamamos esse “endereço” de *Relative Record Number* (Número Relativo do Registro) → **RRN**



Acesso direto

■ Como fazer acesso por RRN?

- É possível calcular o *byte-offset* do registro a partir do RRN
- *byte-offset* = $RRN * \text{tamanho do registro}$
 - 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 $546 * 128 = 69.888 \rightarrow$ i.e., o 1º byte do registro em questão está no deslocamento 69.888 a partir do início do arquivo
 - A quantidade de bytes do cabeçalho do arquivo (se houver) deve ser considerada no cálculo
 - *byte-offset* = $RRN * \text{tamanho_registro} + \text{tamanho_cabeçalho}$
- Conhecendo o *byte-offset*, podemos fazer um *seek()* diretamente para esse byte

Acesso direto

- Se os registros têm tamanho variável, precisaremos conhecer os *byte-offsets* de antemão para poder fazer acesso direto
 - Registros de tamanho variável → precisamos conhecer seu *byte-offset* → endereço do primeiro byte do registro



- Registros de tamanho fixo → conhecendo o **RRN**, podemos calcular o *byte-offset*



Sem um índice, o *byte-offset* ou o RRN não têm muita utilidade prática

Registro de cabeçalho (*Header*)

- Podemos manter informações sobre o arquivo gravadas no 1º registro
 - Registro de cabeçalho (*header*)
- Os dados do cabeçalho auxiliam o uso do arquivo por parte da aplicação
 - P.e.: número de registros, tamanho dos registros (se o tamanho for fixo), nº de campos nos registros, data/hora da última atualização, etc.



- 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 | Organização |
| — Registros de tamanho variável | |
| — Acesso sequencial | Modo de acesso |
| — Acesso direto | |

■ 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 pretende 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*