



# Conceitos de Estrutura de Arquivos

---

## Estruturas de Dados II

Slides cedidos pelo Prof. Gustavo Batista (ICMC-USP)

[<http://www.icmc.usp.br/~gbatista/>]

Discussões adicionais (Profa. Cristina Ciferri) em

<http://wiki.icmc.usp.br/images/3/33/SCC0215012018camposRegistros.pdf>



# **ORGANIZAÇÃO DE ARQUIVOS**



# Organização de Arquivos

---

- Informações em arquivos são, em geral, organizadas logicamente em campos e registros.
- Entretanto, campos e registros são conceitos lógicos, que não necessariamente correspondem a uma organização física.
- Dependendo de como a informação é mantida no arquivo, campos lógicos sequer podem ser recuperados...



# Sequência de bytes (*stream*)

---

- Exemplo:
  - Suponha que desejamos armazenar em um arquivo os nomes e endereços de várias pessoas
  - Suponha que decidimos representar os dados como uma sequência de bytes (sem delimitadores, contadores, etc.)

AmesJohn123MapleStillwaterOK74075MasonAlan90EastgateAdaOK74820



# Sequência de bytes (*stream*)

---

- Exemplo



# Sequência de bytes (*stream*)

---

- A organização adotada para o arquivo é como uma sequência de caracteres.
- Os dados são lidos do teclado e escritos num arquivo.
- Uma vez escritas as informações, não existe como recuperar porções individuais (nome ou endereço).



# Sequência de bytes (*stream*)

---

- Desta forma, perde-se a integridade das unidades fundamentais de organização dos dados:
  - Os dados são agregados de caracteres com significado próprio;
  - Tais agregados são chamados **campos** (*fields*).



# Organização em campos

---

- **Campo:**

- **menor unidade lógica de informação** em um arquivo;
- uma noção lógica (ferramenta conceitual), não corresponde necessariamente a um conceito físico.

- Existem várias maneiras de organizar um arquivo mantendo a identidade dos campos:
  - A organização anterior não proporciona isso...





# Métodos para organização em campos

---

- Fixar o tamanho do campo (comprimento fixo).
- Indicador de comprimento.
- Delimitadores.
- Uso de *tags* (keyword = value).

# Métodos para organização em campos

Ames	John	123 Maple	Stillwater	OK74075377-1808
Mason	Alan	90 Eastgate	Ada	OK74820

(a) Field lengths fixed. Place blanks in the spaces where the phone number would go.

Ames John 123 Maple Stillwater OK 74075 377-1808
Mason Alan 90 Eastgate Ada OK 74820

(b) Delimiters are used to indicate the end of a field. Place the delimiter for the “empty” field immediately after the delimiter for the previous field.

Ames ... Stillwater OK 74075 377-1808 #Mason ... 90Eastgate Ada OK 74820 #...
---

(c) Place the field for business phone at the end of the record. If the end-of-record mark is encountered, assume that the field is missing.

SURNAME=Ames FIRSTNAME=John STREET=123 Maple ... ZIP=74075 PHONE=377-1808 #...
--

(d) Use a keyword to identify each field. If the keyword is missing, the corresponding field is assumed to be missing.

**FIGURE 4.3** Four methods for organizing fields within records to account for possible missing fields. In the examples, the second record is missing the phone number.



# Campos com tamanho fixo

---

- Cada campo ocupa no arquivo um tamanho fixo, pré-determinado (por ex., 4 bytes).
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo.
- Se trata de uma organização simples para gravar e ler dados.



# Campos com tamanho fixo

---

- Vantagem

- facilidade na pesquisa.
- razoável se o comprimento dos campos é realmente fixo, ou apresenta pouca variação.

- Desvantagem

- o espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício).
- solução inapropriada quando se tem uma grande quantidade de dados com tamanho variável.
- dados podem precisar ser truncados.



# Campos com indicador de comprimento

---

- O tamanho de cada campo é armazenado imediatamente antes do dado.
- Se o tamanho do campo é inferior a 256 bytes, o espaço necessário para armazenar a informação de comprimento é um único byte.



# Campos com indicador de comprimento

---

- Vantagem

- economia de espaço de armazenamento, mesmo com a necessidade de se gastar alguns bytes adicionais para guardar o tamanho dos campos.
- dados não precisam ser truncados.

- Desvantagem

- dificuldade na pesquisa.



# Campos separados por delimitadores

---

- Caractere(s) especial(ais) (que não faz(em) parte do dado) é (são) escolhido(s) para ser(em) inserido(s) ao final de cada campo.
- Ex.: para o campo *nome* pode-se utilizar |, tab, #, etc...
- Espaços em branco não serviriam...

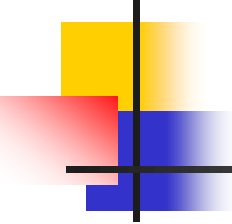


# Campos separados por delimitadores

---

- Vantagem
  - economia de espaço de armazenamento.
- Desvantagens
  - dificuldade na pesquisa.
  - necessidade de escolha de um delimitador que não pertence ao domínio dos dados.





# Uso de uma *tag* do tipo "keyword=value"

---

- Vantagem: o campo fornece informação (semântica) sobre si próprio.
- Fica mais fácil identificar o conteúdo do arquivo.
- Fica mais fácil identificar campos faltantes.
- Desvantagem: as *keywords* podem ocupar uma porção significativa do arquivo.



# Organização em campos

---

- Exemplos



# Organização em registros

---

- **Registro:** um conjunto de campos agrupado.
- Arquivo representado em um nível de organização mais alto.
- Assim como o conceito de campo, um registro é uma ferramenta conceitual, que não necessariamente existe no sentido físico.
- É um outro nível de organização imposto aos dados com o objetivo de preservar o significado.



# Métodos para organização em registros

---

- Tamanho fixo.
- Número fixo de campos.
- Indicador de tamanho.
- Uso de índice.
- Utilizar delimitadores.

# Registros tamanho fixo

Ames	John	123 Maple	Stillwater	OK74075
Mason	Alan	90 Eastgate	Ada	OK74820

(a)

Ames	John	123 Maple	Stillwater	OK	74075	← Unused space →
Mason	Alan	90 Eastgate	Ada	OK	74820	← Unused space →

(b)

Ames	John	123 Maple	Stillwater	OK	74075	Mason	Alan	90 Eastgate	Ada	OK	...
------	------	-----------	------------	----	-------	-------	------	-------------	-----	----	-----

(c)

**FIGURE 4.5** Three ways of making the lengths of records constant and predictable. (a) Counting bytes: fixed-length records with fixed-length fields. (b) Counting bytes: fixed-length records with variable-length fields. (c) Counting fields: six fields per record.



# Registros de tamanho fixo

---

- Analogamente ao conceito de campos de tamanho fixo, assume que todos os registros têm o mesmo número de bytes.
- Um dos métodos mais comuns de organização de arquivos.
- Pode-se ter registros de tamanho fixo com campos de tamanho variável.
  - Neste caso, usa-se delimitadores.

```
struct {  
    char last[10];  
    char first[10];  
    char city[15];  
    char state[2];  
    char zip[9];  
} set_of_fields;
```



# Registros com número fixo de campos

---

- Ao invés de especificar que cada registro contém um número fixo de bytes, podemos especificar um número fixo de campos.
- O tamanho do registro, em bytes, é variável.
- Neste caso, os campos seriam separados por delimitadores.



# Indicador de tamanho para registros

---

- O indicador que precede o registro fornece o seu tamanho total, em bytes.
- Os campos são separados internamente por delimitadores...
- Boa solução para registros de tamanho variável.



# Indicador de tamanho para registros

**FIGURE 4.8** Records preceded by record-length fields in character form.

```
40 Ames John;123 Maple;Stillwater;OK;74075 36 Mason Alan;90  
Eastgate;Ada;OK;74820;
```

# Índice

**FIGURE 4.6** Record structures for variable-length records. (a) Beginning each record with a length indicator. (b) Using an index file to keep track of record addresses. (c) Placing the delimiter '#' at the end of each record.

40Ames;John;123 Maple;Stillwater;OK;74075;36Mason;Alan;90 Eastgate . . .

(a)

Data file:

Ames;John;123 Maple;Stillwater;OK;74075;Mason;Alan . . .

Index file:

00 40 . . .

(b)

Ames;John;123 Maple;Stillwater;OK;74075;#Mason;Alan;90 Eastgate;Ada;OK . . .

(c)



# Utilizar um índice

---

- Um índice externo poderia indicar o deslocamento de cada registro relativo ao início do arquivo.
- Pode ser utilizado também para calcular o tamanho dos registros.
- Os campos seriam separados por delimitadores...



# Utilizar delimitadores

---

- Separar os registros com delimitadores análogos aos de fim de campo.
- O delimitador de campos é mantido, sendo que o método combina os dois delimitadores.
- Note que delimitar fim de campo é diferente de delimitar fim de registro.



# Organização em registros

---

- Não existe um método de organização em registros apropriado para todas as situações. A escolha depende da natureza dos dados e de como eles serão utilizados.



# Como Escolher a Organização de um Arquivo

---

- Considerações a respeito da organização do arquivo (análise)
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?
  - como separar os registros?
  - como identificar o espaço utilizado e o "lixo"?
- Existem muitas respostas para estas questões
  - a escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo



# Análise: Tamanho dos Registros: Exemplo com Decisão Fácil

---

- Arquivo de controle de vendas
  - número do comprador: 8 bytes
  - data no formato DDMMAA: 6 bytes
  - número do item: 4 bytes
  - quantidade vendida: 4 bytes
  - valor da venda: 8 bytes
- Campos de tamanho fixo: 30 bytes



# Análise: Tamanho dos Registros: Exemplo com Decisão Fácil

---

- Registros de 30 bytes
  - cluster de 4KB (4.096 bytes)
  - número de registros por cluster: 136,53
- Registros de 32 bytes (escolhido/mais adequado)
  - cluster de 4KB (4.096 bytes)
  - número de registros por cluster: 128





# Análise: Tamanho dos Registros: Exemplo com Decisão Difícil

---

- Campos com tamanho muito variável
  - nome
  - endereço
- Abordagens simplistas para o tamanho do registro
  - Todos os campos de tamanho fixo
    - [a] soma do tamanho máximo de cada campo
    - [b] soma do tamanho mínimo de cada campo
    - vantagem: simplicidade na pesquisa
    - desvantagem: problemas dos tamanhos máximo e mínimo



# Análise: Tamanho dos Registros: Exemplo com Decisão Difícil

---

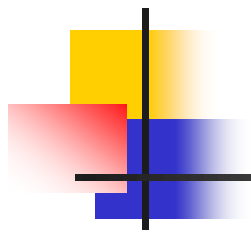
- Todos os campos de tamanho variável
  - Vantagem
    - pode-se aplicar o efeito do tamanho médio
      - nomes mais longos em geral não aparecem no mesmo registros que os endereços mais longos
  - Desvantagem
    - mais dificuldade na pesquisa por campos que não sofrem muita variação



# Análise: Tamanho dos Registros: Exemplo com Decisão Difícil

---

- Decisão interessante
  - campos cujos dados possuem pouca variabilidade: campos de tamanho fixo
  - campos cujos dados possuem grande variabilidade: campos de tamanho variável
- Usa-se uma organização em campos adequada às características dos dados armazenados no arquivo



**ACESSO**



# Acesso a registros

---

- Considerando que os arquivos estão organizados por registros, devemos pensar em como recuperar um registro específico (ao invés de ler todo o arquivo).
- É interessante ter uma chave (*key*) associada a cada registro, chave essa que, baseada no conteúdo do registro, serviria para identificá-lo de maneira única.



# Acesso a registros

---

- Assim como os conceitos de campos e registros, o conceito de chave também é uma ferramenta conceitual importante.
- Por exemplo, no arquivo com nomes e endereços de clientes seria melhor pesquisar em termos dos registros de "Ames" ou de "Mason", em vez de procurar o primeiro ou segundo registros.



# Chaves Primária e Secundária

---

- Uma **chave primária** é, por definição, a chave utilizada para identificar unicamente um registro
  - Ex.: RA, CPF, RG, ...
  - Sobrenome, por outro lado, não é uma boa escolha para chave primária...
- Uma **chave secundária**, tipicamente, não identifica unicamente um registro, e pode ser utilizada para buscas simultâneas por várias chaves (todos os "**Silvas**" que moram em **São Paulo**, por exemplo).



# Escolha da Chave

---

- A chave primária deve ser "*dataless*", isto é, não deve ter um significado associado, e não deve **mudar nunca** (outra razão para não ter significado).
- Uma mudança de significado pode implicar na mudança do valor da chave, o que invalidaria referências já existentes baseadas na chave antiga.





# Forma canônica da chave

---

- “Ana”, “ANA”, ou “ana” devem levar ao mesmo registro.
- **Formas canônicas** para as chaves: uma única representação da chave que conforme com uma regra.
- Ex.: A regra pode ser, “todos os caracteres maiúsculos”:
  - Nesse caso a forma canônica da chave será ANA.



# Busca

---

- Dado um arquivo organizado em registros e uma chave, deseja-se encontrar o registro associado a chave.
- Três possibilidades principais:
  - Busca sequencial;
  - Busca Binária;
  - Acesso Direto.



# Busca Sequencial

---

- Busca pelo registro que tem uma determinada chave em um arquivo:
  - Lê o arquivo, registro a registro, em busca de um registro contendo um certo valor de chave.



# Desempenho da Busca Sequencial

---

- Na busca em RAM normalmente considera-se como medida o número de comparações efetuadas para obter o resultado da pesquisa.
- No contexto de pesquisa em arquivos, o acesso a disco é a operação mais cara e, portanto, o número de acessos a disco efetuados é utilizado como medida do trabalho.
- **Mecanismo de avaliação do custo associado ao método:** contagem do número de chamadas à função de baixo nível READ().



# Desempenho da Busca Sequencial

---

- Exemplo: Supondo que cada chamada a READ lê 1 registro e requer um *seek* (i.e., que todas as chamadas a READ têm o mesmo custo):
  - Uma busca sequencial por "ANA" em 2.000 registros requer, em média, 1.000 leituras (1 se for o primeiro registro – melhor caso; 2.000 se for o último – pior caso; e 1.000, em média).
  - Em geral, o trabalho necessário para buscar um registro em um arquivo de tamanho  $n$  utilizando busca sequencial é  $O(n)$ .



# Vantagens da Busca Sequencial

---

- Vantagem
  - fácil de programar
  - requer estruturas de arquivos simples
  - pode ser aplicada a qualquer arquivo
- Desvantagem
  - pode ser ineficiente



# Busca sequencial é razoável

---

- Na busca por uma cadeia em um arquivo ASCII (comandos unix: grep, cat, wc).
- Em arquivos com poucos registros (da ordem de 10).
- Em arquivos pouco pesquisados.
- Na busca por registros com chaves que se repetem (secundária), para a qual se espera muitos registros (muitas ocorrências).



# Busca Binária

---

- Envolve comparar a chave procurada  $p$  com a chave do registro mediano  $r$ :
  - Se  $p = r$ , pára;
  - Se  $p < r$ , deve-se procurar  $p$  somente na “metade inferior” do arquivo;
  - Se  $p > r$ , deve-se procurar  $p$  somente na “metade superior” do arquivo.





# Busca Binária

---

- Busca binária possui diversas restrições:
  - Requer que o arquivo esteja ordenado;
  - Encontra a chave com poucas comparações ( $O(\log n)$ ), mas requer muitas operações de *seek*;
  - Requer um arquivo com registros de tamanho fixo ou uma estrutura auxiliar de índice.



# Busca Binária

---

- Por outro lado:
  - Se a estrutura auxiliar de índice estiver disponível e for pequena o suficiente para caber em memória principal, a busca binária pode ser feita na estrutura de índice e o arquivo ser acessado uma única vez;
  - Mais sobre isso na aula de índices...



# Acesso Direto

---

- A alternativa mais radical ao acesso sequencial é o **acesso direto**.
- O acesso direto implica em realizar um *seeking* direto para o início do registro desejado (ou do setor que o contém) e ler o registro imediatamente.
- É  $O(1)$ , pois um único acesso traz o registro, independentemente do tamanho do arquivo.



# Posição do início do registro

---

- Para localizar a posição exata do início do registro no arquivo, pode-se utilizar um arquivo índice separado.
- Ou pode-se ter um **RRN (Relative Record Number)** que fornece a posição relativa do registro dentro do arquivo (**byte offset**).



# Posição de um registro com RRN

---

- Para utilizar o RRN, é necessário trabalhar com registros de tamanho fixo:
  - Nesse caso, a posição de início do registro é calculada facilmente a partir do seu RRN:  
*Byte offset = RRN \* Tamanho do registro;*
  - Por exemplo, se queremos a posição do registro com RRN 546, e o tamanho de cada registro é 128, *o Byte offset é 546 x 128 = 69.888.*



# Acesso a arquivos X Organização de arquivos

---

- **Organização de Arquivos:**
  - Registros de tamanho fixo;
  - Registros de tamanho variável.
- **Acesso a arquivos:**
  - Busca sequencial;
  - Busca binária;
  - Acesso direto.



# Acesso a arquivos X Organização de arquivos

---

- Utilizar o arquivo implica em acessá-lo. No exemplo visto, a escolha por acesso direto implica na organização de registros de tamanho fixo. Mas nada impede que um arquivo deste tipo seja acessado sequencialmente.
- Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável.
  - Como acessar esses registros diretamente?

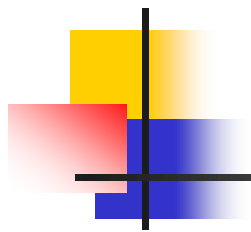


# Acesso a arquivos X Organização de arquivos

---

- Existem também as limitações da linguagem:
  - C permite acesso a qualquer byte, e o programador pode implementar acesso direto a registros de tamanho variável;
  - Pascal exige que o arquivo tenha todos os elementos do mesmo tipo e tamanho, de maneira que acesso direto a registros de tamanho variável é difícil de ser implementado.
- Conclusão:
  - O tipo de acesso e a organização de arquivos são dependentes entre si.





**DEMAIS CONCEITOS**



# Registro Cabeçalho (Header Record)

---

- Em geral, é interessante manter algumas informações sobre o arquivo para uso futuro. Essas informações podem ser mantidas em um *header* no início do arquivo.
- Algumas informações típicas são:
  - número de registros;
  - tamanho de cada registro;
  - campos de cada registro;
  - datas de criação e atualização;
  - A existência de um registro *header* torna um arquivo um objeto auto-descrito. O software pode acessar arquivos de forma mais flexível.