Algoritmos e Estruturas de Dados II - SCC-203

Arquivos: Organização

Gustavo Batista

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...

Seqüência 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 seqüência de bytes (sem delimitadores, contadores, etc.)

```
AmesJohn123MapleStillwaterOK74075MasonAlan90EastgateAdaOK74820
```

Seqüência de bytes (*stream*)

- Uma vez escritas as informações, não existe como recuperar porções individuais (nome ou endereço.)
- 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*.)

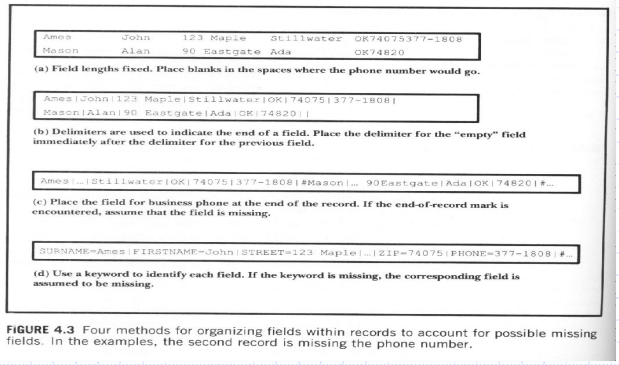
Organização em campos

- **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

- Comprimento fixo.
- Indicador de comprimento.
- Delimitadores.
- Uso de *tags.*

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

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

Campos com tamanho fixo

- 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.
- Razoável se o comprimento dos campos é realmente fixo, ou apresenta pouca variação.

Campos com indicador de comprimento

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

Campos separados por delimitadores

- Caractere(s) especial(ais) (que não fazem parte do dado) 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...

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 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

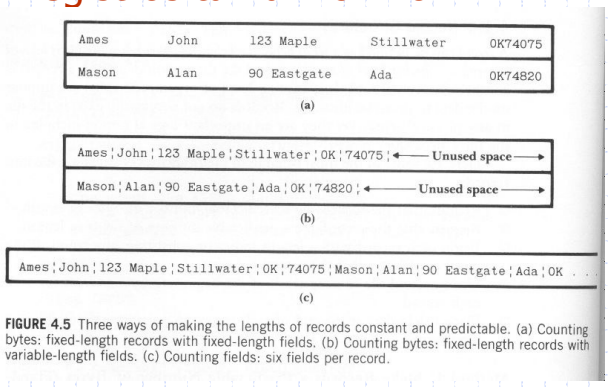


FIGURE 4.5 Three ways of making the lengths of records constant and predictable. (a) Counting bytes: fixed-length records with fixed-length fields. (b) Counting bytes: fixed-length records with variable-length fields. (c) Counting fields: six fields per record.

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.

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

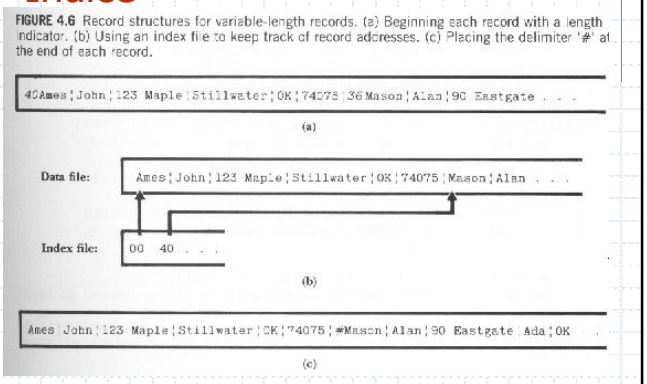


FIGURE 4.6 Record structures for variable-length records. (a) Beginning each record with a length indicator. (b) Using an index file to keep track of record addresses. (c) Placing the delimiter '#' at the end of each record.

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.

Chaves Primária e Secundária

- Uma **chave primária** é, por definição, a chave utilizada para identificar unicamente um registro
  - Ex. nro. USP, 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 seqüencial;
  - Busca Binária;
  - Acesso Direto.

Busca Seqüencial

- 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 Seqüencial

- Na busca em RAM, normalmente consideramos 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 Seqüencial

- 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 seqüencial 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 seqüencial é *O(n).*

Vantagens da Busca Seqüencial

- Fácil de programar
- Requer estruturas de arquivos simples

Busca seqüencial é razoável

- Na busca por uma cadeia em um arquivo ASCII .
- Em arquivos com poucos registros (da ordem de 10).
- Em arquivos pouco pesquisados.
- Na busca por registros com chaves que se repetem, 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 seqüencial 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*) (ou byte offset)** que fornece a posição relativa do registro dentro do arquivo.

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 seqüencial;
  - Busca binária;
  - Acesso direto.

Acesso a arquivos X Organização de arquivos

- Considerações a respeito da organização do arquivo
  - 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

Acesso a arquivos X Organização de arquivos

- Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável.
- Como acessar esses registros diretamente?
- 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.

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.

Arquivos auto-descritivos e cabeçalhos

- É possível colocar informações elaboradas nos cabeçalhos dos arquivos, de modo que o arquivo fique auto-descritivo (Metadados).
- Exemplo de informações no cabeçalho:
  - nome de cada campo;
  - largura de cada campo;
  - número de campos por registro;
  - quais campos são opcionais.

Observação

- Ver programas em C nos livros texto (Folk 92, 98) que ilustram a criação de arquivos com essas várias organizações.

Agradecimentos

- Slides da profa. Cristina Ferreira.

45