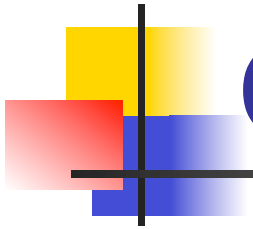


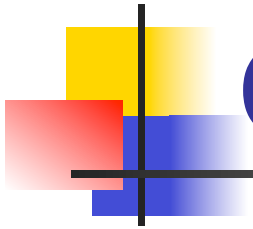
Organização de Arquivos

Leandro C. Cintra

M.C.F. de Oliveira

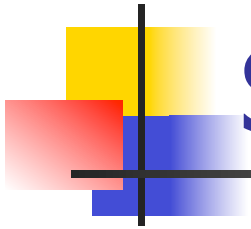
Thiago A. S. Pardo

Cristina D. A. Ciferri



Organização de Arquivos

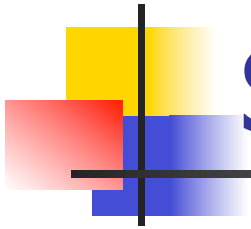
- Informações em arquivos são, em geral, organizadas logicamente em campos e registros
- Entretanto, campos e registros são conceitos lógicos, que não necessariamente correspondem a uma organização física
- Dependendo de como a informação é mantida no arquivo, campos lógicos sequer podem ser recuperados...



Sequência de bytes (*stream*)

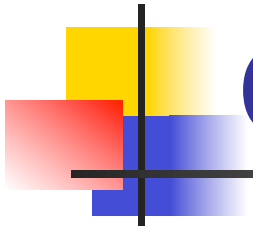
- Exemplo:
  - Suponha que desejamos armazenar em um arquivo os nomes e endereços de várias pessoas
  - Suponha que decidimos representar os dados como uma sequência de bytes (sem delimitadores, contadores, etc.)

AmesJohn123 MapleStillwaterOK74075MasonAlan90 EastgateAdaOK74820



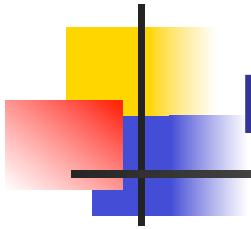
Sequência de bytes (*stream*)

- 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*)



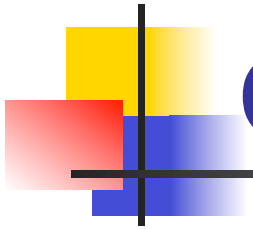
Organização em campos

- **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

- Forçar todos os campos para um tamanho (comprimento) fixo
- Começar cada campo com um indicador de tamanho (comprimento)
- Colocar delimitadores entre campos
- Usar expressões (*tags*) "keyword=value" para identificar cada campo e seu conteúdo



Campos com tamanho fixo

- Cada campo ocupa no arquivo um tamanho fixo, pré-determinado
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo

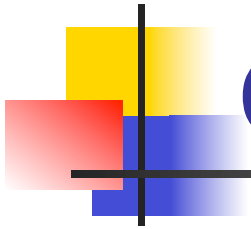
Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Campos com tamanho fixo

```
char last[10];
char first[10];
char city[15];
char state[2];
char zip[9];
```

ou

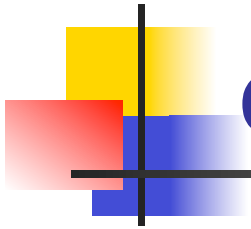
```
struct {
        char last[10];
        char first[10];
        char city[15];
        char state[2];
        char zip[9];
} set_of_fields;
```



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 apenas se o comprimento dos campos é realmente fixo, ou apresenta pouca variação

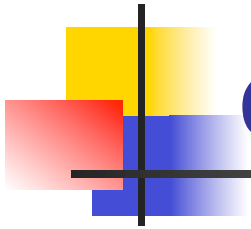
Campos com indicador de comprimento



- Tamanho de cada campo
  - armazenado imediatamente antes do dado
- Tamanho do campo menor que 256 bytes
  - espaço necessário para armazenar a informação: um único byte

05Maria05Rua 10312310São Carlos
04João05Rua A0325509Rio Claro
05Pedro06Rua 10025610Rib. Preto

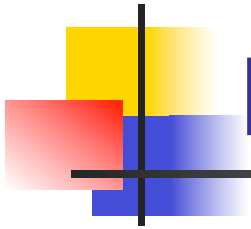
Campos separados por delimitadores



- Caractere(s) especial(ais) (que não fazem parte do dado)
  - escolhido(s) para ser(em) inserido(s) ao final de cada campo
- Ex.: para o campo *nome*
  - pode-se utilizar /, tab, #, etc ...
  - não pode-se usar espaços em branco ...

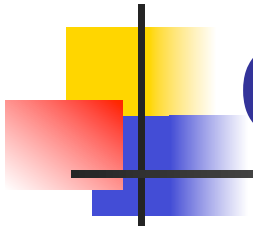
```
Maria|Rua 1|123|São Carlos|
João|Rua A|255|Rio Claro|
Pedro|Rua 10|56|Rib. Preto|
```

Uso de *tags*
Expressão "keyword=value"



- Vantagens
  - campo fornece informação semântica sobre si
  - fica mais fácil identificar o conteúdo do arquivo
  - fica mais fácil identificar campos "vazios"
    - *keyword* não aparece
- Desvantagem
  - as *keywords* podem ocupar uma porção significativa do arquivo

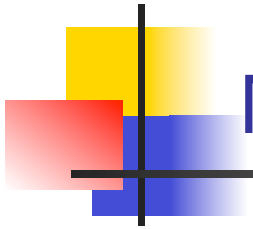
```
Nome=Maria|Endereço=Rua 1|Número=123|Cidade=São Carlos|
Nome=João|Endereço=Rua A|Número=255|Cidade=Rio Claro|
Nome=Pedro|Endereço=Rua 10|Número=56|Cidade=Rib. Preto|
```



Organização em registros

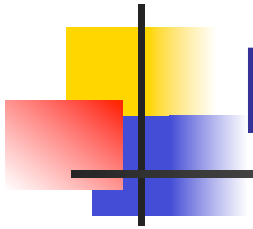
- Registro
  - conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade
  - permite a representação de um arquivo em um nível de organização mais alto

Assim como o conceito de campo, um registro é uma ferramenta conceitual, que não necessariamente existe no sentido físico



Métodos para organização em registros

- Forçar todos os registros para um tamanho fixo
- Forçar todos os registros para conterem um número fixo de campos
- Começar cada registro com um indicador de tamanho
- Usar índice para manter informação de endereçamento
- Colocar delimitadores entre registros



Registros de tamanho fixo

- 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 fixo
  - registros de tamanho fixo com campos de tamanho variável

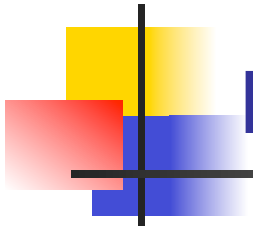
Registros de tamanho fixo

Registro de tamanho fixo e campos de tamanho fixo:

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Registro de tamanho fixo e campos de tamanho variável:

Maria|Rua 1|123|São Carlos| ← Espaço vazio →|
João|Rua A|255|Rio Claro| ← Espaço vazio →|
Pedro|Rua 10|56|Rib. Preto| ← Espaço vazio →|



Registros com número fixo de campos

- Cada registro contém um número fixo de campos
  - o tamanho do registro, em bytes, é variável
- Neste caso, os campos seriam separados por delimitadores

Registro com número fixo de campos:

Maria|Rua 1|123|São Carlos|João|Rua A|255|Rio Claro|Pedro|Rua 10|56|Rib. Preto|

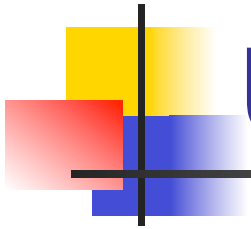
Indicador de tamanho para registros

- O indicador que precede o registro fornece o seu tamanho total, em bytes
- Os campos são separados internamente por delimitadores.

Método muito utilizado para manipular registros de tamanho variável

Registro iniciados por indicador de tamanho:

28Maria|Rua 1|123|São Carlos|25João|Rua A|255|Rio Claro|27Pedro|Rua 10|56|Rib. Preto|

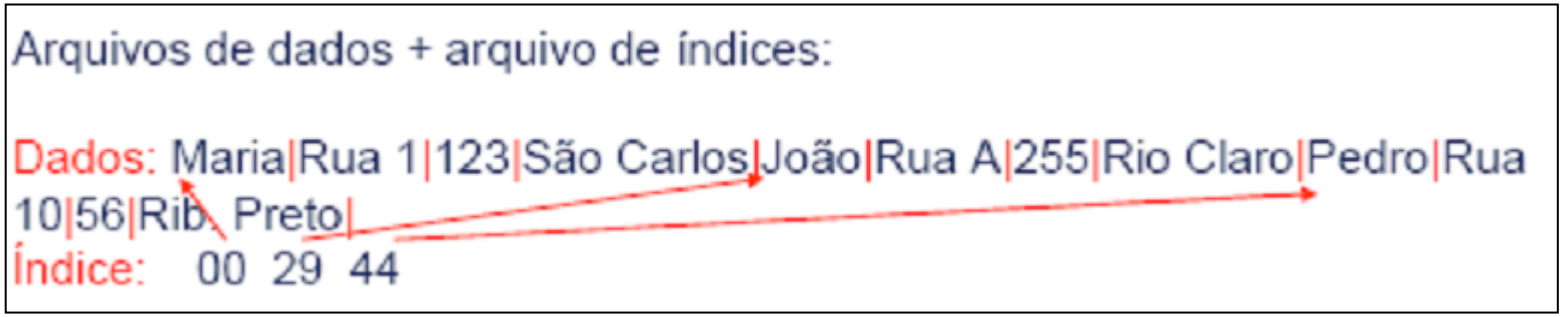


Utilizar um índice

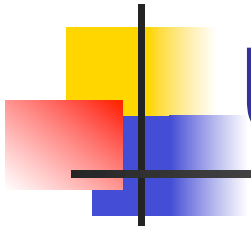
- Arquivo secundário que mantém informações sobre o endereço do primeiro *byte* de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo
- *Byte offset*
  - permite encontrar o começo de cada registro
  - permite calcular o tamanho dos registros

Utilizar um índice

- Característica adicional
  - campos separados por delimitadores



- Impacto
  - vantagem: flexibilidade
  - desvantagem: necessidade de acessar dois arquivos e armazenar conjuntamente

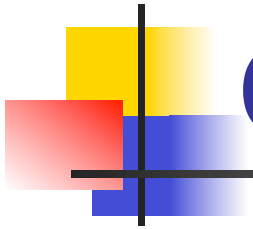


Utilizar delimitadores

- 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

Registro delimitado por marcador (#):

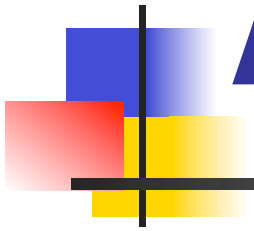
Maria|Rua 1|123|São Carlos|#João|Rua A|255|Rio Claro|#Pedro|Rua 10|56|Rib. Preto|



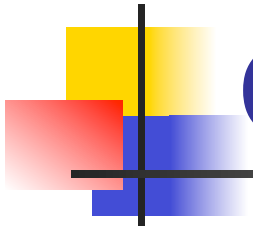
Observações

- Nenhum dos métodos descritos é apropriado para todas as situações
- Escolha do método depende
  - da natureza dos dados
  - para o que eles serão usados

Ver programas em C no livro texto
que ilustram a criação de arquivos
com as diferentes organizações estudadas!



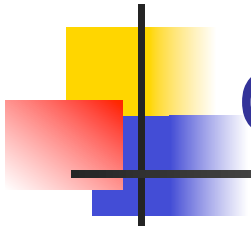
Acesso a registros



Chaves

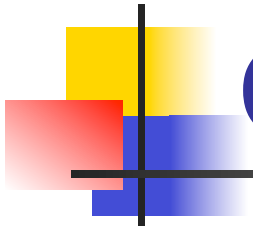
- Uma chave (*key*) está associada a um registro e permite a sua recuperação

- O conceito de chave é também uma ferramenta conceitual importante



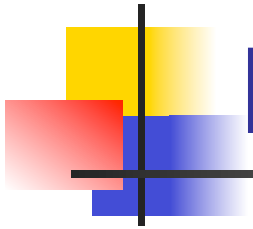
Chaves Primária e Secundária

- 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 vários registros
  - Ex. todos os “**Silvas**" que moram em **São Paulo**



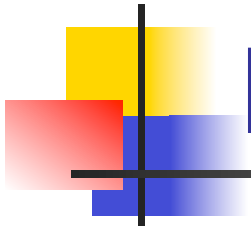
Chaves Distintas

- O ideal é que exista uma relação um a um entre chave e registro.
- Se isso não acontecer, é necessário fornecer uma maneira do usuário decidir qual dos registros é o que interessa
- A solução mais simples consiste em evitar tais confusões garantindo que a cada chave corresponda um único registro



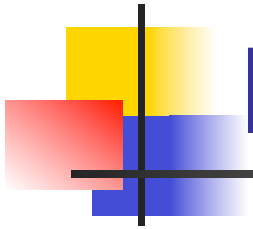
Escolha da Chave Primária

- 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



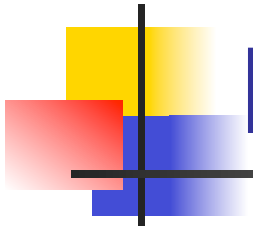
Forma canônica da chave

- "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

- Sequencial x Acesso Direto
- Desempenho
  - pesquisa em RAM
    - número de comparações efetuadas
  - pesquisa em disco
    - números de acessos a disco
- Mecanismo de avaliação do custo
  - contagem do número de chamadas à função de leitura de arquivo (READ)

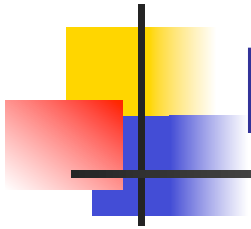


Busca sequencial

- Busca pelo registro que tem uma determinada chave em um arquivo
- Lê o arquivo, registro a registro
- Custo

$C_{busca\_sequencial} = n$

onde n é o tamanho do arquivo

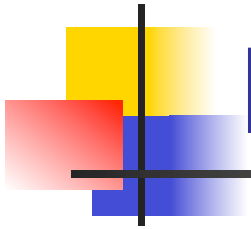


Blocagem de Registros

- Considerações
  - *seeking* é a parte mais lenta de uma operação de acesso a disco
  - custo de buscar e ler um registro, depois buscar e ler outro registro
    - maior que o custo de buscar (e depois ler) dois registros sucessivos de uma só vez
- Solução
  - ler um bloco de registros por vez, e então processar a leitura dos registros desse bloco em RAM

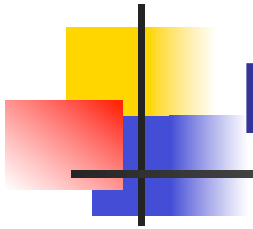
$$C_{busca\_sequencial} = b$$

onde b é o número de blocos que contêm os registros



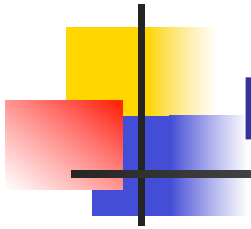
Blocagem de Registros

- Melhora o desempenho, mas o custo continua diretamente proporcional ao tamanho do arquivo
- Reflete a diferença entre o custo de acesso à RAM e o custo de acesso a disco
- Não altera o número de comparações em RAM
- Aumenta a quantidade de dados transferidos entre o disco e RAM
- Economiza tempo porque reduz o número de operações de *seeking*



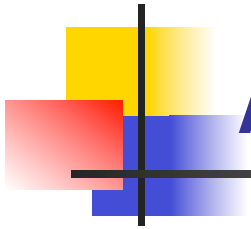
Busca Sequencial

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



Busca Sequencial

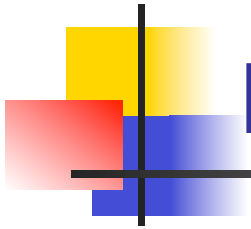
- Quando usar?
  - na busca por uma cadeia específica em um arquivo ASCII
  - em arquivos com poucos registros (i.e., 10)
  - em arquivos pouco pesquisados (i.e. armazenamento terciário)
  - na busca por registros com um certo valor de chave secundária, para a qual se espera muitos registros (muitas ocorrências)



Acesso Direto

- Alternativa mais radical ao acesso sequencial
- Realiza um *seeking* direto para o início do registro desejado (ou do setor que o contém) e lê o registro imediatamente
- Custo
  - um único acesso traz o registro, independentemente do tamanho do arquivo

$$C_{acesso\_direto} = 1$$

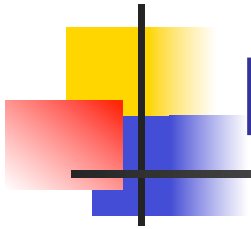


Posição do início do registro

- Uso de um arquivo índice separado

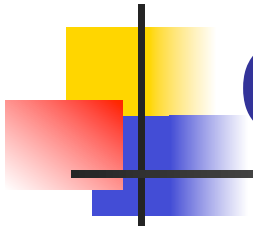
ou

- Uso do RRN (*relative record number*)
  - fornece a posição relativa do registro dentro do arquivo
  - permite identificar o byte offset



Posição de um registro com RRN

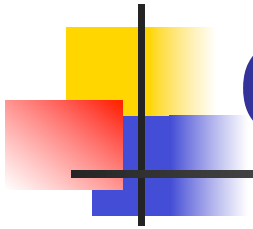
- Para utilizar o RRN, é necessário trabalhar com registros de tamanho fixo
- Posição de início do registro:
  - *byte offset* = RRN x tamanho do registro
  - Por exemplo, se queremos a posição do registro com RRN 546, e o tamanho de cada registro é 128:
    - *byte offset* é 546 x 128 = 69.888



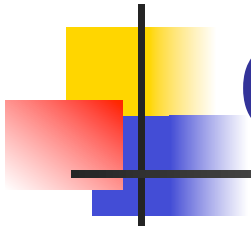
Acesso a arquivos X Organização de Arquivos

- **Organização de Arquivos**
  - registros de tamanho fixo
  - registros de tamanho variável
- **Acesso a Arquivos**
  - acesso sequencial
  - acesso direto

Acesso a arquivos X Organização de Arquivos



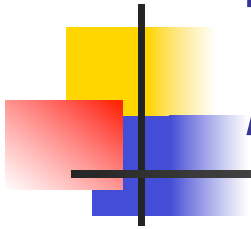
- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?
  - como separar os registros?
  - como identificar o espaço utilizado e o "lixo"?
- Existem muitas respostas para estas 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*)



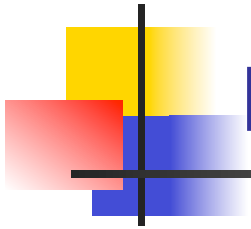
- Em geral, é interessante manter algumas informações sobre o arquivo para uso futuro
  - Essas informações podem ser mantidas em um cabeçalho no início do arquivo
  - A existência de um registro cabeçalho torna um arquivo um objeto auto-descrito
    - o software pode acessar arquivos de forma mais flexível, porém o software tem que ser mais elaborado

Registro Cabeçalho (*header record*)

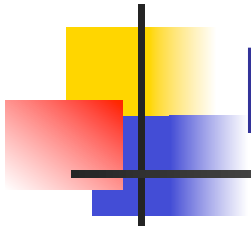
- Algumas informações típicas
  - Número de registros
  - Tamanho de cada registro
  - Nomes dos campos de cada registro
  - Tamanho dos campos
  - Datas de criação e atualização
- Pode-se colocar informações elaboradas

Metadados: dados que descrevem os dados primários em um arquivo

Estrutura e tamanho dos registros

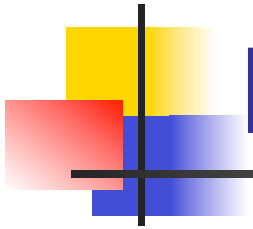


- Registros com tamanho fixo
  - vantagem
    - RRN pode ser usado para prover acesso direto aos registros
  - pergunta
    - qual o tamanho dos registros?
      - dependente do tamanho dos campos



Exemplo com Decisão Fácil

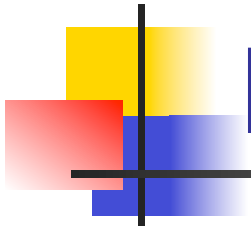
- Arquivo de controle de vendas
  - número do comprador: 6 dígitos
  - data no formato DDMMAA: 6 dígitos
  - número do item: 5 dígitos
  - quantidade vendida: 3 dígitos
  - valor da venda: 10 caracteres
- Campos de tamanho fixo: 30 bytes



Desempenho

- Registros de 30 bytes
  - páginas de disco de 4KB (4.096 bytes)
  - número de registros por página: 136,53
- Registros de 32 bytes
  - páginas de disco de 4KB (4.096 bytes)
  - número de registros por página: 128

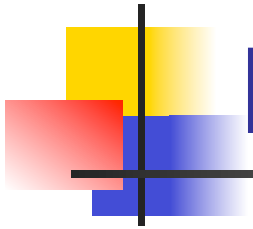
tamanho do registro deve se encaixar no tamanho da página de disco



Exemplo com Decisão Difícil

- Campos com tamanho muito variável
  - nome
  - endereço
- Abordagens simplistas para o tamanho do registro
  1. soma do tamanho máximo de cada campo
  2. soma do tamanho mínimo de cada campo

usar uma estrutura de campos adequada



Registros de Tamanho Fixo

- Campos de tamanho fixo
  - simplicidade
  - problemas das opções 1 e 2
- Campos de tamanho variável
  - pode-se aplicar o efeito do tamanho médio
    - *heurística: nomes mais longos em geral não aparecem no mesmo registros que os endereços mais longos*