

# **Banco de Dados 2**

## **Organização de Arquivo**

Prof. Silvio R. R. Sanches



Cornélio Procópio

# Organização de Arquivo

- ▶ Um banco de dados é mapeado em diferentes arquivos
- ▶ Esses arquivos residem permanentemente nos discos
- ▶ Um arquivo é organizado logicamente como uma sequência de registros, mapeados para blocos de disco



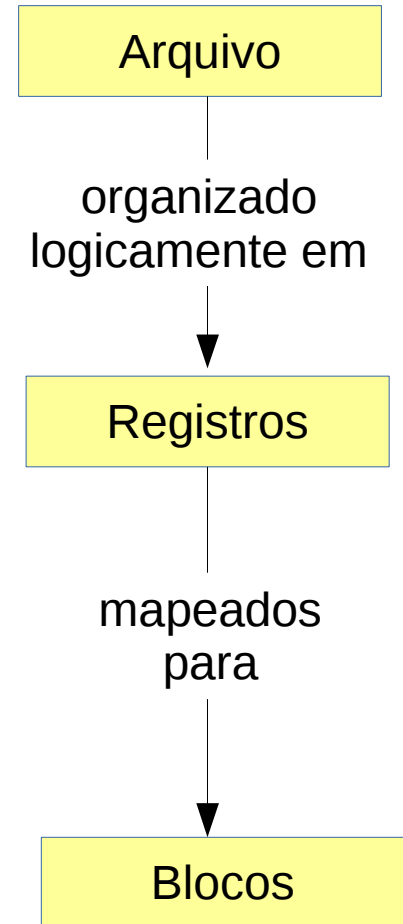
# Organização de Arquivo



- ▶ Os arquivos representam uma construção básica nos sistemas operacionais
- ▶ Devemos pressupor a existência de um sistema de arquivos básico
- ▶ Precisamos representar os modelos de dados lógicos em termos de arquivos:



- ▶ Cada arquivo é particionado logicamente em unidades de armazenamento de tamanho fixo, os **blocos**



Blocos são as unidades de alocação de armazenamento e transferência de dados  
Nos discos, representam um número fixo de setores contíguos

# Organização de Arquivo

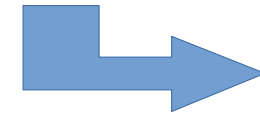
- ▶ Por padrão, a maioria dos bancos de dados utiliza tamanhos de blocos de 4 a 8 kilobytes

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	33343	El Said	History	60000

- ▶ Um bloco pode conter vários registros

- ▶ Devemos presumir que:

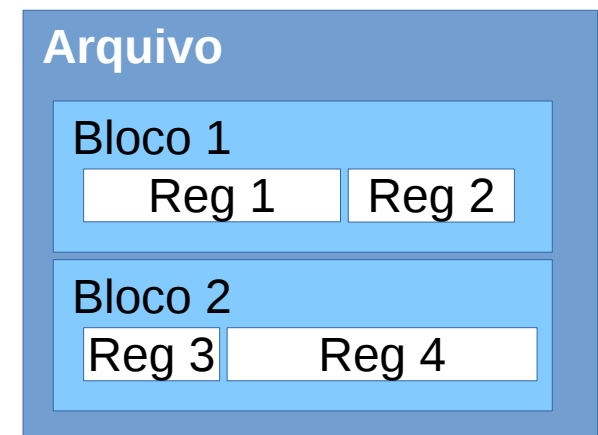
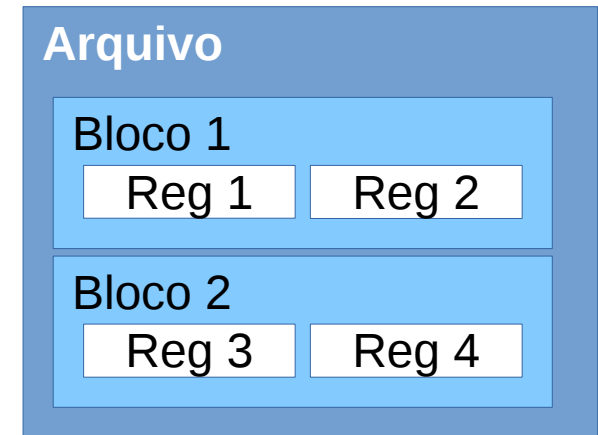
- ▶ **Nenhum registro é maior do que um bloco**
- ▶ **Cada registro está totalmente contido em um único bloco**
  - ▶ Não possui parte em um bloco e parte em outro
  - ▶ Essa restrição agiliza o acesso aos itens de dados



Há vários tipos de grandes itens de dados que podem ser maiores que um bloco de disco. Nesses itens o armazenamento é feito separado e um ponteiro para o item é armazenado no registro

# Organização de Arquivo

- ▶ Abordagens para mapeamento do banco de dados para arquivos:
- ▶ **Utilizar vários arquivos e armazenar registros de apenas um tamanho fixo em um arquivo específico**
- ▶ **Estruturar os arquivos de modo que possa acomodar vários tamanhos de registros**



# Registros de Tamanho Fixo

- ▶ Considere o arquivo de um registro chamado *instrutor*:
  - ▶ Cada registro desse arquivo está definido (em pseudocódigo) como:

**type** instrutor = **record**

*ID* **varchar**(5);

*nome* **varchar**(20);

*nome\_dept* **varchar**(20);

*salário* **numeric**(8,2)

**end**

**Tipo *numeric* no PostgreSQL:**

Exemplo: o número 235100,41

**Precisão** 8 (total de dígitos significativos)

**Escala** 2 (número de dígitos da parte decimal)

# Registros de Tamanho Fixo

- ▶ Suponha que:
  - ▶ Cada caractere ocupe 1 byte
  - ▶ O tipo numeric(8,2) ocupe 8 bytes
  - ▶ Atribuámos aos atributos *ID*, *nome* e *nome\_dept* o máximo de bytes que eles podem conter

```
type instrutor = record
    ID varchar(5);
    nome varchar(20);
    nome_dept varchar(20);
    salario numeric(8,2)
end
```

O registro instrutor terá quantos bytes?

53

# Registros de Tamanho Fixo

- ▶ Técnica simples:
  - ▶ Usa os primeiros 53 bytes para o primeiro registro
  - ▶ Os 53 bytes seguintes para o segundo registro
  - ▶ E, assim, sucessivamente

```
type instrutor = record  
    ID varchar(5);  
    nome varchar(20);  
    nome_dept varchar(20);  
    salario numeric(8,2)  
end
```

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

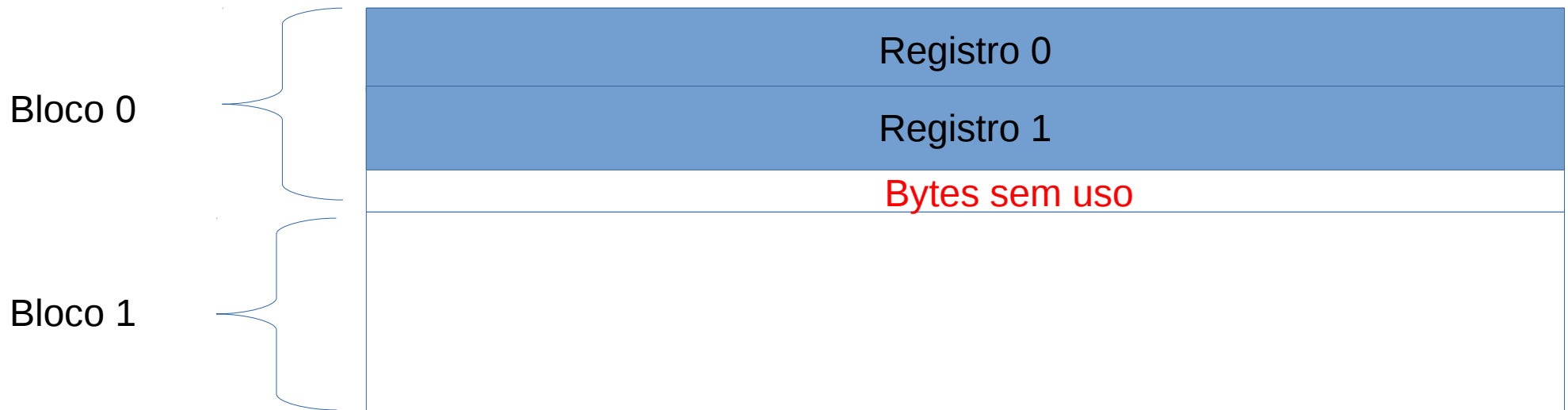


# Registros de Tamanho Fixo

- ▶ Problemas:
  - ▶ Os registros ultrapassarão os limites dos blocos:
    - ▶ Parte de um registro será armazenado em um bloco e parte em outro
    - ▶ Quantos acessos de bloco é necessários para ler ou gravar nesse registro?
      - ▶ É necessário dois acessos
      - ▶ Exceto se o tamanho do bloco for múltiplo de 53 (improvável)
  - ▶ É difícil excluir um registro dessa estrutura:
    - ▶ O espaço deixado pelo registro excluído deve ser preenchido por outro registro ou marcados para que sejam ignorados

# Registros de Tamanho Fixo

- ▶ Para evitar o primeiro problema:
  - ▶ Atribuímos a um bloco uma quantidade de registros que se acomodaria totalmente nesse bloco
    - ▶ Dividir o tamanho do bloco pelo tamanho do registro e descartar a parte fracionária
    - ▶ Os bytes restantes ficarão sem uso



# Registros de Tamanho Fixo

- ▶ Quando um registro for excluído, podemos deslocar sucessivamente os registros posteriores para ocupar os espaços

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Arquivo da figura anterior com o registro 3 excluído

# Registros de Tamanho Fixo

Exclusão

## ► Problema:

- Movimentação de grande quantidade de registros

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

## ► Possível solução:

- Mover o registro do final para o espaço ocupado

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

# Registros de Tamanho Fixo

- ▶ **Não é desejável** mover registros para ocupar um espaço liberado por um registro excluído:
  - ▶ Exige acessos adicionais ao bloco
- ▶ **Solução:** usar um cabeçalho:
  - ▶ Terá informações sobre o arquivo
  - ▶ Armazenaremos o endereço do próximo registro disponível



# Registros de Tamanho Fixo

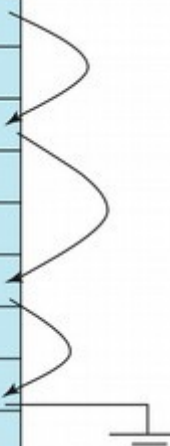
- ▶ Armazenar o endereço do primeiro registro excluído no cabeçalho do arquivo.
- ▶ Utilize este primeiro registro para armazenar o endereço do segundo registro excluído, e assim por diante
- ▶ Pode pensar nesses endereços armazenados como ponteiros, uma vez que "apontam" para a localização de um registro.

header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

# Registros de Tamanho Fixo

- ▶ Na inserção de um novo registro:
  - ▶ Usamos o registro apontado pelo cabeçalho
  - ▶ Mudamos o ponteiro do cabeçalho para que aponte para o próximo disponível
  - ▶ Se não houver espaço disponível, inserimos no final

header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000



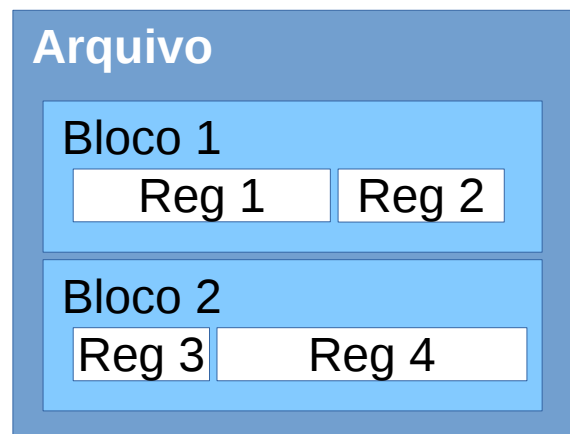
# Exercícios





# Registros de Tamanho Variável

- ▶ Dois problemas:
- ▶ Como representar um **único registro**, de modo a permitir a extração fácil dos atributos individuais
- ▶ Como armazenar registros de tamanho variável **dentro de um bloco**, de modo que os registros de um bloco sejam facilmente extraídos



# Registros de Tamanho Variável

- ▶ A representação de um registro de tamanho variável geralmente tem duas partes:
  - ▶ Uma parte inicial com atributos de **tamanho fixo**
  - ▶ Seguida por dados de atributos de **tamanho variável**
- ▶ Atributos de tamanho fixo:
  - ▶ Valores numéricos, datas, strings de tamanho fixo
  - ▶ Recebe a quantidade de bytes necessária para armazenar o respectivo valor

# Registros de Tamanho Variável

- ▶ Atributos de tamanho variável:
  - ▶ Exemplo: tipo *varchar*
- ▶ Representados na parte inicial do registro pelo par:
  - ▶ Deslocamento:
    - ▶ Indica onde os dados desse atributo começam dentro do registro
  - ▶ Comprimento:
    - ▶ Tamanho expresso em bytes do atributo de tamanho variável

# Registros de Tamanho Variável

- ▶ Atributos de tamanho variável (cont):
  - ▶ Valores armazenados de forma contígua após a parte de tamanho fixo inicial do registro
- ▶ Parte inicial:
  - ▶ Armazena um tamanho fixo de informação sobre cada atributo, seja ele de tamanho fixo ou variável

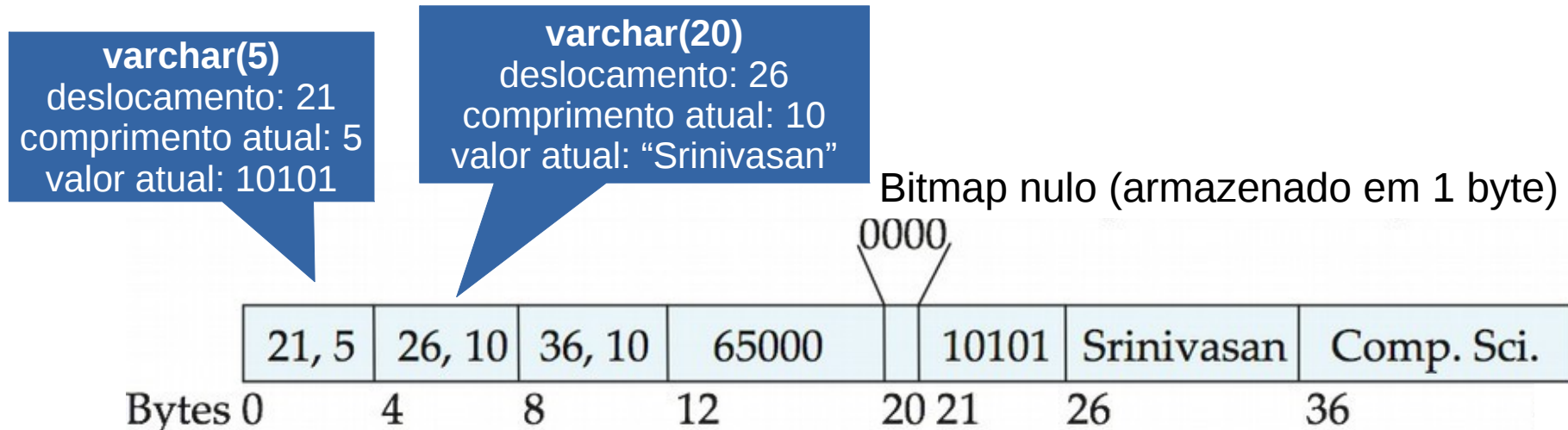
# Registros de Tamanho Variável

**type** instrutor = **record**

```
ID varchar(5);  
nome varchar(20);  
nome_dept varchar(20);  
salário numeric(8,2)
```

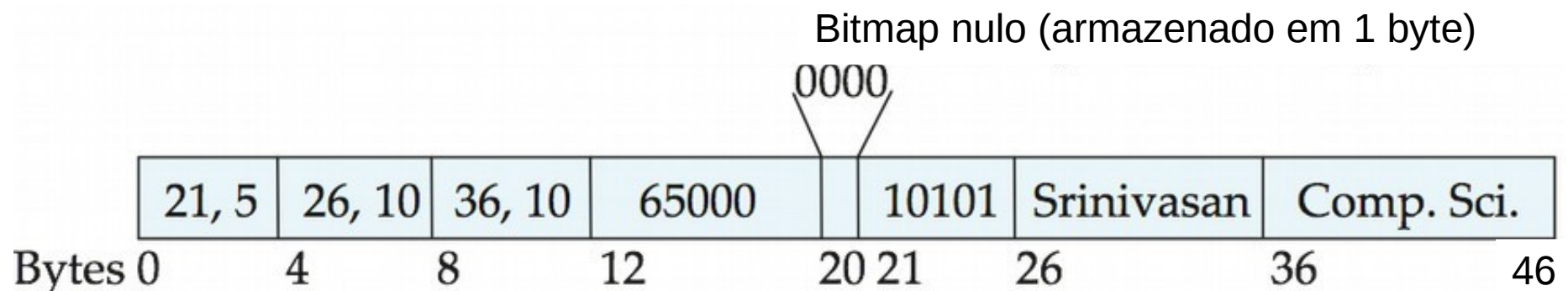
**end**

- ▶ Três primeiros atributos *ID*, *nome* e *nome\_dept* são strings de tamanho variável
- ▶ Quarto atributo *salário* é um número de tamanho fixo
- ▶ Valores de **deslocamento** e **comprimento** estão armazenados em 2 bytes cada um deles (total de 4 bytes por atributo)
- ▶ Pressupomos que:
  - ▶ O atributo *salário* é armazenado em 8 bytes:
  - ▶ Cada *string* usa uma quantidade de bytes igual a de seu número de caracteres



# Registros de Tamanho Variável

- ▶ Bitmap nulo:
  - ▶ Indica quais atributos do registro têm o valor nulo
  - ▶ Nesse registro específico:
    - ▶ Se o salário fosse nulo, o quarto bit do bitmap seria definido como 1
    - ▶ O valor do salário armazenado nos bytes de 12 a 19 seria ignorado
  - ▶ Como o registro tem 4 atributos, o bitmap nulo desse registro cabe em 1 byte



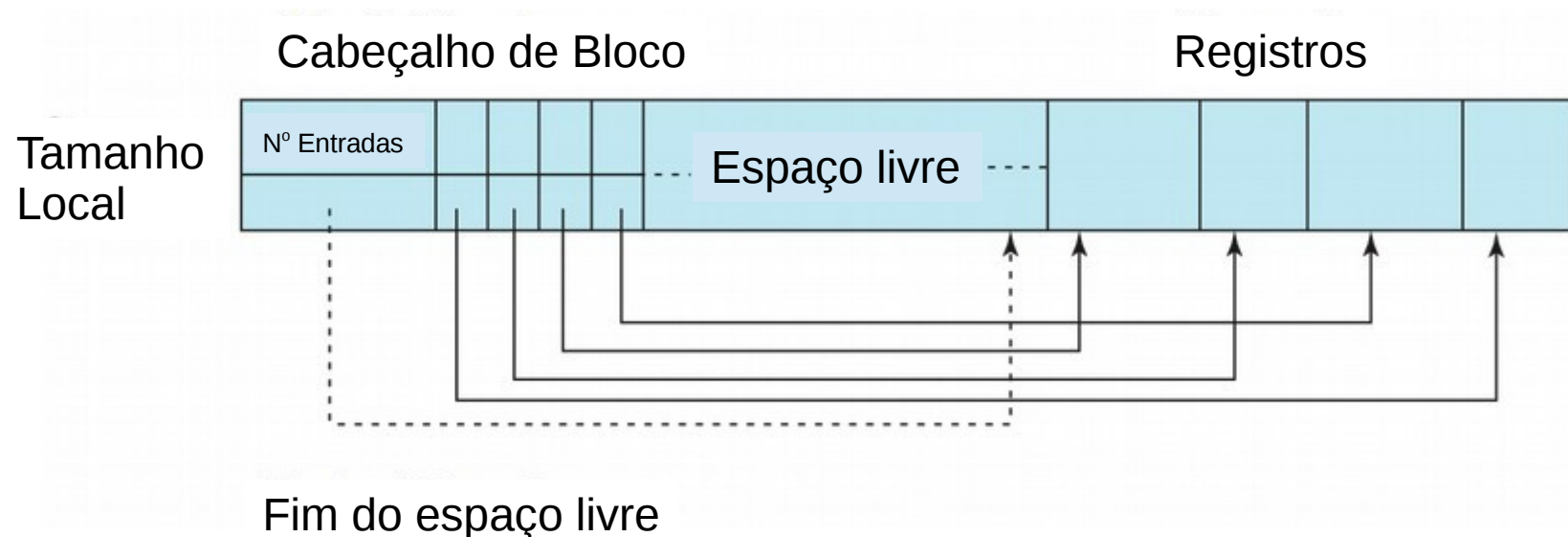
# Registros de Tamanho Variável

- ▶ Como armazenar registros de tamanho variável em um bloco?



# Registros de Tamanho Variável

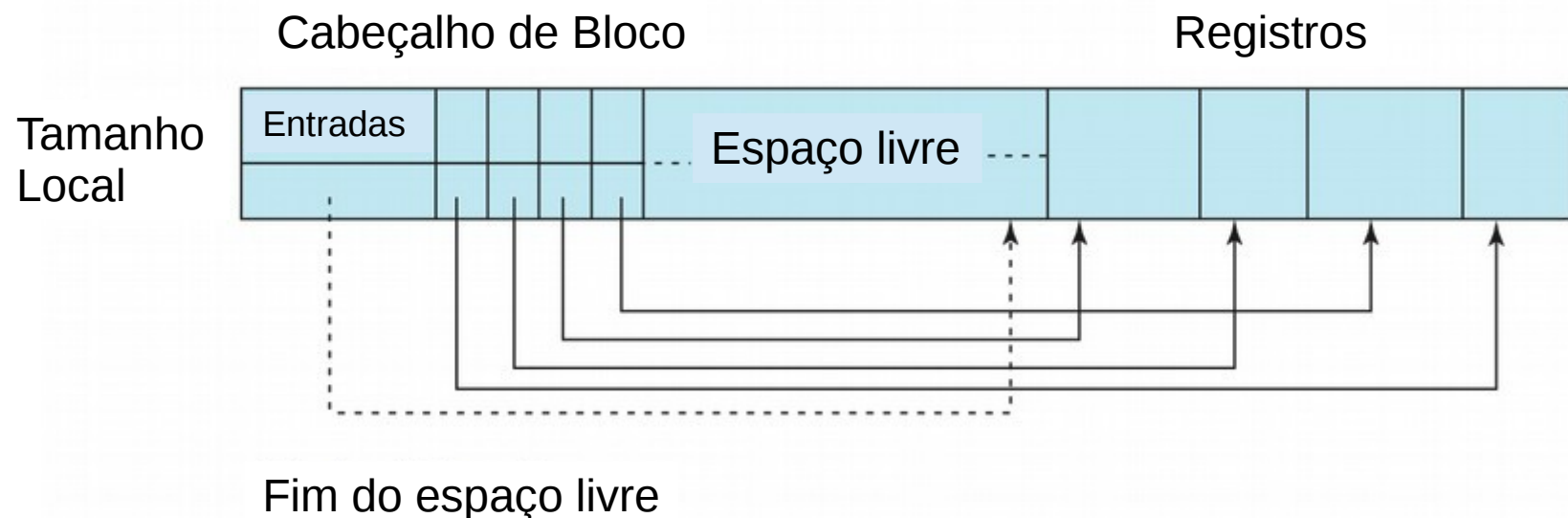
- ▶ Existe um **cabeçalho** no início de cada bloco que contem as seguintes informações:
  - ▶ O número de entradas de registro no cabeçalho
  - ▶ O final do espaço livre no bloco
  - ▶ Um *array* cujas entradas contêm o local e o tamanho de cada registro





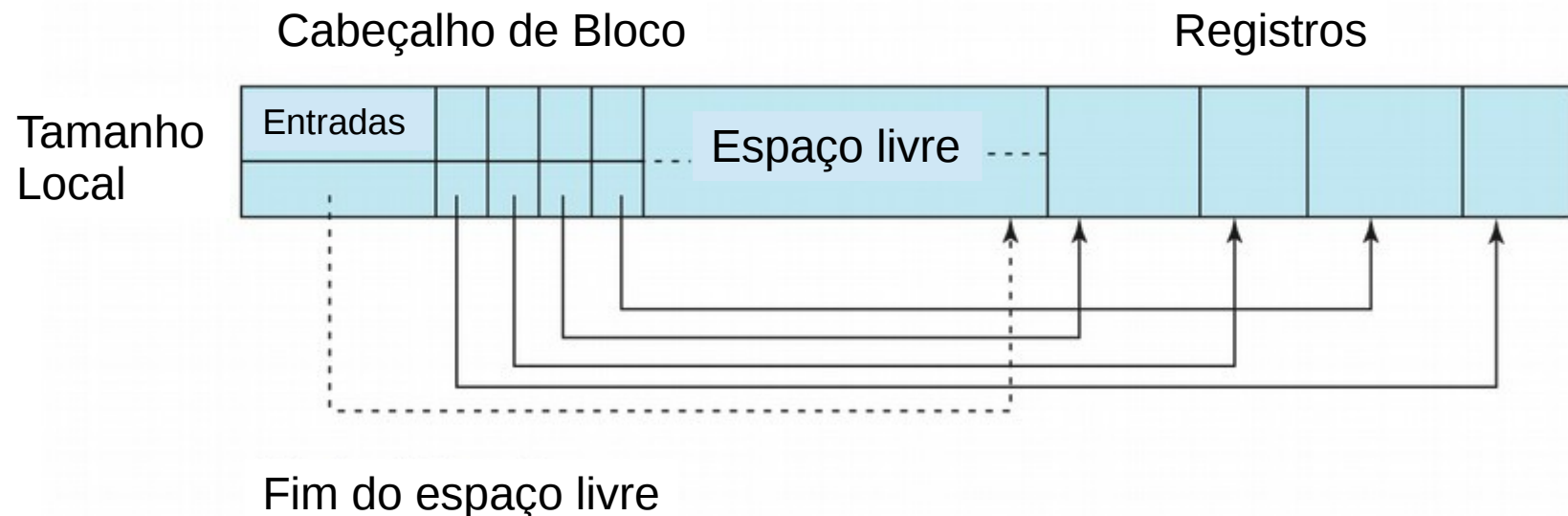
# Registros de Tamanho Variável

- ▶ Os registros reais são alocados de forma *contígua* no bloco, começando do final do bloco
- ▶ O espaço livre no bloco é contíguo entre a entrada final no *array* do cabeçalho e o primeiro registro
- ▶ Se um registro for inserido:
  - ▶ O espaço é alocado para ele no final do espaço livre
  - ▶ Uma entrada contendo seu tamanho e local é acrescentada ao cabeçalho



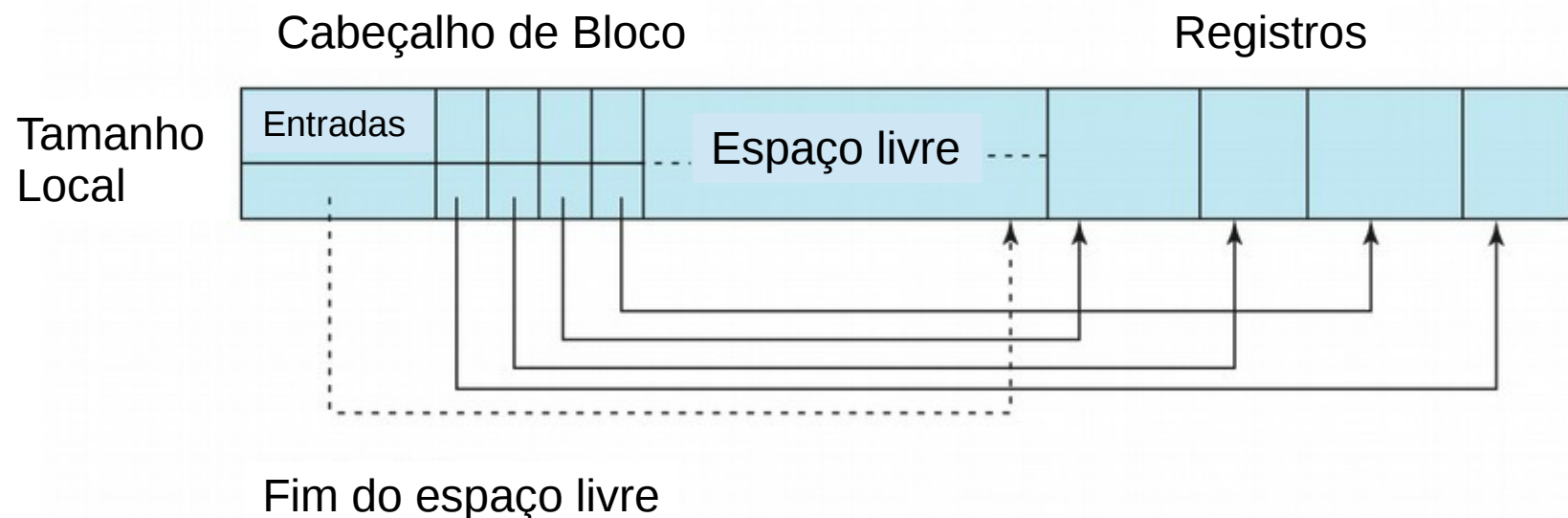
# Registros de Tamanho Variável

- ▶ Se um registro for excluído:
  - ▶ O espaço que ele ocupa é liberado
  - ▶ Sua entrada é marcada como excluída
    - ▶ Exemplo: seu tamanho marcado como -1
  - ▶ Os registros no bloco antes do registro excluído são movidos
    - ▶ O espaço livre criado pela exclusão é ocupado
    - ▶ Todo o espaço livre novamente fica entre a entrada final no *array* do cabeçalho e o primeiro registro



# Registros de Tamanho Variável

- ▶ O ponteiro de fim do espaço livre no cabeçalho também é atualizado corretamente
- ▶ O custo para mover os registros não é muito alto, pois o tamanho de um bloco é limitado
  - ▶ Tipicamente de 4 a 8 kilobytes



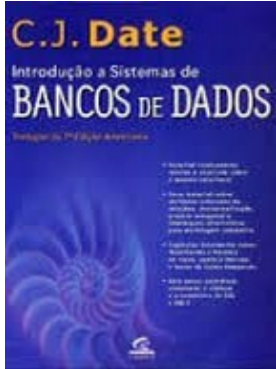
# Dados Grandes

- ▶ Os bancos de dados normalmente armazenam dados que podem ser muito maiores do que um bloco de disco
- ▶ Exemplo:
  - ▶ Uma imagem ou uma gravação de áudio podem ter vários megabytes
  - ▶ Um objeto de vídeo pode ocupar vários gigabytes
- ▶ A SQL admite os tipos **blob** e **clob**, que armazenam objetos binários e de caracteres muito grandes

# Dados Grandes

- ▶ A maioria dos bancos de dados relacionais restringe o tamanho de um registro para não ser maior que o tamanho de um bloco
  - ▶ Simplifica o gerenciamento
- ▶ Objetos grandes normalmente são armazenados em um arquivo especial em vez de serem armazenados com os outros atributos (curtos) do registro que ocorrem
- ▶ Um ponteiro (lógico) para o objeto é, então, armazenado no registro que contém o objeto grande
  - ▶ Objetos grandes normalmente são representados por organizações de arquivos em árvore B+

# Bibliografia Básica



- ▶ DATE, C. J. Introdução a sistemas de bancos de dados. Rio de Janeiro, RJ: Campus, 2000.



- ▶ ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de banco de dados. 4. ed. São Paulo: Pearson Addison-Wesley, 2005.



- ▶ SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de banco de dados. Rio de Janeiro, RJ: Elsevier, 2006.