# Bancos de Dados I
# Representação de Dados

Prof. Altigran Soares da Silva
IComp/UFAM


V2018.1

## Resumo

- Elementos de Dados e Campos
- Registros
- Dados e Registros de Tamanho Variável
- Alocação de Registros em Blocos
- Buffers
- Representação de Endereços de Bloco e de Registro
- Modificações em Registros
- Organização de Arquivos
- Modelo de Computação Baseado em E/S

# Elementos de Dados e Campos

## Elementos de Dados

```
CREATE TABLE Product (
  pid INT PRIMARY KEY,
  name CHAR(20),
  description VARCHAR(200),
  maker CHAR(10) REFERENCES Company(name))
```

- Como são representados internamente os valores de atributos, tuplas, referências, etc.?

# Valores Numéricos

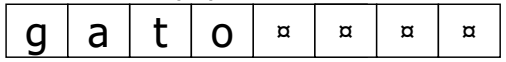
- Inteiros: 2 bytes (curto) ou 4 bytes
  - Ex: 35 é 00000000 00100011
- Real, ponto flutuante
  - Mantissa de *n* bits, expoente de *m* bits

# Valores Alfa-Numéricos

- Caracteres
  - Vários esquemas de codificação
  - Mais popular é o ASCII
  - Exemplos
    - A: 1000001
    - a: 1100001
    - 5: 0110101
    - LF: 0001010

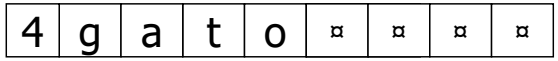
# Valores Alfa-Numéricos (2)

- Tamanho fixo – CHAR (n)
  - Ex: CHAR (8)




- Tamanho variável – VARCHAR (n)
  - Terminação com "null"
    - Ex: VARCHAR (8)


  - Tamanho codificado
    - Ex: VARCHAR (8)




# Valores Alfa-Numéricos (3)

| | CHAR(n) | VARCHAR(n) |
|---|---|---|
| Livro | N+1 | N+1 |
| PGSQL | N+4 | L+4 |
| MSSQL | N? | L+2? |
| MySQL | N | L+1 |
| Oracle | N | L+2? |

## Outros Tipos

- Boolean
  - TRUE: 1111 1111
  - FALSE: 0000 0000
- Enumeráveis
  - Ex: VERMELHO → 1, VERDE → 3, AZUL → 2 ROXO → 4

## Datas e Tempo

- Datas – Várias opções
  - Inteiro – nr. de dias desde 1º. de Janeiro de 1900
  - 8 caracteres: AAAAMMDD
  - 7 caracteres: AAAADDD
- Tempo
  - Inteiro – nr. de segundos desde 00:00:00
  - Caracteres: HHMMSSFF

## Observações

- Basicamente duas formas de implementação
  - Itens de tamanho fixo
  - Itens de tamanho variável
    - Em geral, o tamanho é dado no início da representação
- Tipos de dados
  - Presentes no catálogo – meta dados
  - Determinam como o padrão de bits deve ser implementado

# Registros

## Registros

- Registros
  - Coleção de de itens de dados relacionados.
  - Os itens são chamados ”Campos”
- Principais alternativas:
  - Formato Fixo vs. Formato Variável
  - Tamanho Fixo vs. Tamanho Variável

## Registros de Formato Fixo

- Depende de informação do esquema
  - Número de campos
  - Tipo de cada campo
  - Ordem dentro do registro
  - Semântica de cada campo
- Esquema na forma de meta-dados
  - Catálogo ou Dicionário: conjunto dos meta-dados
  - Armazenado em separado e não com o registro

## Exemplo
## Formato e Tamanho Fixo

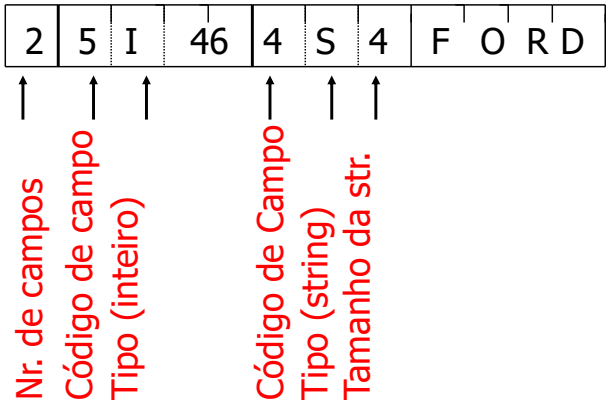
- Esquema
  (1) Mat, inteiro
  (2) Nome, 10 caracteres
  (3) Depart, inteiro
- Registros

| | | |
|---|---|---|
| 55 | s m i t h | 02 |
| 83 | j o n e s | 01 |

## Formato Variável

- O próprio registro contêm informação de formato. *É auto-descritivo*
- Útil para
  - Registros “esparsos”
  - Campos passíveis de repetição
  - Formatos em definição

# Exemplo
# Formato e Tamanho Variável




- Ao invés dos códigos de campo, poderiamos usar strings.
- Neste caso elas seriam chamadas TAGS

# Campos com repetição

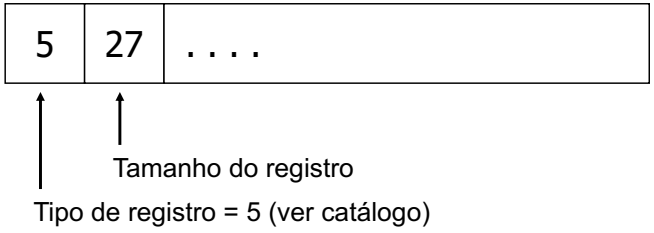
- Opções
  - Tamanho fixo X Cabeçalho
- Exemplo
  - Funcionário com um ou mais filhos

| 3 | NFun: José | Filho: Maria | Filho: Juca |
|---|---|---|---|

| NFun: José | Filho: Maria | Filho: Juca | ¤ |
|---|---|---|---|

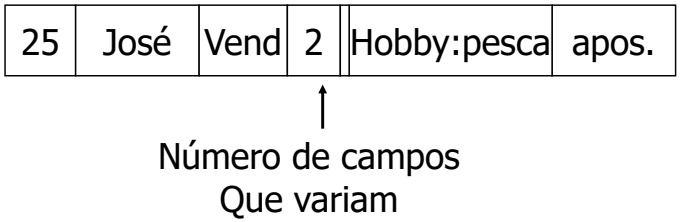
# Variações – Record Type

- Incluir meta-dados no registro
  - Ocupam bytes iniciais do registro
  - Tamanho
  - "time-stamps"
    - Criação, modificação, etc.
  - Tipo
    - Aponta para o esquema do registro no catálogo

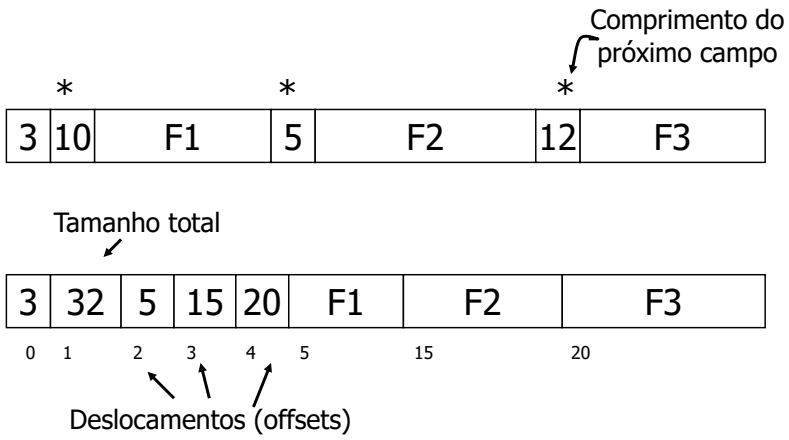



# Formatos Híbridos

- Formatos Híbridos
  - Uma parte é fixa, outra é variável
  - Exemplo:
    - Todos os funcionários têm matrícula, nome e departamento
    - Outros campos variam



# Outros formatos – Exemplos




# NULL - Representação

- NULL pode ser representado como um "valor reservado"
  - Diferentes representações para diferentes tipos de dados
  - Solução ingênua
- Se cabeçalhos forem usados, é possível codificar a ausência do atributo/campo

# NULL - Representação

- Se no cabeçalho o campo tem comprimento zero, então ele poderia ser considerado NULL
  - Problema string de comprimento zero
- Acrescentar um bit para cada campo do cabeçalho que pode assumir NULL

# BLOBS

- BLOB = Binary Large Objects
- Usados para representar:
  - Imagens (GIF, JPG); Filmes (MPEG); Áudio; etc.
- Armazenamento:
  - Seqüências de Blocos
  - Blocos podem ser contíguos ou encadeados
  - Podem ser espalhados em vários discos para aumentar a banda de transferência
- Alternativa
  - Armazenar os BLOBS em arquivos externos ao BD
  - Manter um apontador no registro para este arquivo

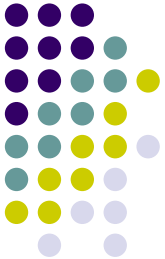
## BLOBS

- No BD
  - Manutenção das propriedades ACID
  - Portabilidade
  - Controle de Acesso
  - Melhor gerenciamento
  - Indexação
- Externo ao BD
  - Potencialmente mais eficiente
  - BD menos carregado
  - Manipulação externa ao BD
  - Se usado para a Web, cache em separado

## Outras técnicas

- Compressão
  - Dentro do registro
  - Coleção de registros
- Encriptação

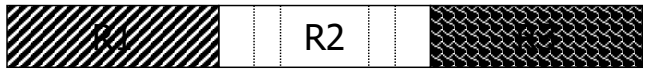
# Alocação de Registros em Blocos



## Alocando Registros em Blocos

- (1) Separação de registros
- (2) Espalhada vs. Não-espalhada
- (3) Agrupamento
- (4) Divisão de registros
- (5) Sequenciamento
- (6) Indireções

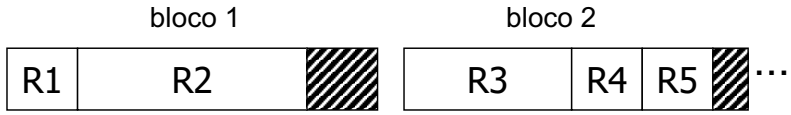
# Separação de Registros



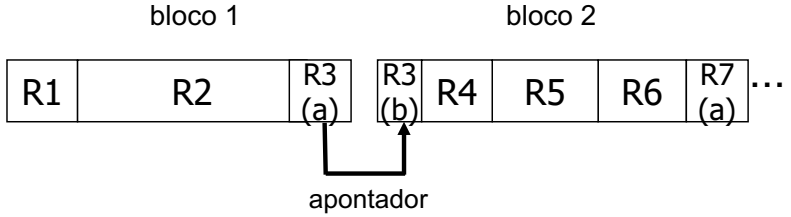

- Opções
  - Registros de tamanho fixo
  - Usar marcadores
  - Armazenar o comprimento ou offsets dos registos
    - Dento de cada registro
    - No cabeçalho do bloco

# Alocação Espalhada vs. Não-Espalhada

- Não espalhada




- Espalhada

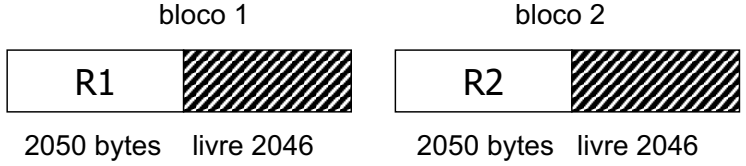



# Alocação Espalhada vs. Não-Espalhada

- Alocação não espalhada
  - É muito mais simples
  - Disperdiça espaço
- Alocação espalhada
  - Inevitável se tamaho do registro > tamanho do bloco

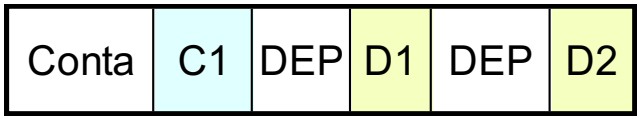
# Exemplo – Não espalhado

- Arquivo
  - $10^6$ registros de 2.050 bytes (fixos)
  - Espaço necessário = 2,05 x $10^9$
- Sistema
  - Tamanho do bloco = 4096 bytes
  - Espaço ocupado = 4096 x $10^6$ = 4,10 x $10^9$
- Utilização de 50%
- Espaço desperdiçado = 2,05 x $10^9$



# Agrupamento

- Representação de registros de tipos diferentes no mesmo bloco
- Justificativa: Registros que são frequentemente acessados em conjunto alocados próximos
- Exemplo:
  - Conta + Depósitos



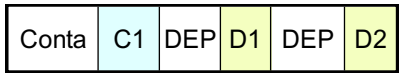

Bloco

# Agrupamento

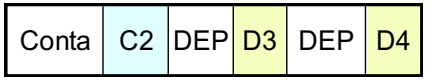
- Se Q1 é freqüente, o agrupamento é útil
- Se Q2 é freqüente, o agrupamento atrapalha

```
Q1: select NCONTA,CLIENTE,DATA,VALOR
    from   CONTA C, DEPOSITO D
    where  C.NCONTA = D.NCONTA

Q2: select NCONTA,CLIENTE,ENDEREÇO,...
    from   CONTA C
```



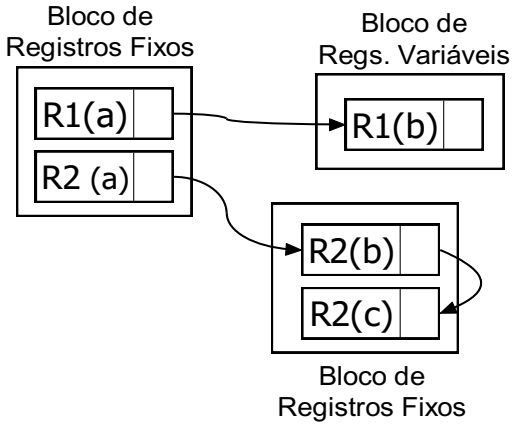

Bloco 1




Bloco 2

# Divisão de Registros

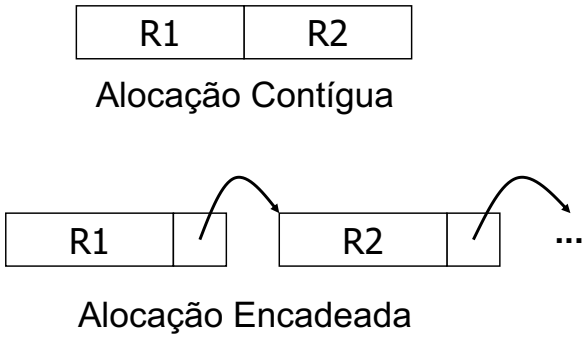
- Tipicamente usado para registros de formato híbrido
- Parte fixa em um bloco e parte variável em outro bloco




# Sequenciamento

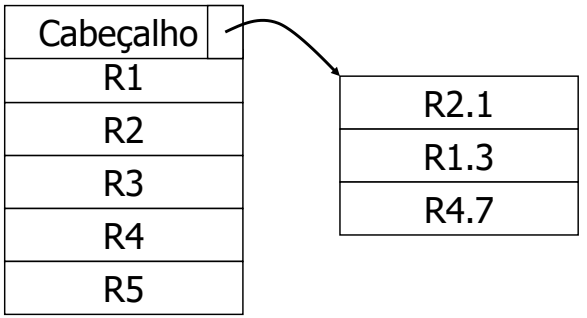
- Ordenação de registros em um arquivo e seus blocos através de uma valor chave
- Arquivos deste tipo são chamados *seqüenciais*
- Útil sempre que for freqüente o processamento do arquivo na seqüência da chave escolhida

## Sequenciamento – Opções




## Sequenciamento – Opções

- Área de Overflow




## Buffers

## Endereços Virtuais

- Os SO modernos usam *memória virtual*
  - O espaço de endereçamento é maior que a memória principal
  - É mantida uma camada de indireção entre endereços virtuais e endereços reais
  - A memória principal (física) serve como um cache dos endereços com mais probabilidade de acesso
  - Endereços com menos probabilidade de acesso são armazenados no disco quando a MP enche

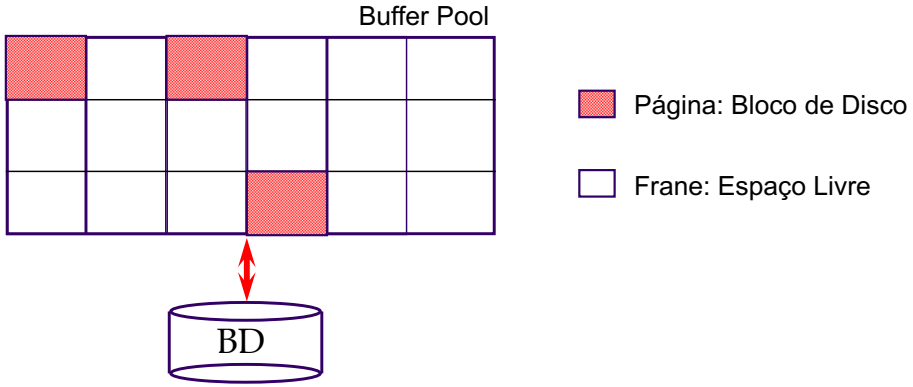
# Endereços Virtuais (2)

- Blocos transferidos para a memória são chamados de *páginas*
  - Acesso a disco: grande latência relativa a taxa de transferência (throughput)
  - Tira proveito do princípio da localidade de referência
  - Overhead de transferência é amortizado se considerarmos muitas requisições

# Gerência de Buffers (3)

- Os SGBDs usam uma estratégia similar à memória virtual
  - Buffer: Armazenam blocos de disco na MP
  - Buffer Pool: Conjunto de buffers
- Gerência de Buffers
  - Independe do que é feito pelo SO
  - Pre-carregamento de blocos com probabilidade de uso futuro
  - Blocos "pregados" (pinned blocks)
    - Blocos essenciais ou frequentemente acessados
  - Force-Writing
    - Forçar a escrita de blocos

# Gerência de Buffers (4)




# Gerência de Buffers (5)

- Quando uma página é requisitada pela 1ª. vez, é colocada num frame livre
  - pin_count = quantos processos precisam da página "pregada"
- Quando o SGBD escreve na página na memória, ela é marcada como "suja"
- Somente paginas não "pregadas" podem ser substituídas
- Políticas de substituição: LRU, Clock, MRU

# Representação de Endereços de Bloco e de Registro

## Indireções

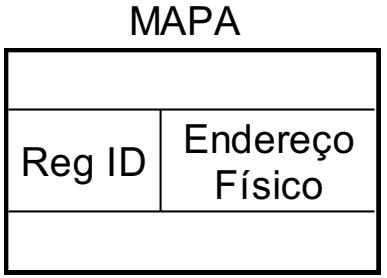
- Como referenciar registros
- Várias opções:
  - Desde usar o endereço físico até usar indireções


## Referência Puramente Física

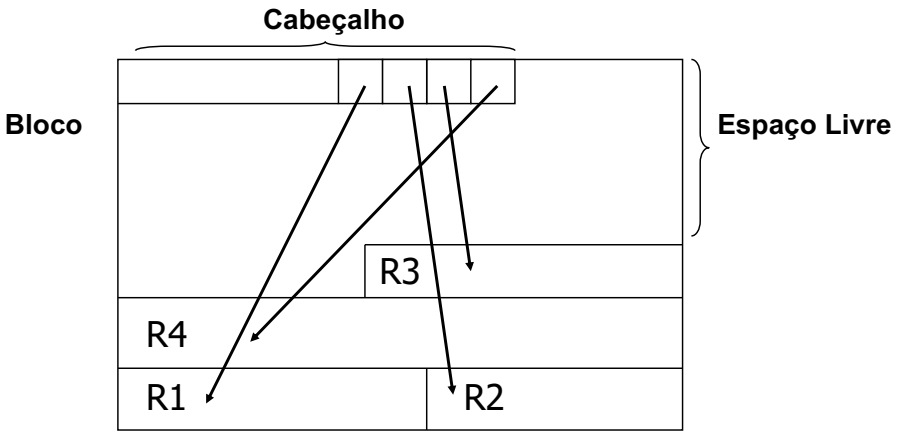
- Endereço do Registro =
  - Endereço do Bloco + Offset no Bloco
- Endereço do bloco =
  - ID do dispositivo +
  - Nr. do Cilindro +
  - Nr. da Trilha +
  - Nr. do Bloco

## Referência Totalmente Indireta

- Reg ID segue uma codificação de bits qualquer
  - + Flexibilidade para manipulação: inserção, remoção
  - - Custo para a manutenção da indireção



# Exemplo: Indireção no Bloco




# Cabeçalho de Bloco

- Informação que descreve o bloco
- Disposta no inicio do bloco
- Pode conter:
  - ID do Arquivo – Identifica a relação ou BD
  - ID do bloco
  - Diretório de registros
  - Apontador para espaço livre
  - Tipo do bloco – ex. “overflow”, “registro do tipo 4”, etc.
  - Ponteiro para outros blocos do mesmo tipo
  - Timestamp

# Indireções em Memória

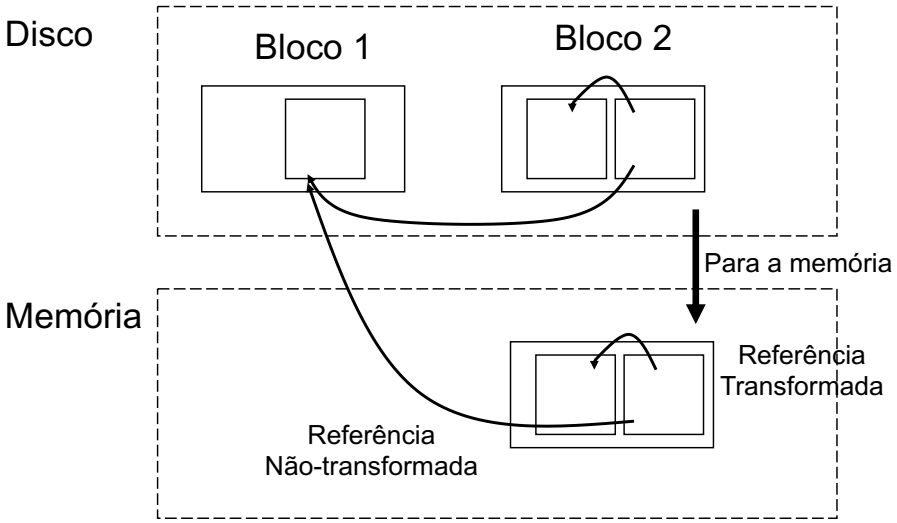
- Quando um bloco é trazido para a memória, recebe um endereço de memória principal
- O gerente de buffer usa outra tabela de endereços
- Implicação: referências feitas a partir da memória devem ser refeitas

| Endereço em Memória | Endereço Lógico |
|---|---|
| M1 | L1 |
| M2 | L2 |
| M3 | L3 |

# Referências Transformadas

- Transformação de Referências ou Pointer Swizzling
  - Processo de substituir referências lógicas ou físicas por referências de memória
- Ainda é necessário acessa a tabela de tradução, mas as próximas referências são mais rápidas

# Referências Transformadas (2)




# Referências Transformadas (3)

- Automática:
  - Quando o bloco é lido para a memória, todos os apontadores que precisam de transformação são transformados
- Sob demanda:
  - A transformação é feita a primeira vez que a referência é feita
- Sem transformação
  - Sempre usar a tabela de tradução de endereços

# Referências Transformadas (4)

- Quando os blocos retornam para o disco, a transformação tem de ser desfeita
- Perigo: podem haver referências para estes blocos
- Soluções
  - “Pregar” (pin) os blocos
  - Manter uma lista de referência para estes blocos

# Modificações em Registros

## Registros: Operações Físicas

- Principais operações que demanda reorganização da estrutura de armazenamento
  - Remoção
  - Inserção
- São freqüentes
- Demandam mais cuidado quando feitas no disco

## Remoção

- Opções
  - Remoção física: liberar espaço imediatamente
  - Remoção lógica: marcar como removido
    - Re-uso: pode demandar o encadeamento dos registros marcados como removidos
    - Formas de marcar:
      - Caractere especial
      - Campo especial
      - Utilizar um mapa

## Remoção: Tradeoffs

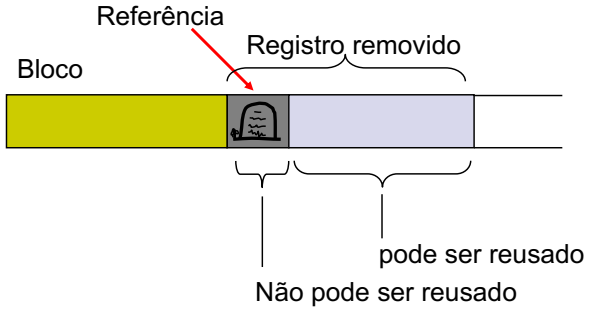
- Remoção física:
  - Qual o custo de mover os registros válidos de tal forma a liberar o espaço ocupado por registros removidos?
- Remoção lógica:
  - Quanto de espaço é perdido?
    - Registros removidos
    - Marcas de remoção
    - Encadeamento dos removidos

## Remoção – Indireções

- O que acontece com as referências (apontadores) a um registro quando ele é removido?
  - Apontadores pendurados (Dangling Pointers)
- Tratamento com Tombstones (Lápides)
  - Deixar um marca no mapa ou no local onde estava o registro

# Remoção – Indireções (2)

- Tombstones
  - IDs físicas




# Remoção – Indireções (3)

- Tombstones
  - Marcar a ausência do registro no mapa de alocação

mapa

| ID | LOC |
|---|---|
| | |
| 7788 | |
| | |

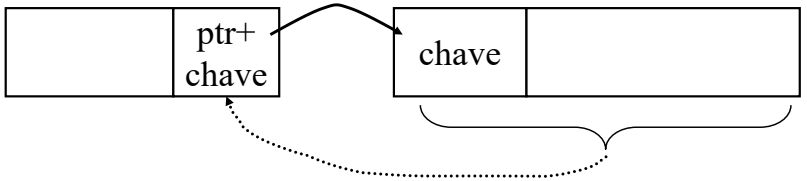
A ID 7788 e a entrada do mapa não podem ser reusadas

# Remoção – Indireções (4)

- Tratamento com IDs
  - Cada registro armazena um ID
  - Ao seguir o apontador, verificar se este leva ao registro correto


# Remoção – Indireções (5)

- Tratamento com Chaves
  - Acrescentar um valor de chave ao apontador
  - Ao seguir o apontador, verificar se este leva à chave correta



# Inserções

- Caso mais simples: arquivo seqüencial
  - Inserir novo registro no fim do arquivo
  - Se houver uma lista de removidos, reaproveitar espaço
  - Tratamento especial se os registros são de tamanho variável
- Caso mais complicado: arquivos ordenados
  - Deixar espaço disponível na vizinhança da inserção
  - Usar área de overflow

# Inserções – Questões

- Quanto de espaço livre deve ser deixado por bloco/trilha/cilindro
- O qual freqüente devem ser as reorganização de área de arquivo e de overflow