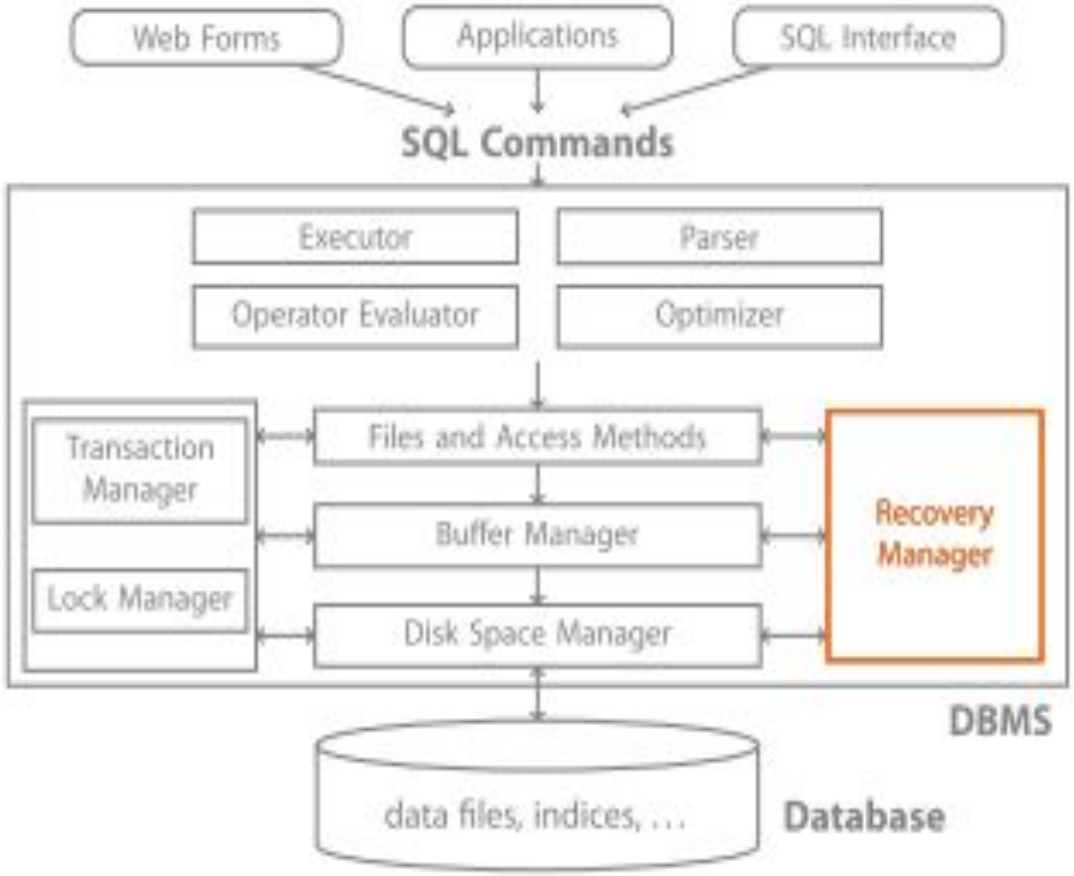
Gerenciamento de disco e buffer

SBGD


Web Forms
Applications
SQL Interface
SQL Commands
Executor
Parser
Operator Evaluator
Optimizer
Transaction Manager
Lock Manager
Files and Access Methods
Buffer Manager
Disk Space Manager
Recovery Manager
DBMS
data files, indices, . . .
Database

Introdução

O que mais implica no projeto de um SGBD

- Operações de escrita (write);
- Operações de leitura (read)

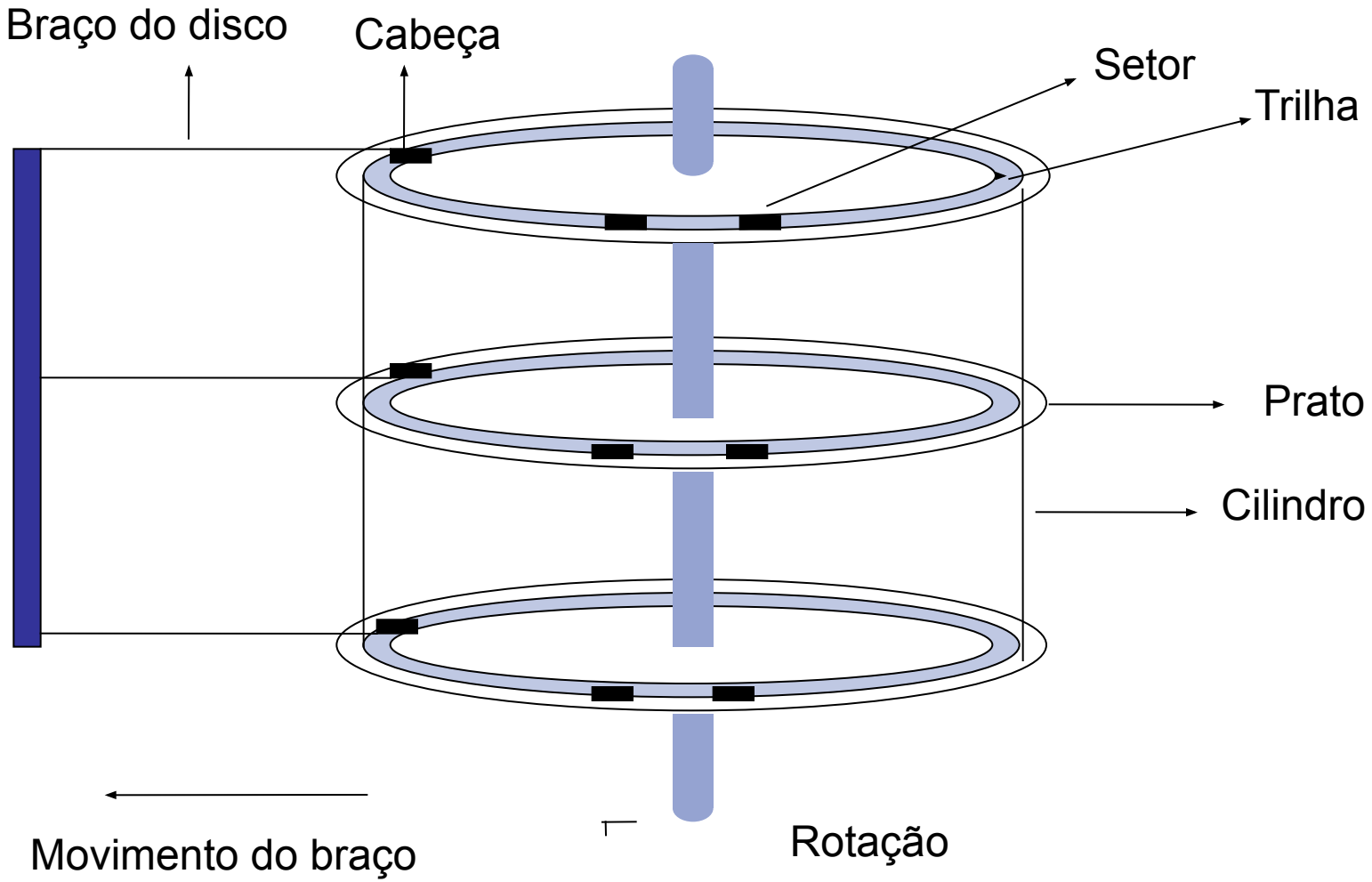
Ambas as operações têm um custo muito alto (tempo) e devem ser planejadas cuidadosamente

Discos Rígidos

Dados são armazenados e recuperados em unidades chamadas *blocos de disco*.

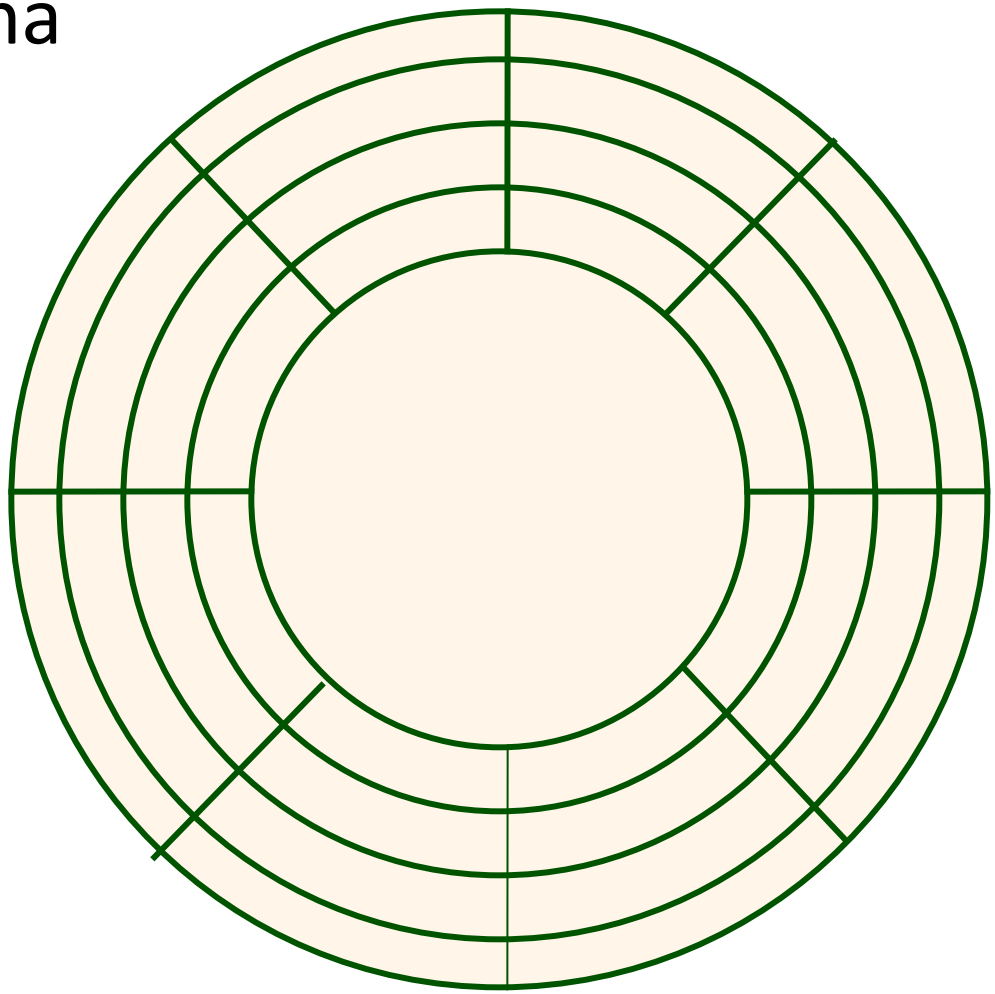
O tempo para recuperar um bloco no disco depende da posição onde se encontra.

Estrutura do Disco Rígido



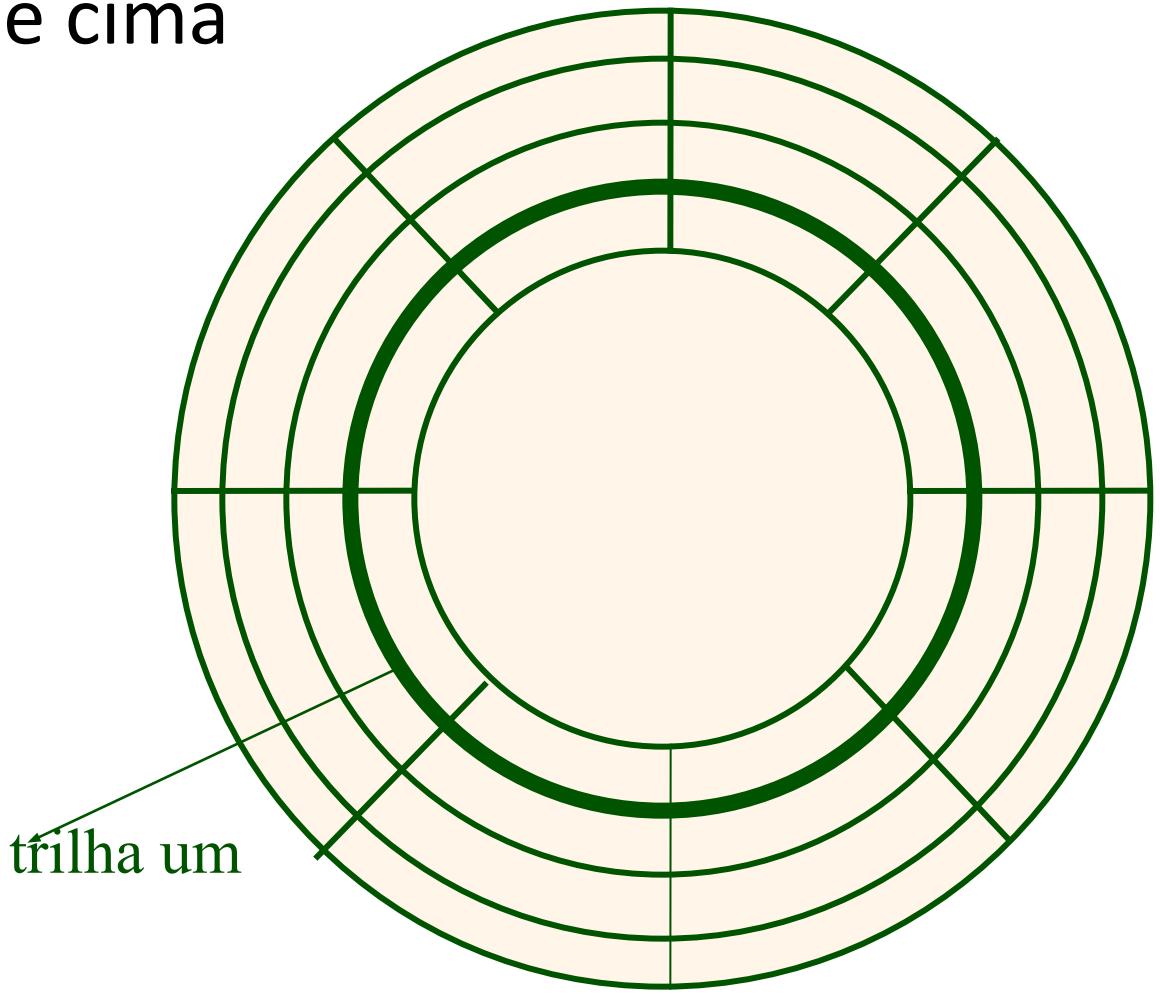
Componentes Disco Rígido

- Visto de cima



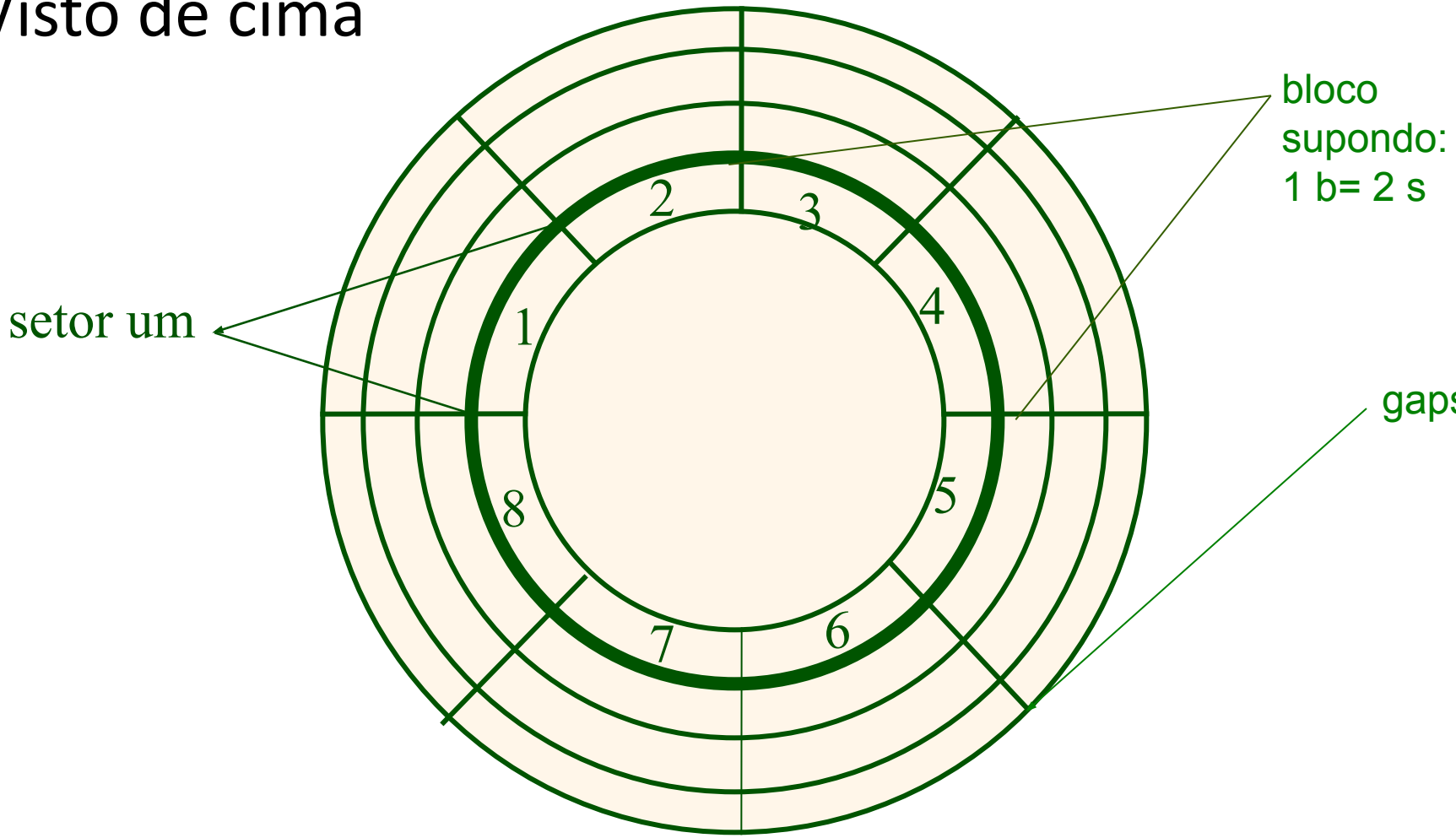
Componentes Disco Rígido

- Visto de cima

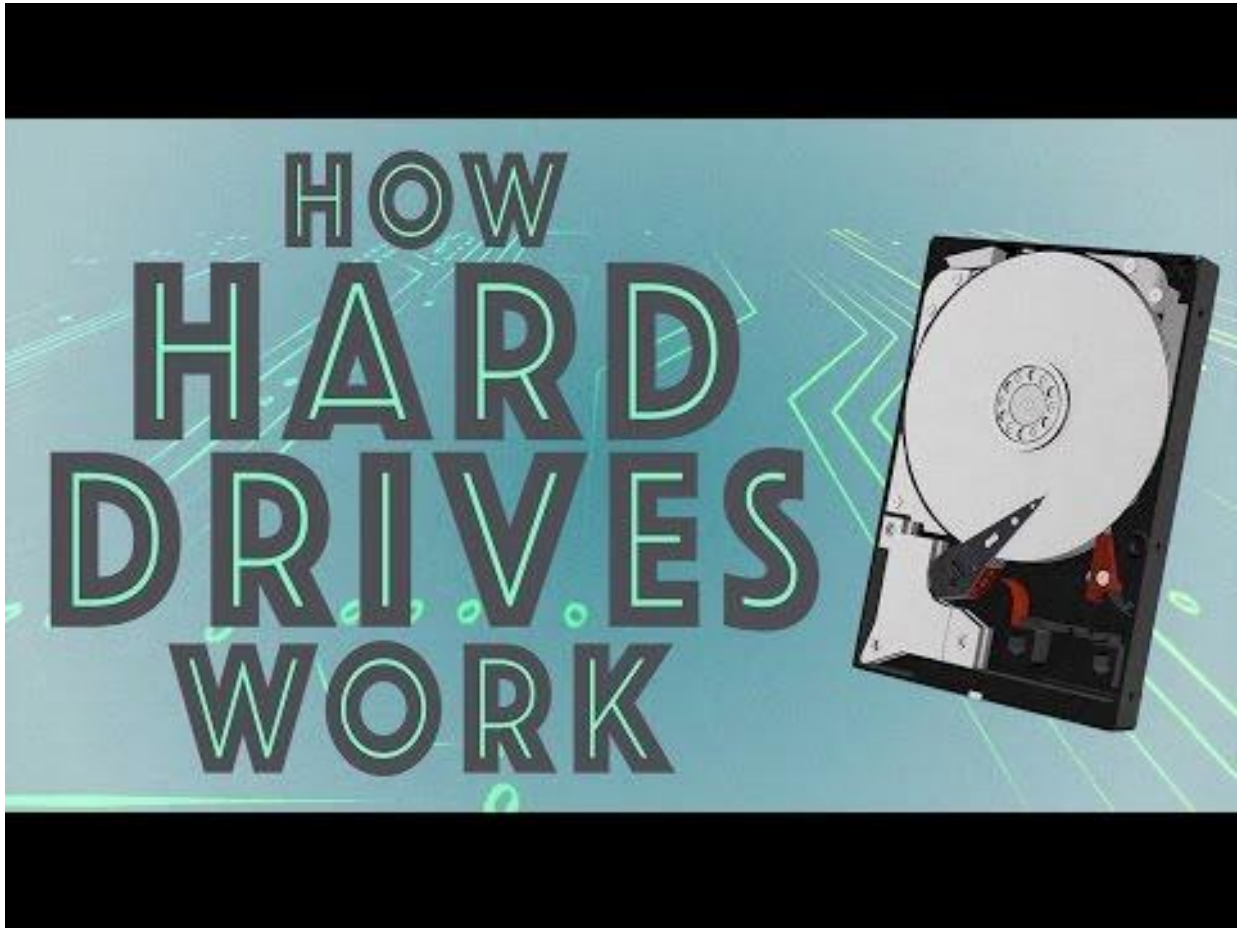


Componentes Disco Rígido

- Visto de cima



Componentes Disco



Operações no Disco Rígido

- Cabeça de Leitura e Gravação
  - Posicionada próxima a superfície do disco
  - Lê ou escreve dados codificados magneticamente
- Um disco:

  -100 mil trilhas por polegada

  -1 milhão de bits por polegada nas trilhas
- Os setores são indivisíveis
- Gaps podem representar 10% do total da trilha

Capacidade (Exemplo)

- Supondo um disco com:
  - 8 pratos
  - 2^{16} trilhas por superfície (65.536)
  - ≈ 2^{8} setores por trilha (256)
  - 2^{12} bytes por setor=4096
- Temos:
  - 16 superfícies com 65.536 trilhas cada e 256 setores por trilha. Cada setor com 4096 b
  - 1 Tb de capacidade. 1 trilha armazena ~1 Mb
  - Se um bloco utiliza 4 setores, temos 64 blocos/trilha

Tamanhos

- Trilha:
  - tamanho é uma característica do disco

    Não pode ser alterado
- Bloco:
  - Tamanho pode ser configurado quando o disco é inicializado
  - Deve ser um múltiplo do tamanho de 1 setor

Tempos de Acesso

- Procura
  - Tempo para mover as cabeças dos discos para a trilha na qual um bloco desejado está localizado.
- Rotação
  - Tempo para que o bloco desejado se posicione sob a cabeça do disco = meia rotação em média;
- Transferência
  - Tempo para ler ou escrever no bloco = tempo de rotação do disco sobre o bloco.

Tempos de Acesso

- Rotação 7.200 rpm (1 rotação em 8,33 ms)
- 65.536 trilhas
- A disco “gasta” 1 ms para atravessar 4.000 trilhas. Assim, a cabeça “atravessa” o disco em 16,38 ms (65.536/4.000)
- Gaps ocupam 10% do espaço de uma trilha
- 1 Bloco == 4 setores
- 1 trilha tem 256 setores

Solid State Disk (SSD)

Sem tempo de busca (seek time)

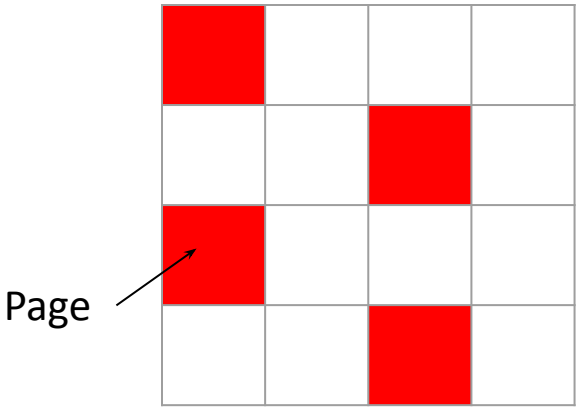
Sem latência de rotação (rotational latency)

Desafio

- Custo de escrita e leitura são diferentes

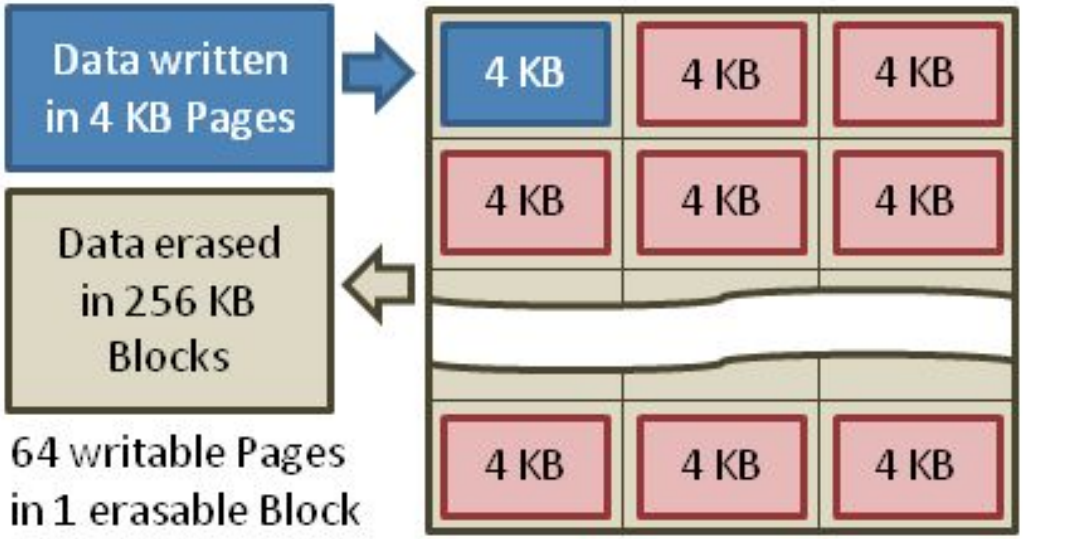
Escritas SSD

Block (256kb)



Operation	Area
Read	Page
Program (Write)	Page
Erase	**Block**

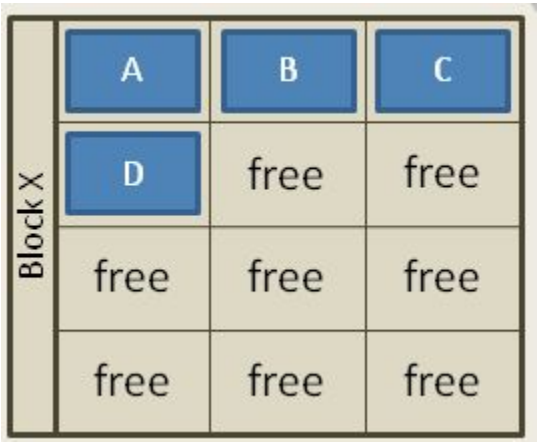
Escritas SSD



https://en.wikipedia.org/wiki/Solid-state_drive

Escritas no SSD

Write A, B, C, D

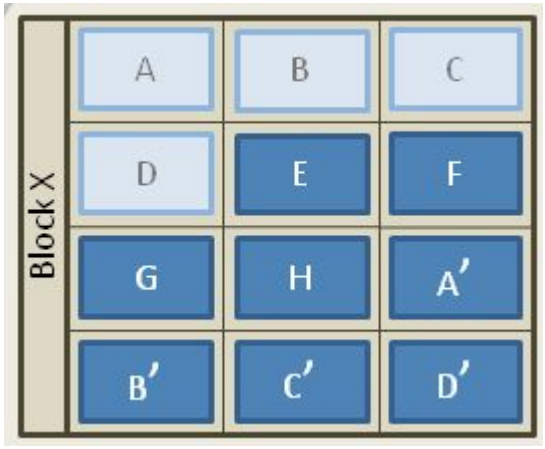


Escritas no SSD

Write A, B, C, D

Write E, F, G, H and

A’, B’, C’, D’



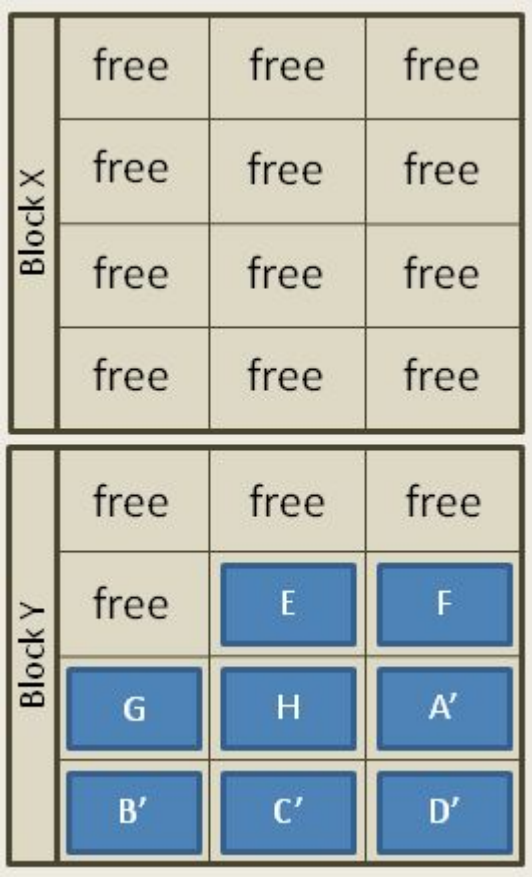
Escritas no SSD

Write A, B, C, D

Write E, F, G, H and

A’, B’, C’, D’

Página obsoletas são apagadas copiando o dado para um novo bloco



Atividade 1

1- Explique o motivo de uma escrita de um bloco no SSD ser mais lenta que uma leitura.

2- Você acredita que o disco rígido será descontinuado nos próximos anos? Explique.

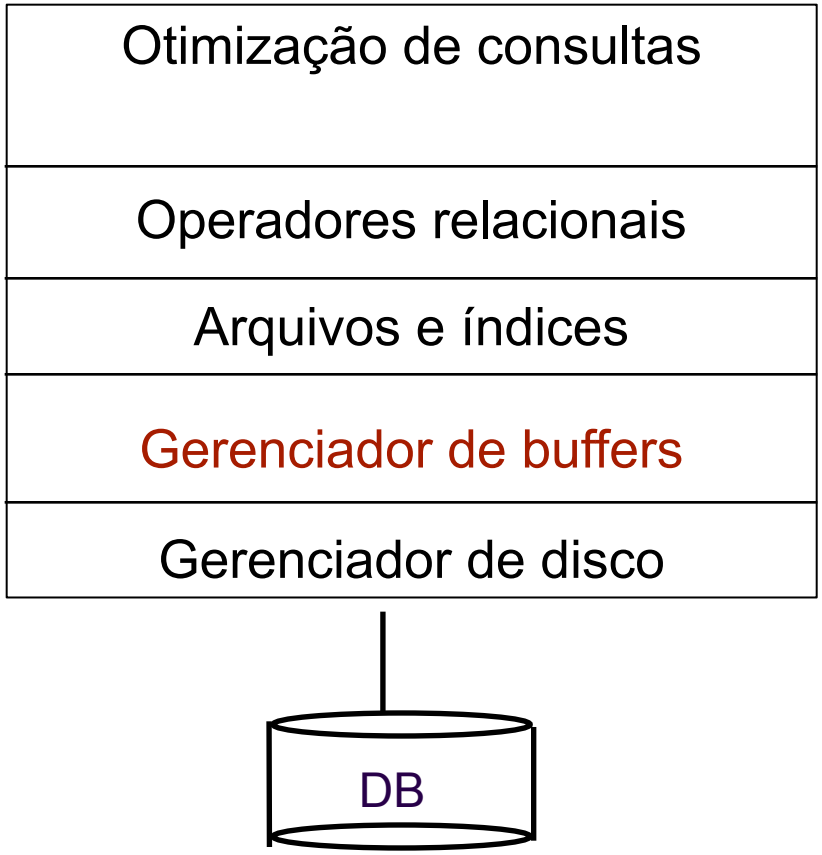
3- Explique como acontece uma operação de atualização no SSD vs HD.

4- Até então vimos na disciplina o sistema transacional, log e concorrência. Qual (ou quais) desses mecanismos você acha que será(ão) mais impactado(s) com a utilização de um disco de alta velocidade SSD? **Explique**

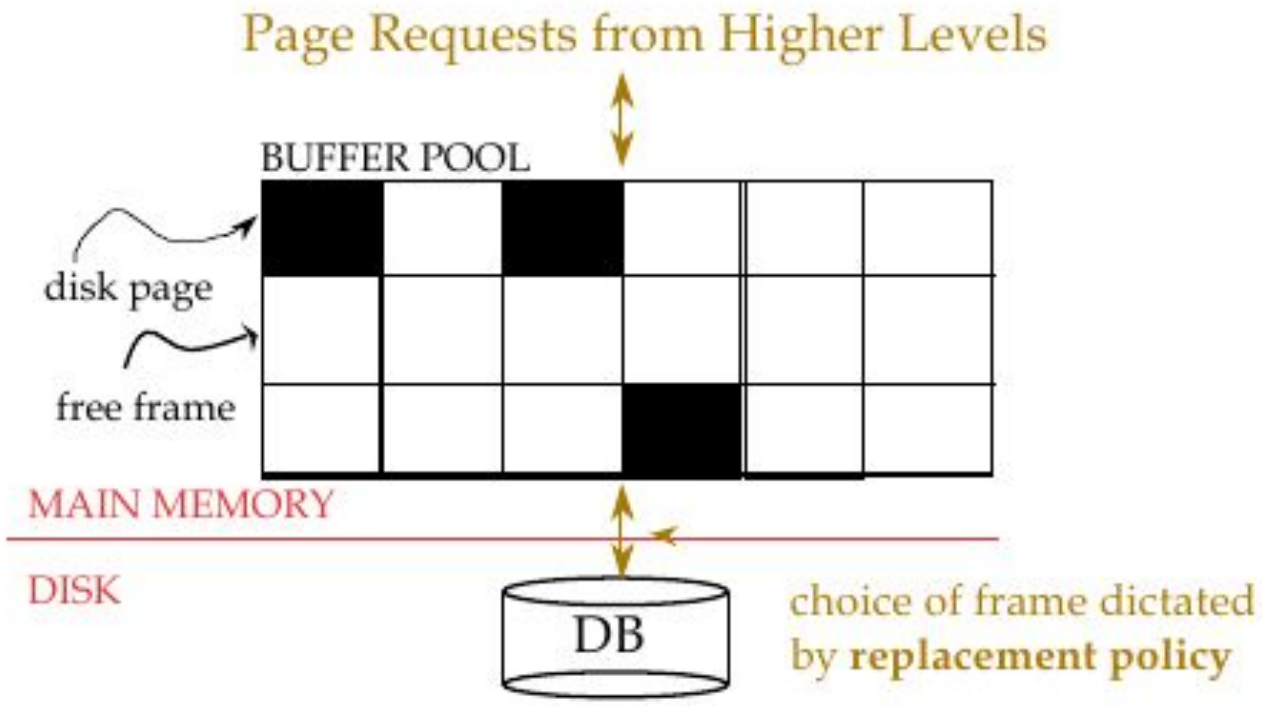
Atividade Prática!

```
create extension pg_freespacemap;
create table teste (id int4, test text);
insert into teste values(generate_series(1,10), 'hello world!');
select ctid from teste;
select pg_relation_filepath('teste');
select * from pg_freespace('teste');
sudo hexdump -C  main/base/
delete from teste id < 50;
vacuum teste;
```

Banco de dados relacional



Gerenciador de buffers



[Ramakrishna et. al. 3 edição, pag 264]

Otimização do Acesso

Buffer: porção da memória principal que armazena cópias dos blocos do disco.

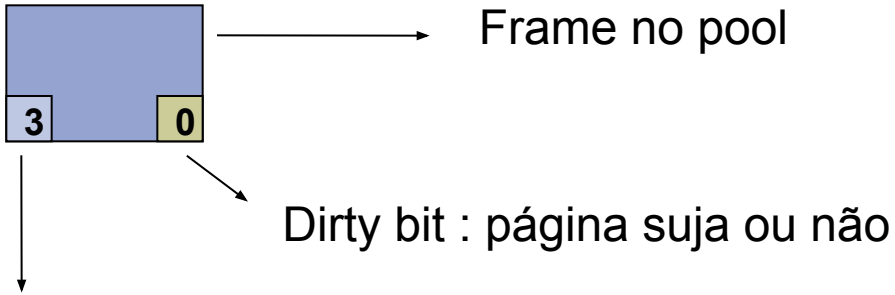
- O buffer é organizado em páginas que, geralmente, se relacionam 1:1 com os blocos

Gerenciador de buffer: subsistema de um SGBD responsável para gerenciar espaços no buffer

Gerenciador de Buffer

- Testa se o dado procurado está no buffer
- Traz a página do disco para a memória
- Procura frames livres (espaço memória) para alocar a página
- Aciona algoritmo para liberar a página
- Aloca página
- Caso o frame tiver que ser reutilizado, propaga modificação no disco.

Algoritmo de Gerenciamento



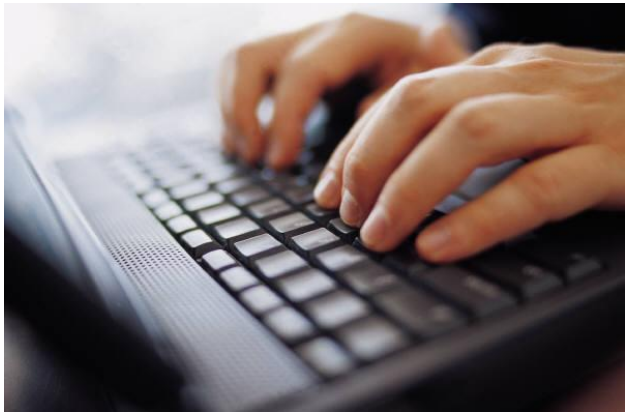
Pin-count = número de vezes que a página foi solicitada para consultas ou modificações mas não foi liberada ainda.

Inicialmente :

Dirty bit := 0

Pin-count := 0

Considere o seguinte cenário



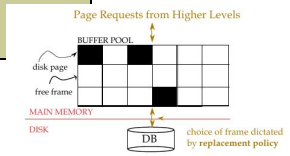
Usuário consulta banco de dados

SELECT *
From EMP
WHERE EMP.CPF = 40333994598

Processador de Consultas

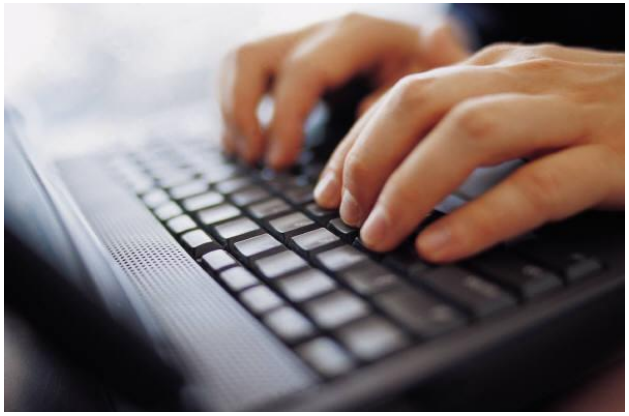
Solicita dados
para consulta (leitura)

Gerenciador de Buffer



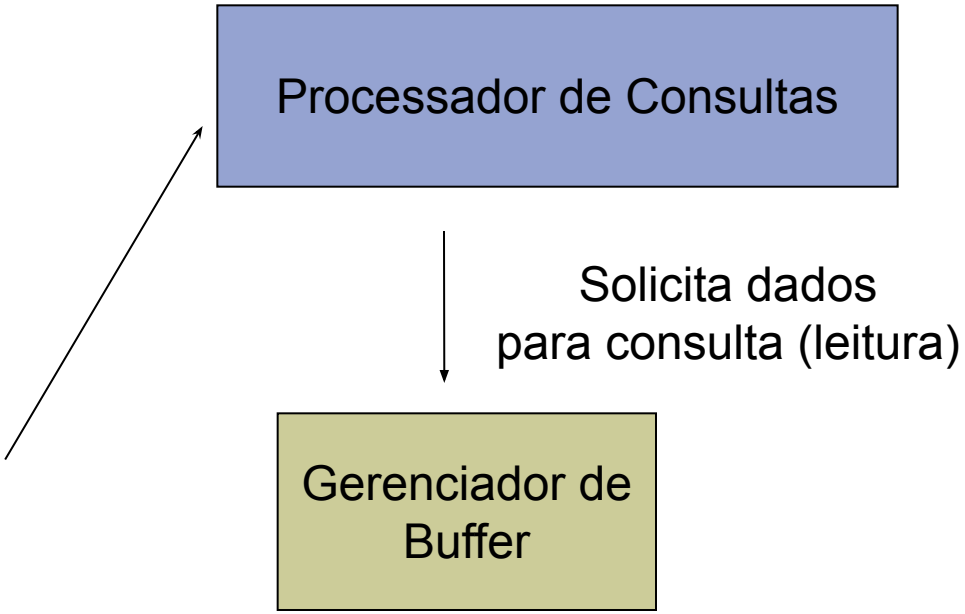
28

Considere o seguinte cenário



Usuário consulta banco de dados

SELECT *
From EMP
WHERE EMP.CPF = 40333994598



Está na memória? Não,
vou buscar no disco
Problema: não tem
espaço na memória

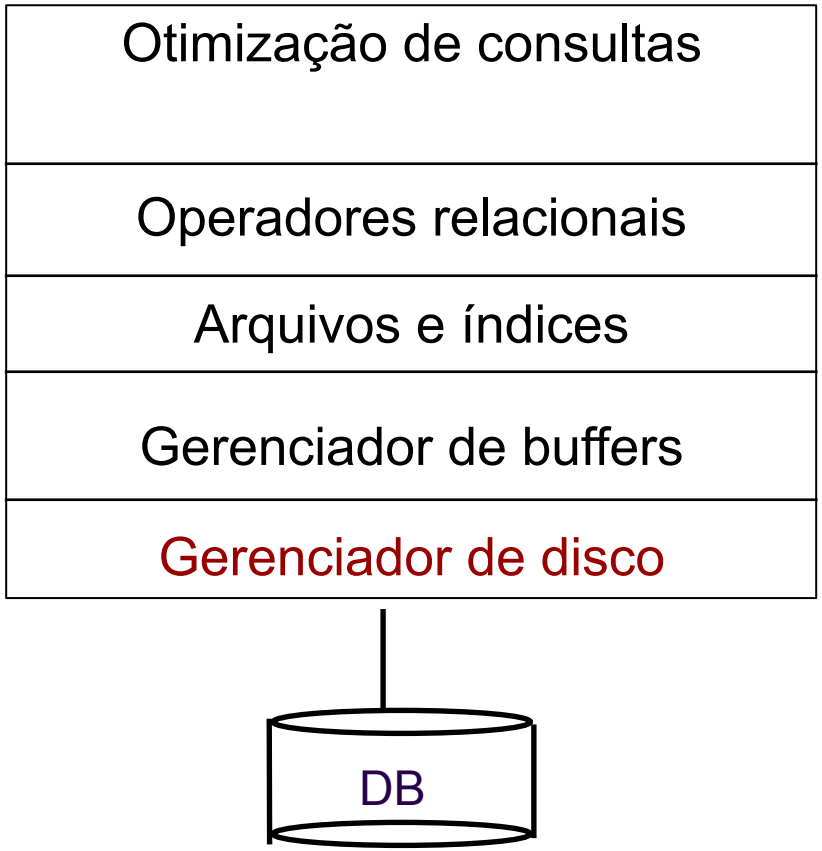
Considere o seguinte cenário

- Verifica se existem frames com *pin-count* = 0
- **Caso positivo**: aciona gerenciador de disco
  - Gerenciador de disco posiciona cabeça de leitura sobre o endereço da primeira página do arquivo EMP (conhecido do gerenciador de buffer)
  - Gerenciador de disco providencia a transferência da página.
  - Gerenciador de buffer vai alocar a página em um frame com *pin-count* = 0
  - Qual frame será escolhido ?
    - Usa sua política de substituição (LRU, MRU, random)
  - Verifica o dirty bit deste frame

Considere o seguinte cenário

- **Dirty bit = 1**:
  - grava a página atual do frame no disco
  - Aloca a nova página no frame
  - Incrementa *pinout* do frame
  - Retorna o endereço do frame para o processador de consultas
- **Dirty bit = 0**:
  - Aloca a nova página no frame
  - Incrementa *pinout* do frame
  - Retorna o endereço do frame para o processador de consultas

Banco de dados relacional



Arquivos

Blocos são transmitidos entre disco e memória, mas...

O SGBD trabalha no nível de registro e arquivos

Arquivo: uma coleção de páginas, que contém um conjunto de registros. Estes devem suportar:

- Inserção/remoção/atualização
- Busca de registro em particular
- Busca de todos os registros

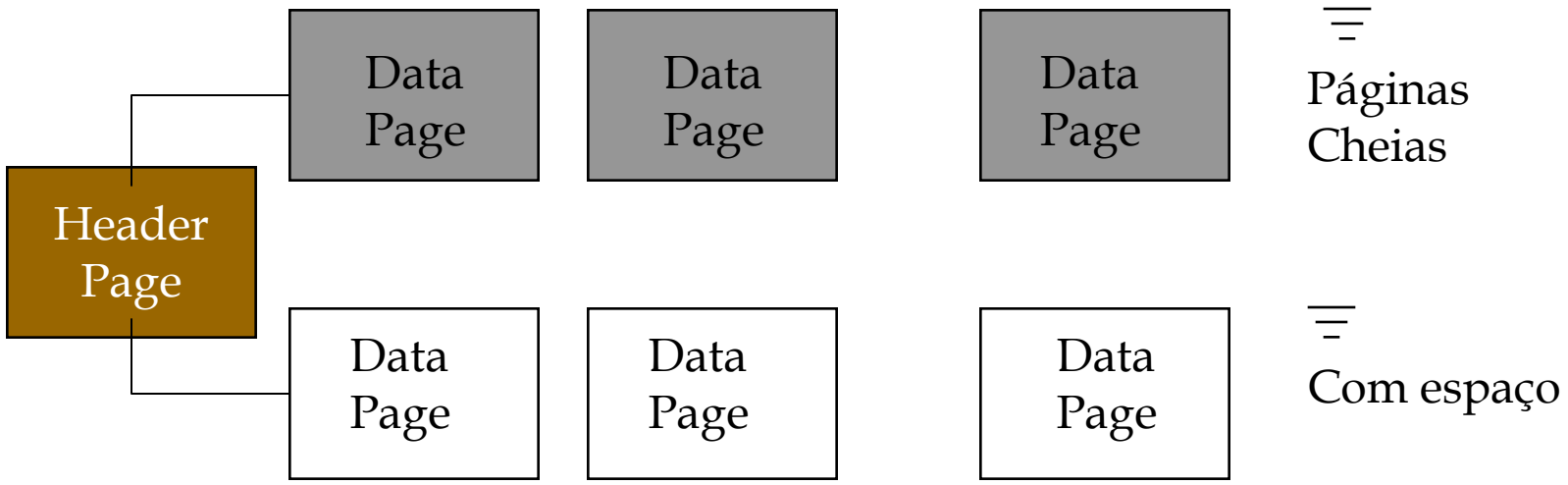
Arquivos não ordenados - Heap

Estrutura mais simples de armazenar registros sem ordenamento

Para suportar as operações, é importante:

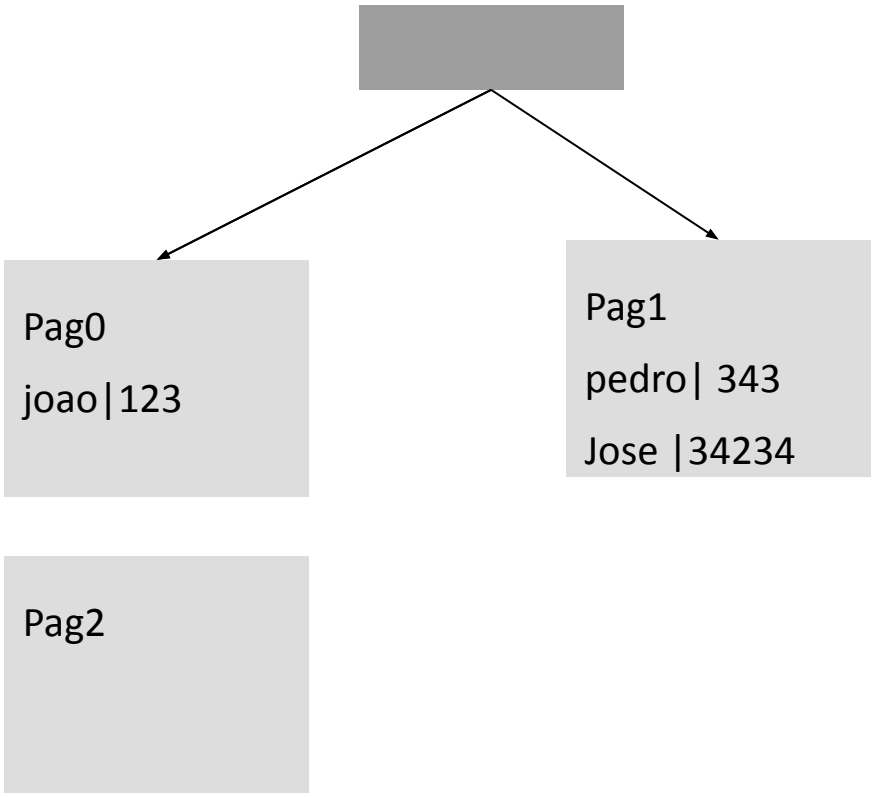
- Manter o rastro das páginas no arquivos
- Manter o rastro das páginas vazias
- Manter o rastro dos registros em cada página

Arquivos não ordenados - Heap

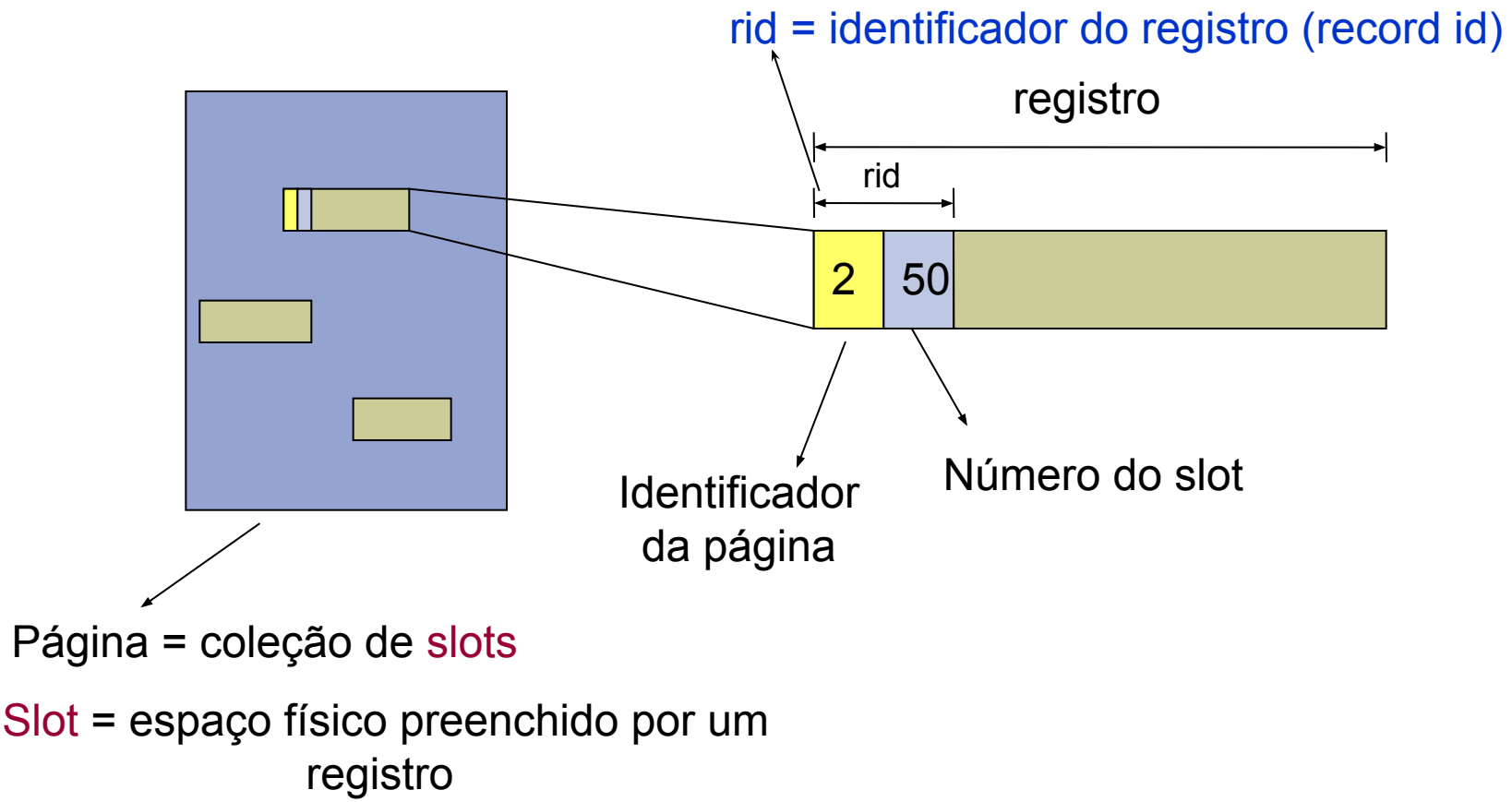


O ponteiro da página inicial (header page) deve ser armazenado;

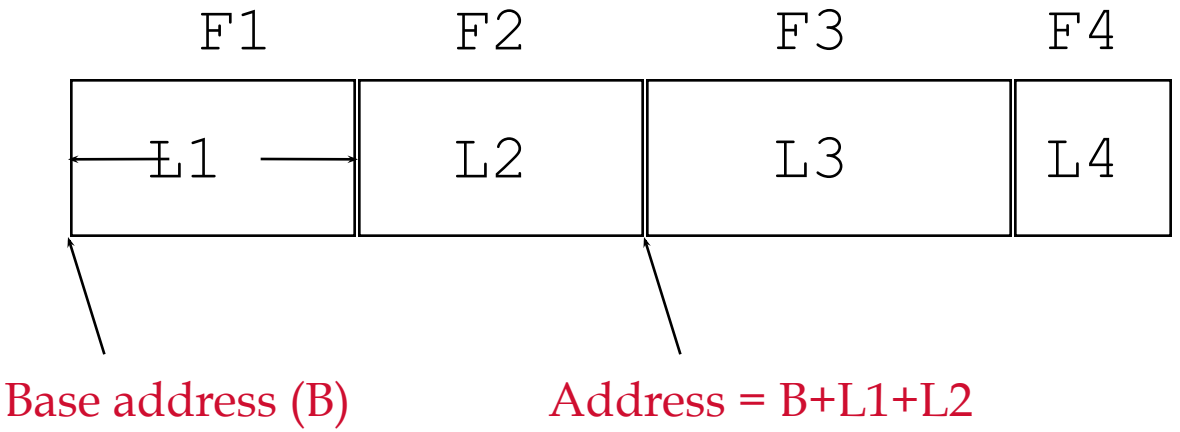
Arquivos não ordenados - Heap



Organização dos registros no disco

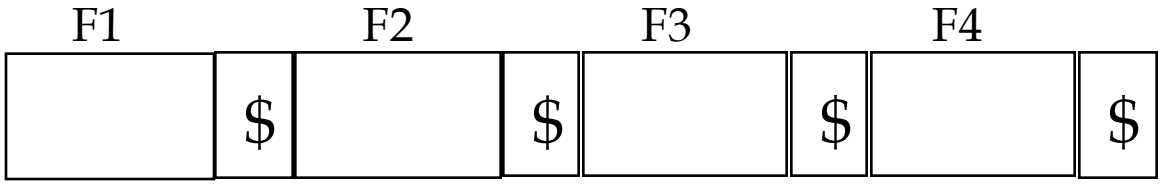


Formato dos registros no disco

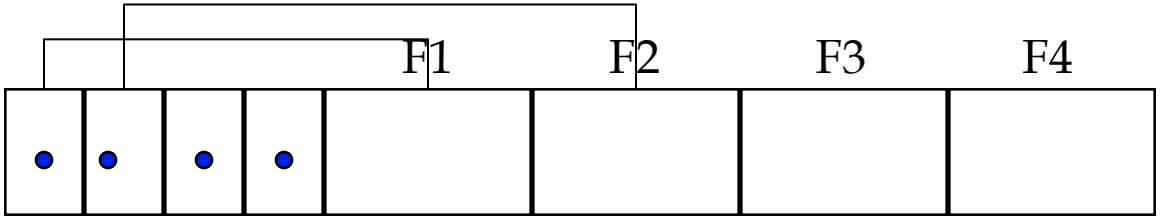


Formato dos registros no disco

Duas alternativas para tamanhos variados

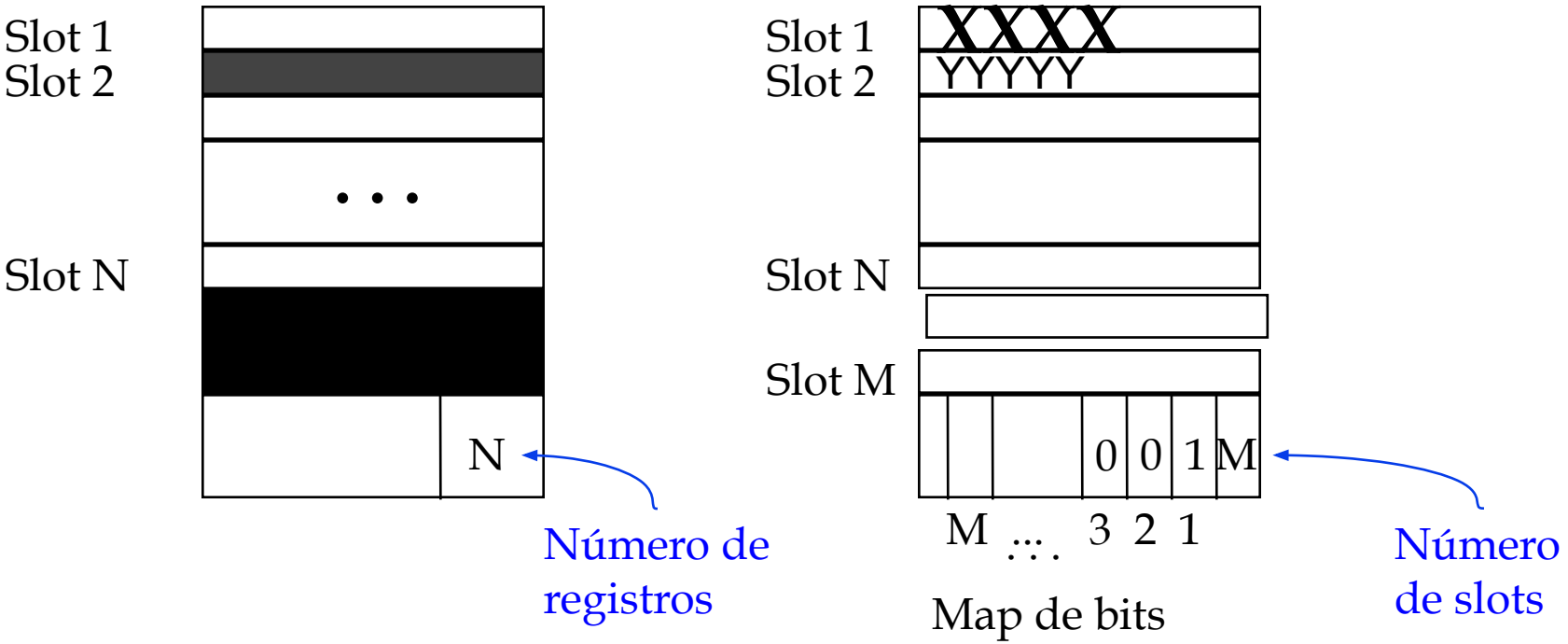


Campos determinados por caractere especial



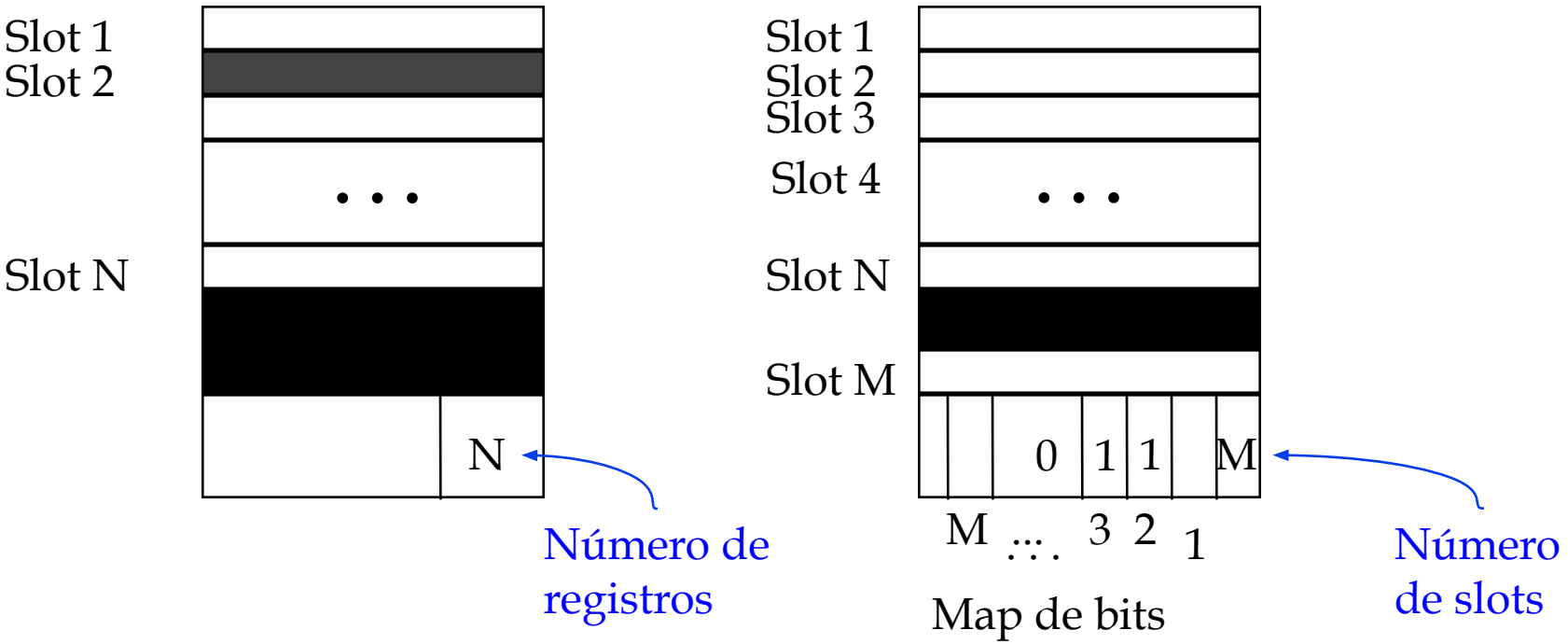
Array de offsets

Páginas com formato fixo



Record id = <page id, slot #>.

Páginas com formato fixo



Record id = <page id, slot #>.

Atividade A

create table Movie(name char(30), address varchar (255), data date*)

*date ocupa 10 bytes

Exemplo: Suponha que os registros da tabela Movie serão armazenados em páginas de 4kb. O cabeçalho do registro ocupa 12 bytes (ponteiro para o esquema, tamanho registro, timestamp). Quantos registros cabem na página (sem o mapa de bits)?

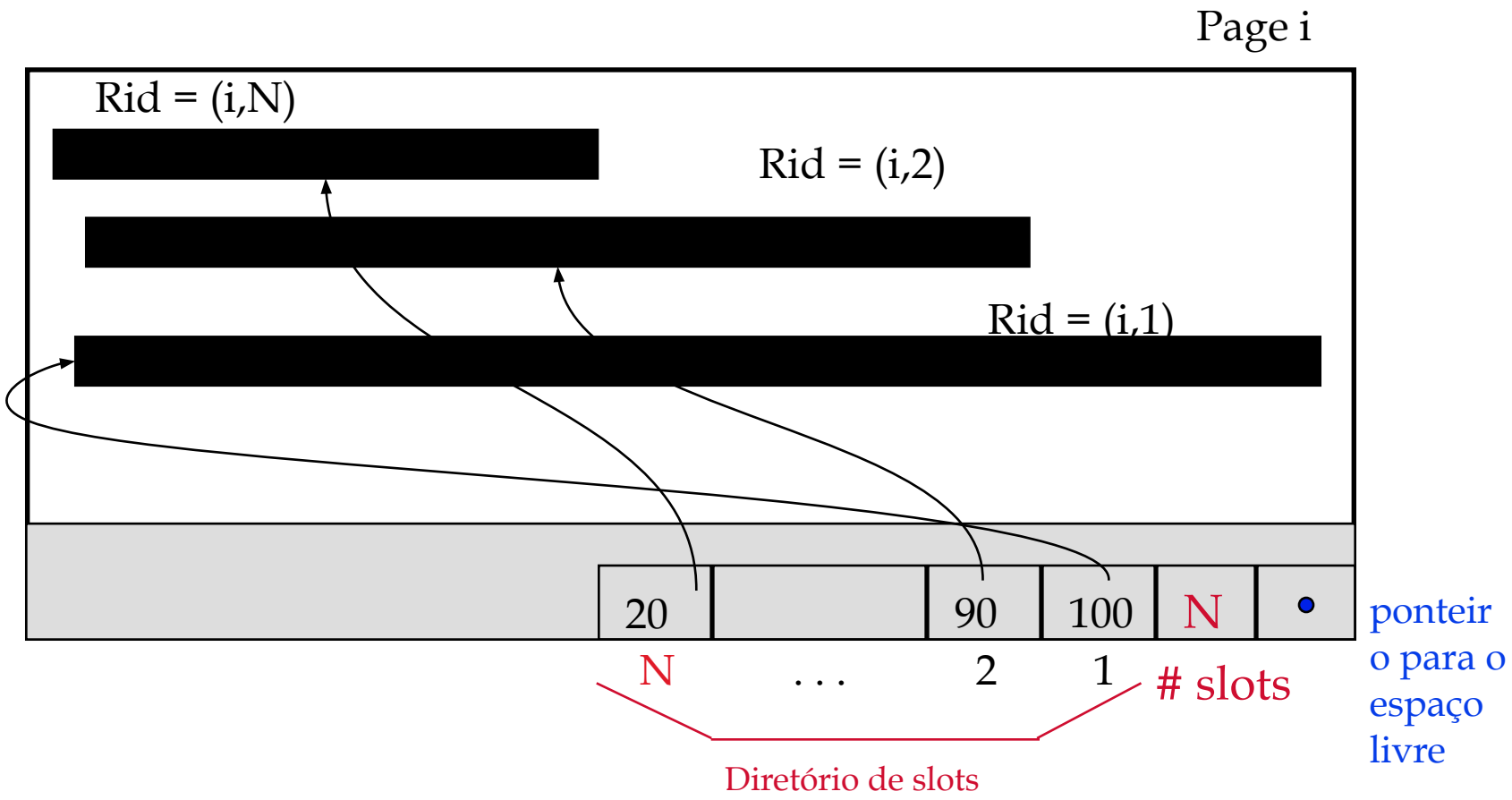
Atividade B

A)Construa um diagrama de página de tamanho fixo usando um mapa de bits. A tabela armazena 20 registros de 12 bytes cada.

B) Insira 3 registros

C) Apague o registro 0 e 2 previamente inseridos

Registros com tamanhos variados

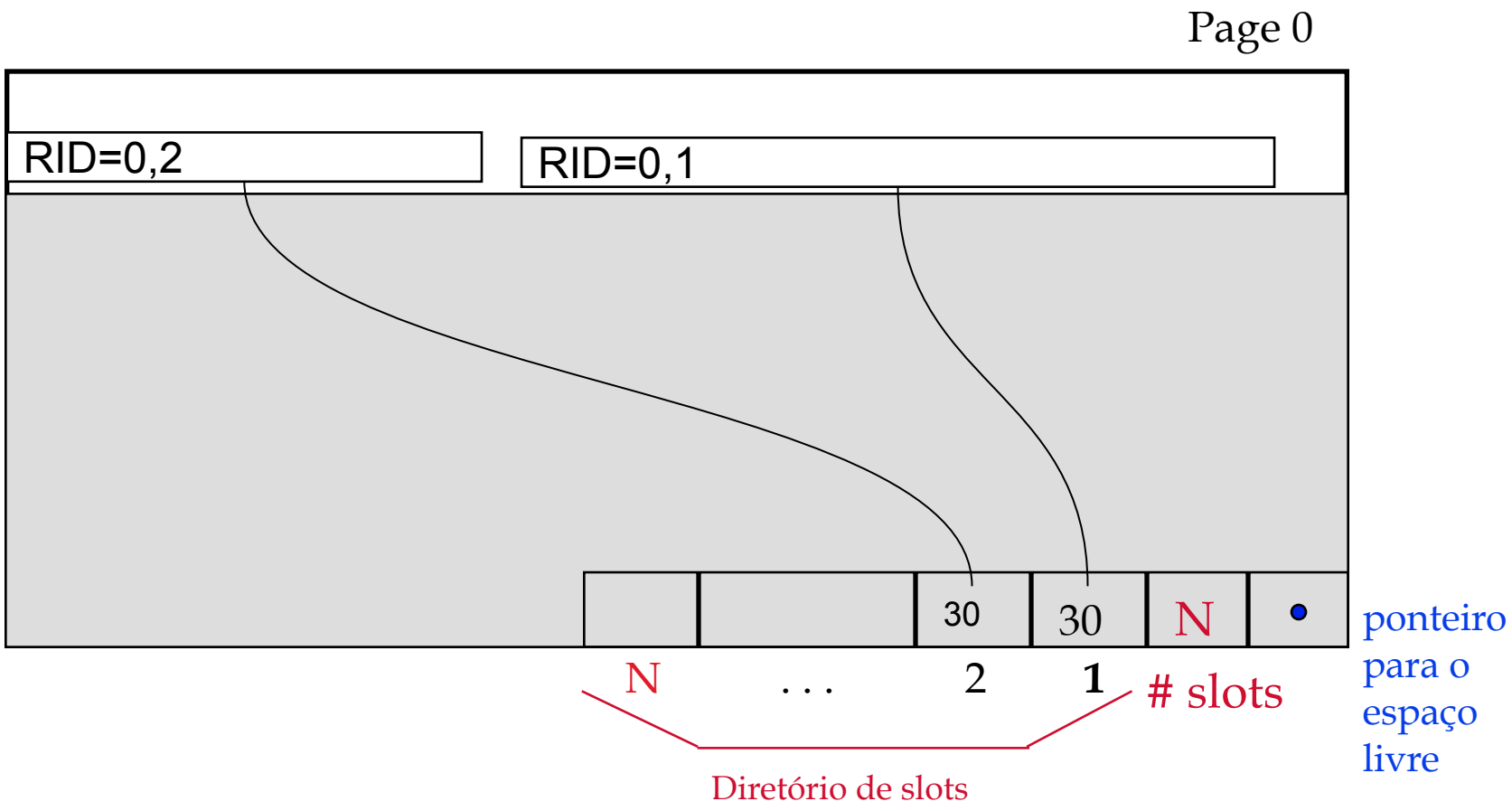


- *É possível mover registros sem alterar o RID*

Exemplo

O que acontece na Atividade B, em relação ao espaço ocupado pelo registros, se o atributo “nome” fosse alterado para varchar(255)?

Registros com tamanhos variados



Atividade C

Insira os registros abaixo em uma página com tamanho variado e monte o diagrama de página. Os registros tem 200 bytes no máximo cada. Tamanho total da página 4kb.

Create table teste (varchar[200] x)

1) Inserir : “A”, “BBBBBBBB” , “DDDDDDDDDDDDDDD"
2) Demonstre o que acontece caso o primeiro registro seja alterado para “ABXXXXX”
3) Demonstre o que acontece caso os registros terem seu tamanho alterado para varchar[300].

Atividade D - Entrega até 01/02 neste slide

Create table teste (varchar[30] x, int y)

Monte o diagrama do registro conforme abaixo:

A) Insira 3 registros na tabela acima.

“a”,1 ; “b”,2; “c”,3

B) Remova o primeiro registro

C) Insira 1 novo registro.

“x”,4

Atividade E - Entrega até 01/02 neste slide

Create table teste (varchar[30] nome, int idade, varchar[50] end)

Monte o diagrama do registro conforme abaixo:

A) Insira 3 registros na tabela acima.

‘joao’,40, ‘rua getulio vargas 21’ ;

‘maria’,45, ‘rua porto alegre N 35E’ ;

‘pedro’,30,’rua getulio vargas 210’ ;

B) Remova o segundo registro

C) Insira 1 novo registro.

“joao pedro”,40,’rua italia 1’;

The Oversized-Attribute Storage Technique- TOAST

- Postgres não permite fragmentar registros
- Página possuem tamanho de 8kb
- Registro possuem tamanho máximo de 2kb

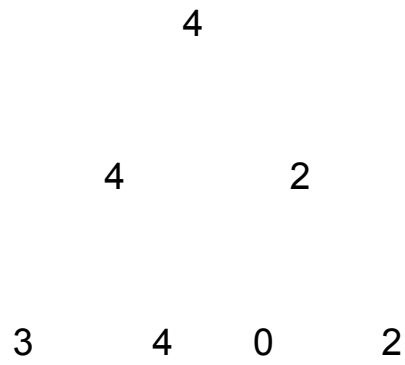
Possibilidade de fragmentar colunas (toast-The Oversized Attribute Storage Technique)

Heaps- postgres

Como encontrar um espaço nas páginas para armazenar um registro?

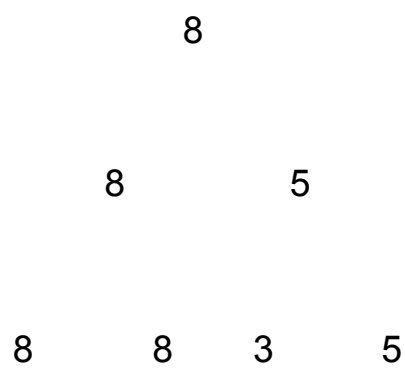
Heaps- postgres

Como encontrar um espaço nas páginas para armazenar um registro?



Atividade

1-Na árvore abaixo, é possível encontrar 10 unidades de espaço disponíveis? Explique



2) Atualize a árvore acima após a alocação de 6 novas unidades de espaço.

Índices (preview)

Um heap file permite recuperar registros:

- A partir de um RID ou Busca sequencial por todos os registros

Algumas vezes precisamos recuperar registros em alguma ordem:

- ÍNDICES!!!