



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Informática
Integrado / Análise e Desenvolvimento de Sistemas / Licenciatura em Computação

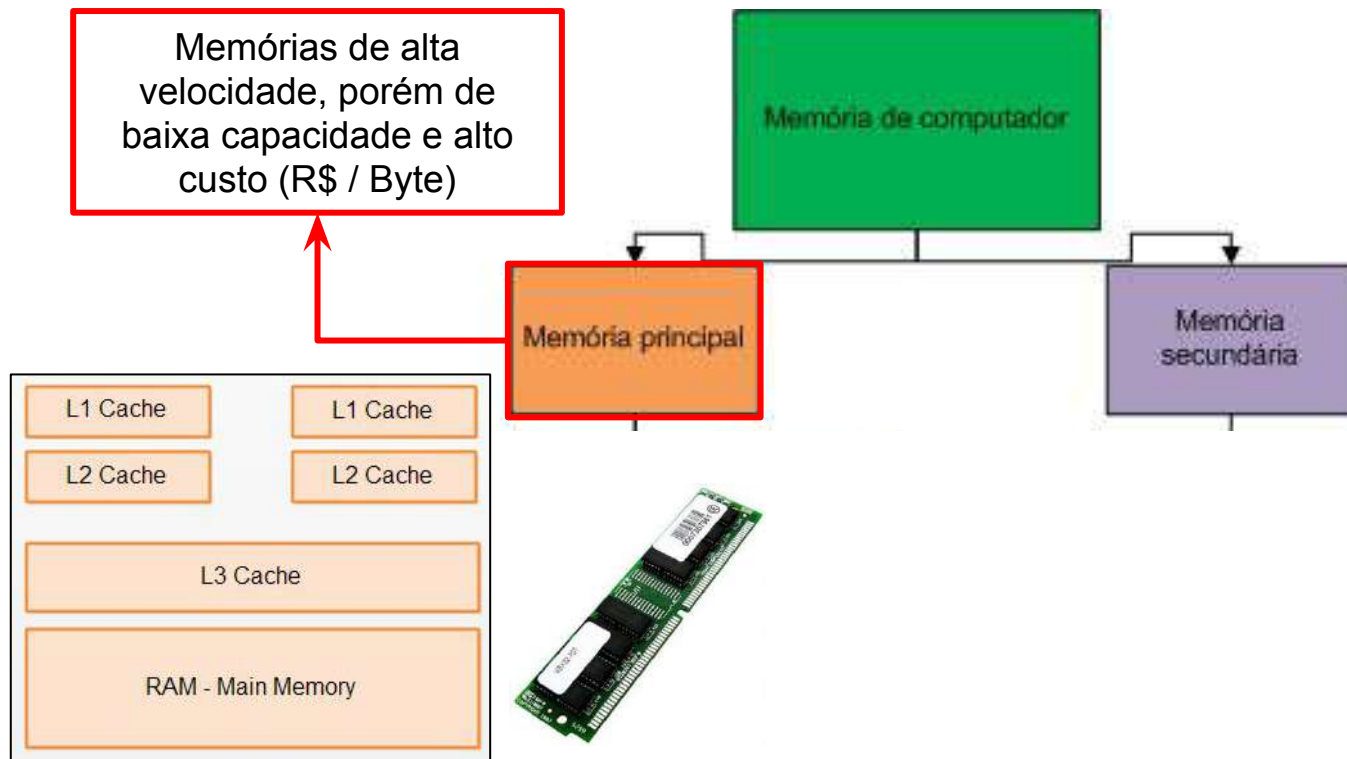
Estruturas de Arquivos em Bancos de Dados

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

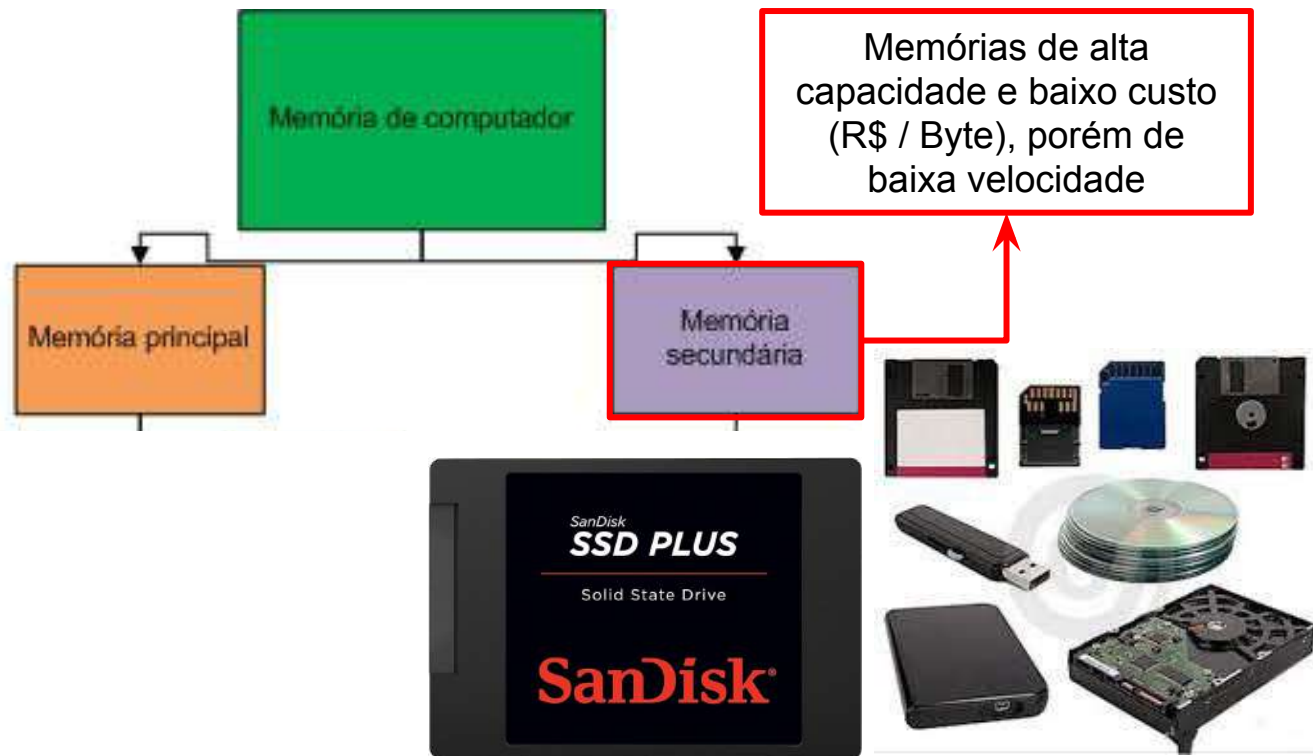
Armazenamento de dados em disco

- O armazenamento de registros das tabelas de um banco de dados é feito em arquivos
- O armazenamento de dados segue a hierarquia de memória, definida pelo sistema operacional:
 - **Armazenamento primário (memória principal)**
 - **Armazenamento secundário (memória secundária)**

Hierarquia de Memórias

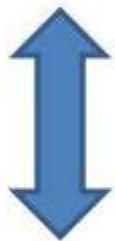


Hierarquia de Memórias

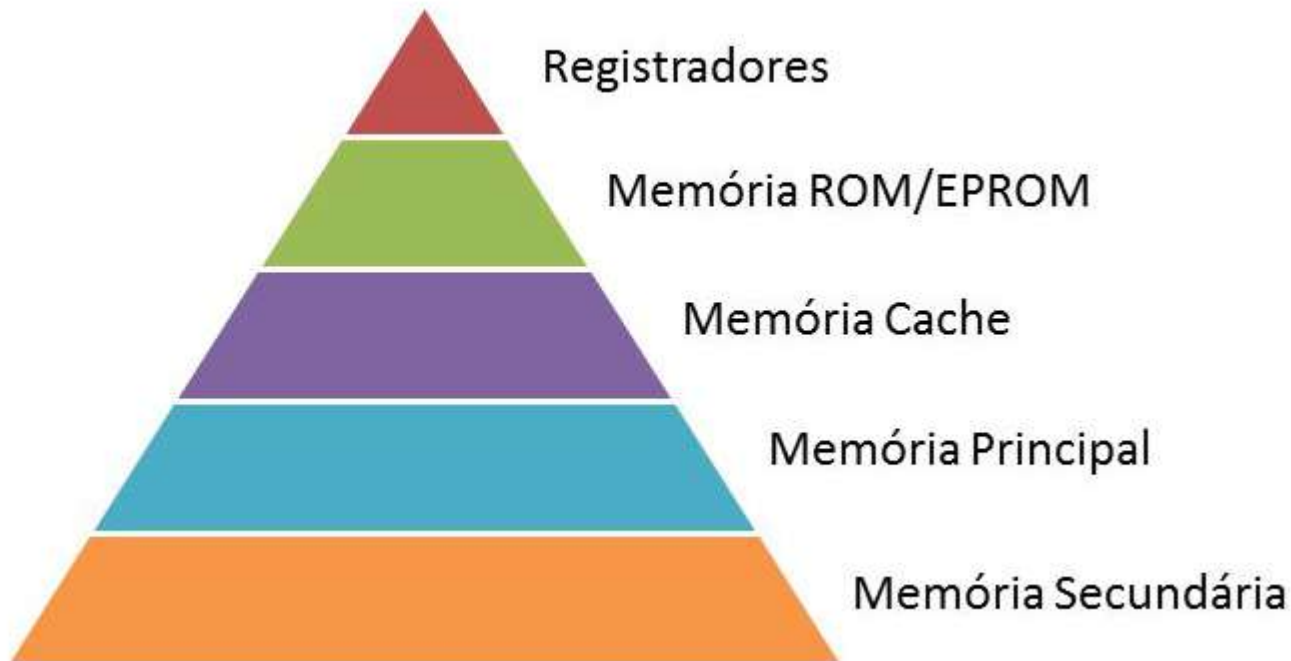


Hierarquia de Memórias

Custo alto
Velocidade alta
Capacidade baixa

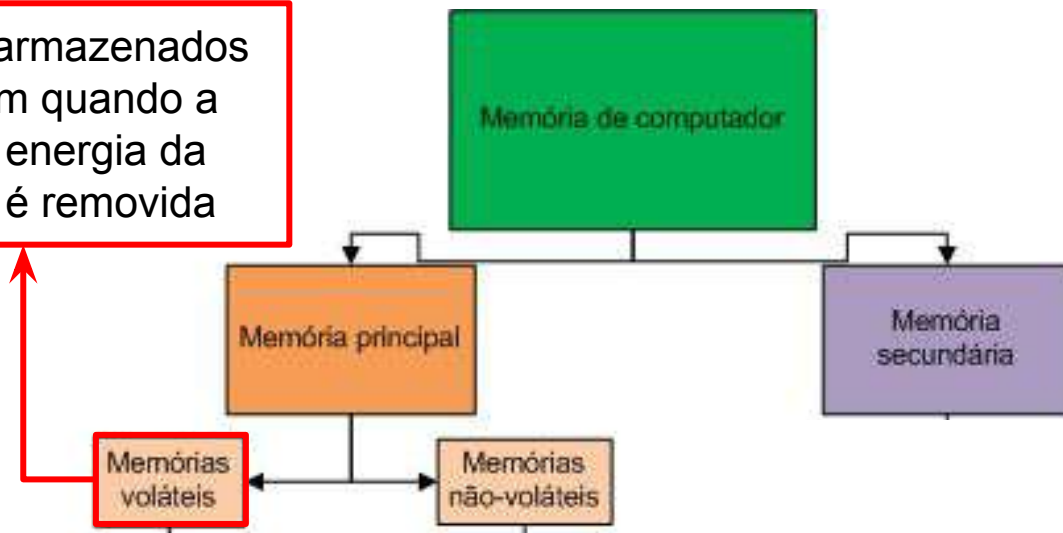


Custo baixo
Velocidade baixa
Capacidade alta

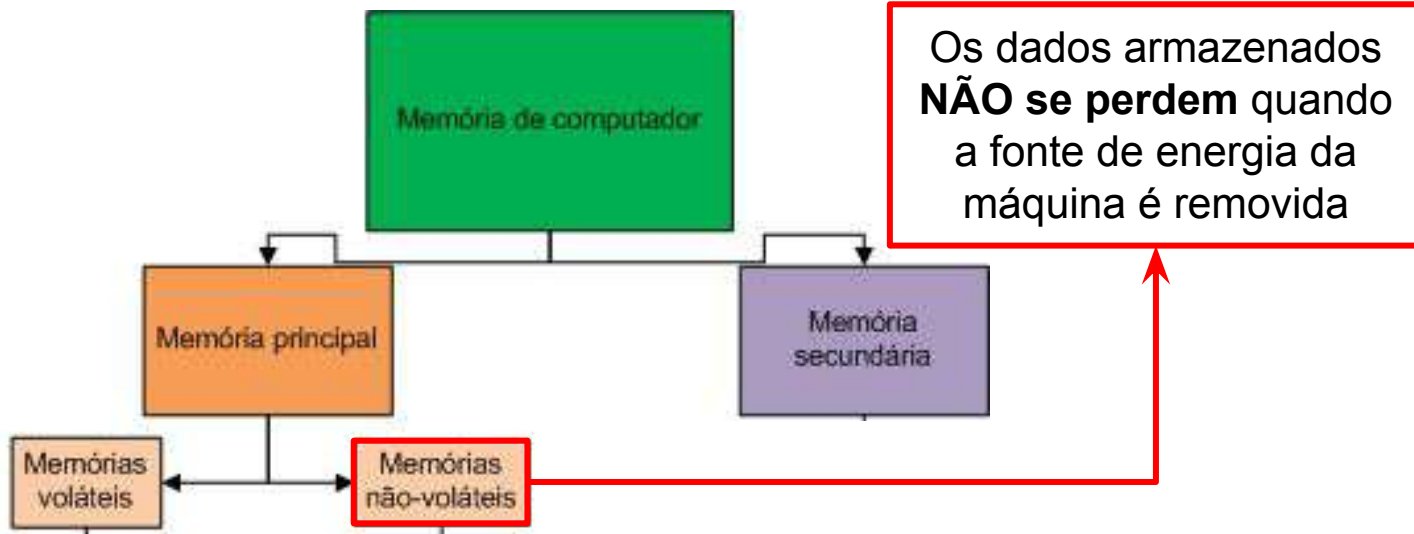


Hierarquia de Memórias

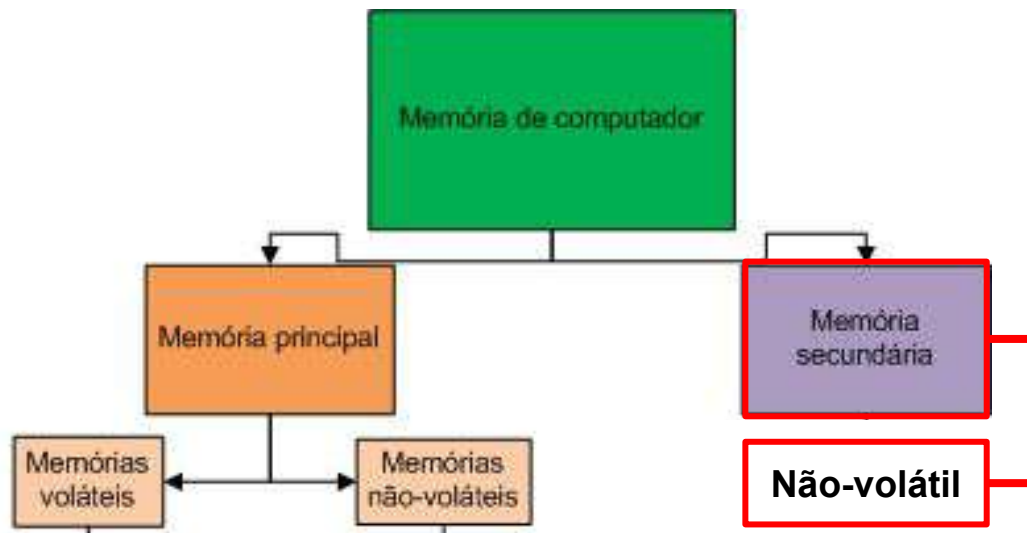
Os dados armazenados se perdem quando a fonte de energia da máquina é removida



Hierarquia de Memórias



Hierarquia de Memórias



Os dados de um banco de dados são armazenados nas memórias secundárias, que são memórias não-voláteis de alta capacidade.

Porque?

Porque armazenar Dados na Memória Secundária ?

- Bancos de dados armazenam grandes volumes de dados
 - Não é possível armazenar um banco de dados inteiro dentro da memória principal, dada a sua capacidade limitada
 - Custo de armazenamento de dados na memória principal é maior do que na memória secundária

Porque armazenar Dados na Memória Secundária ?

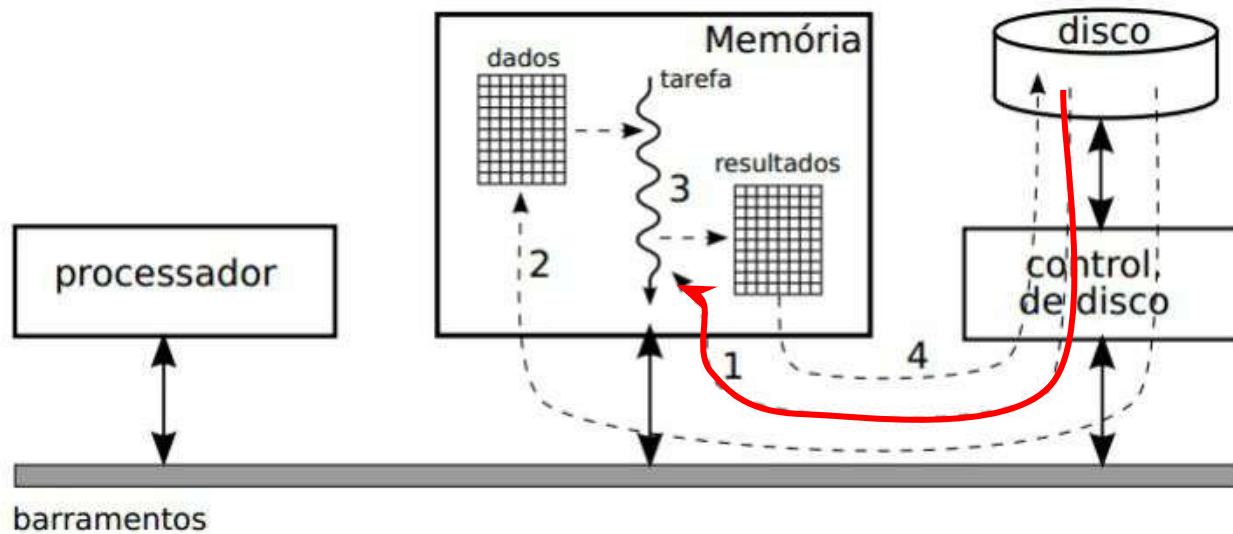
- Bancos de dados precisam armazenar dados de maneira permanente (SGBDs precisam de memórias não-voláteis)
- Perda de dados é menos frequente em memórias não-voláteis, como as memórias secundárias

Armazenamento de Dados em Bancos de Dados

- **Problema:** Acessar registros do banco de dados diretamente da memória secundária pode comprometer o desempenho do sistema
 - **Fato:** Aplicações acessam apenas uma pequena parte do bancos de dados em um dado instante de tempo
 - **Solução:** Transferir os dados da memória secundária para a principal, e vice-versa, conforme for necessário

Armazenamento de Dados em Bancos de Dados

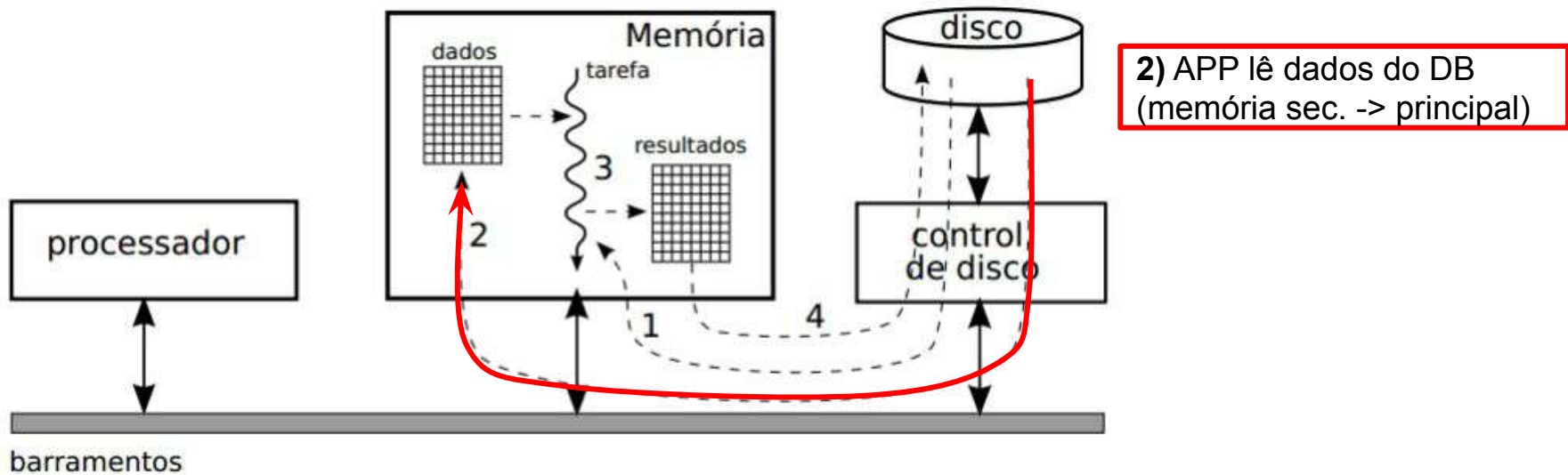
- **Solução:** Transferir os dados da memória secundária para a principal, e vice-versa, conforme for necessário



1) Carregar a aplicação

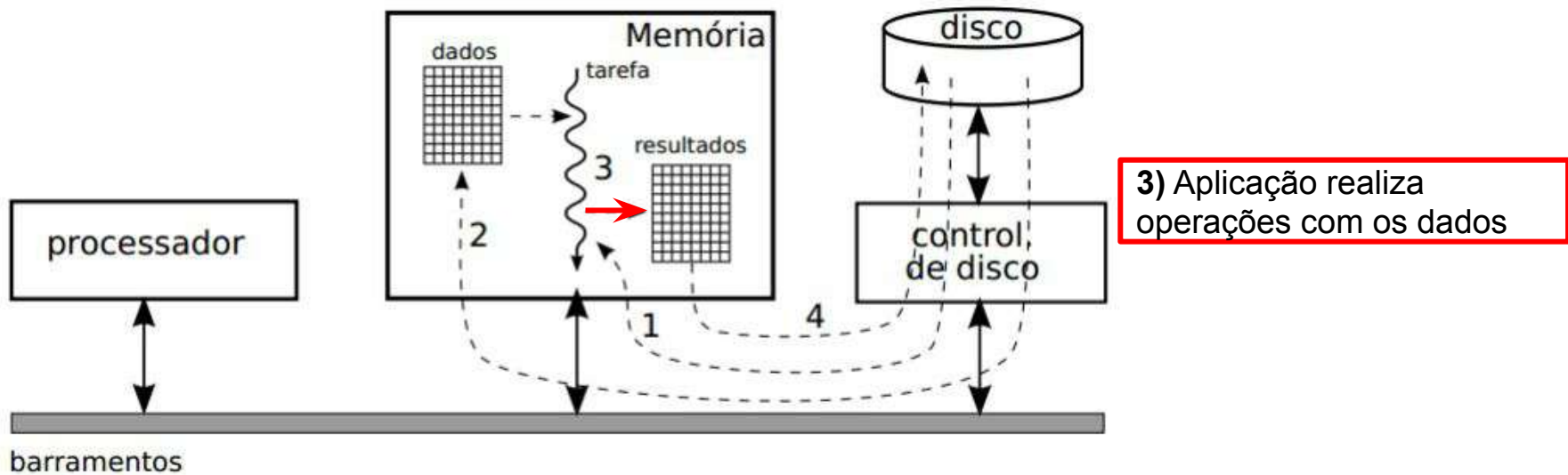
Armazenamento de Dados em Bancos de Dados

- **Solução:** Transferir os dados da memória secundária para a principal, e vice-versa, conforme for necessário



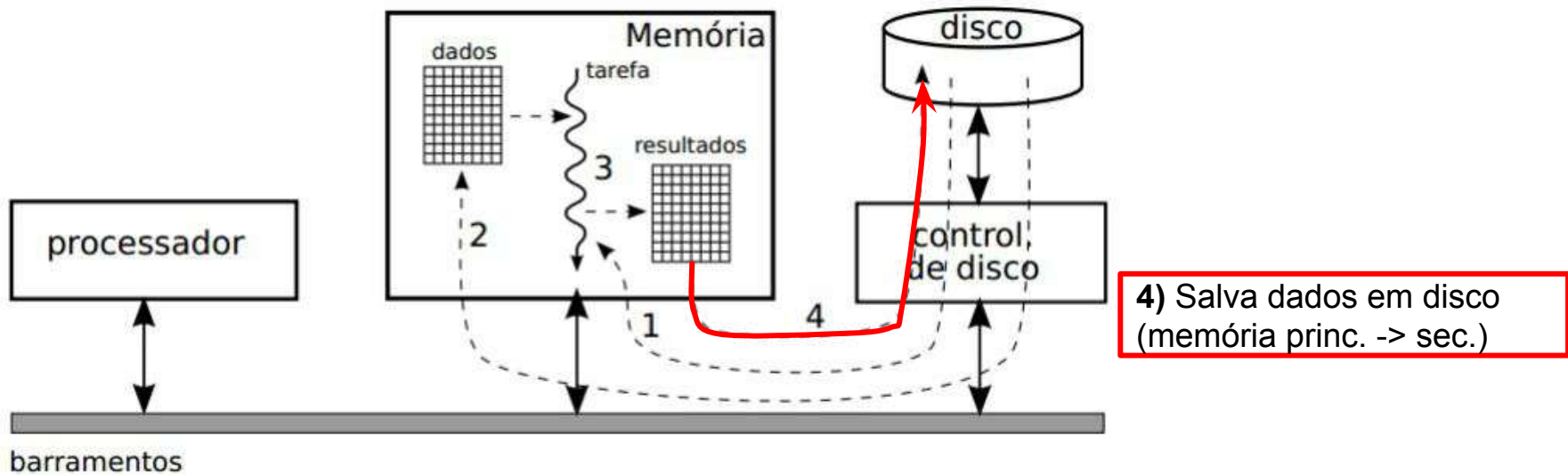
Armazenamento de Dados em Bancos de Dados

- **Solução:** Transferir os dados da memória secundária para a principal, e vice-versa, conforme for necessário



Armazenamento de Dados em Bancos de Dados

- **Solução:** Transferir os dados da memória secundária para a principal, e vice-versa, conforme for necessário



Transferência de Dados em um HDD

- A transferência de dados entre memórias (principal e secundária) é feita em **blocos**
 - Toda vez que uma transferência de dados é iniciada, as seguintes tarefas são executadas:
 - Posicionar o cabeçote de leitura/gravação do HDD sobre a faixa / cilindro correto do HDD
 - Girar o HDD para o setor correto
 - Realizar a leitura/gravação usando cabeçote magnético

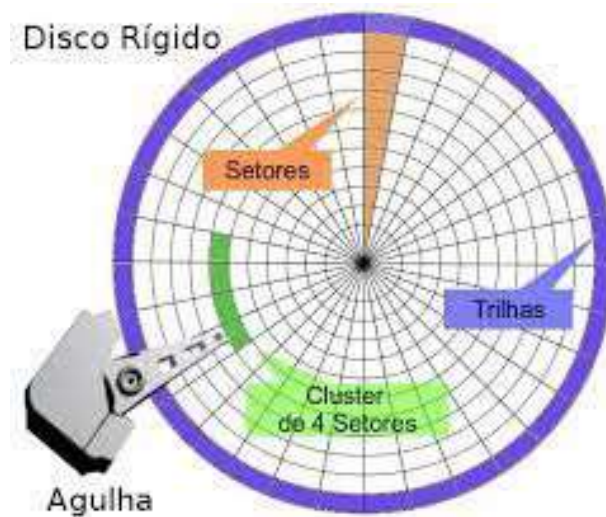
Transferência de Dados em um HDD

- A transferência de dados entre memórias (principal e secundária) é feita em **blocos**
 - Toda vez que uma transferência de dados é iniciada, as seguintes tarefas são executadas:
 - Posicionar o cabeçote de leitura/gravação do HDD sobre a faixa / cilindro correto do HDD
 - Girar o HDD para o setor correto
 - Realizar a leitura/gravação usando cabeçote magnético

A tarefa de transferir dados demanda tempo. Logo, **transferir blocos de dados é mais eficiente** do que transferir poucos bytes isoladamente.

Transferência de Dados em um HDD

- Posicionar o cabeçote de leitura/gravação do HDD sobre a faixa / cilindro correto do HDD
- Girar o HDD para o setor correto
- Realizar a leitura/gravação usando cabeçote magnético



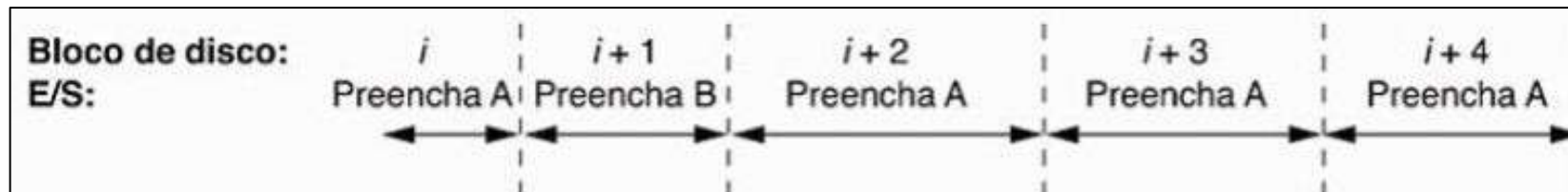
Transferência de Dados em um HDD

- Posicionar o cabeçote de leitura/gravação do HDD sobre a faixa / cilindro correto do HDD
- Girar o HDD para o setor correto
- Realizar a leitura/gravação usando cabeçote magnético

Será que é possível melhorar o desempenho da leitura/gravação de dados em uma memória secundária?

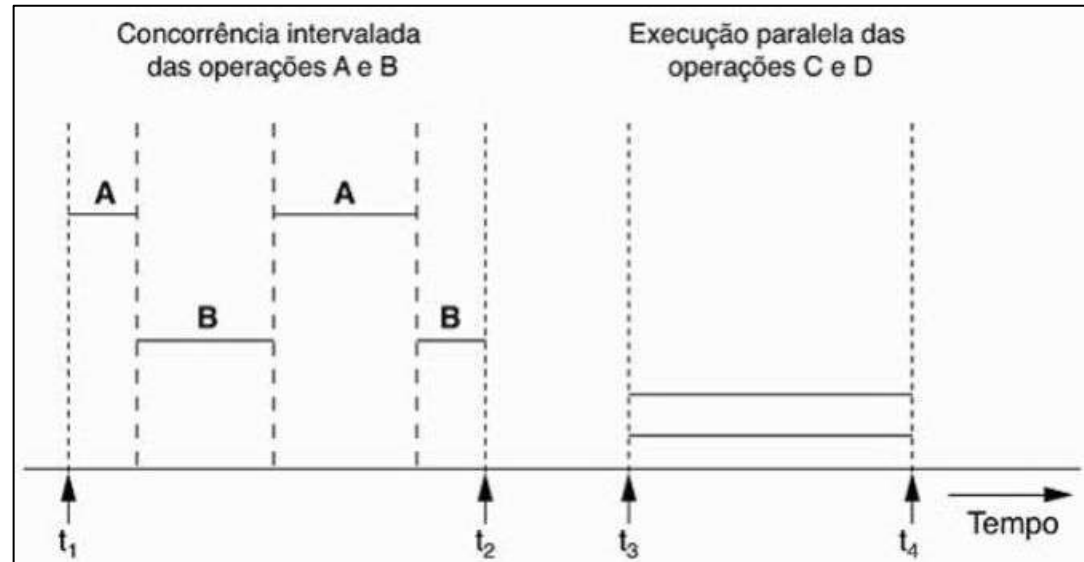
Buffering de Dados

- Buffer é um espaço de memória reservado para armazenamento temporário de dados
 - Podemos ter vários buffers, um para cada aplicação



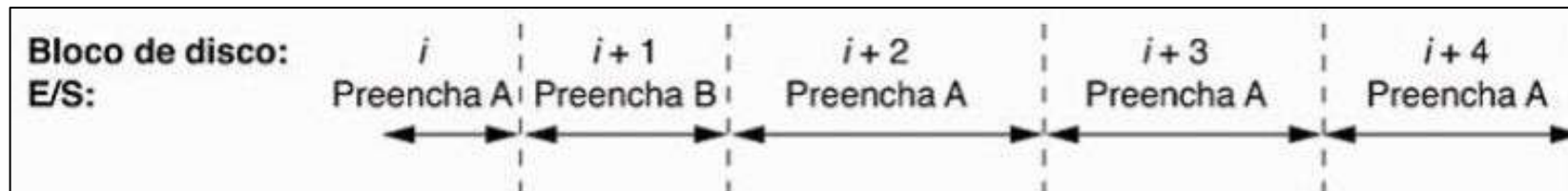
Buffering de Dados

- Buffers agilizam o acesso a dados, pois enquanto dados estão sendo lidos ou gravados em um buffer, a CPU pode processar informações de outros buffers



Buffering Duplo

- Técnica que permite leitura / gravação contínua de dados em blocos de disco contínuos
 - *“Blocos de dados são armazenados um após o outro no HDD”*
 - Elimina tempo de busca e atraso nas transferências de dados
 - Não precisamos mover cabeçote magnético ou motor do HDD a cada leitura de setor



Armazenamento de registros em um SGBD relacional

- Registros são tuplas de atributos e cada atributo possui um domínio (tipo de dados)
 - O domínio do atributo define que tipo de dados ele armazena, e quantos bytes de dados são necessários
 - **Ex:**
 - CHAR(30) precisa de 30 Bytes para armazenar uma string de tamanho 30
 - INT precisa de 4 Bytes para representar um inteiro de 32 bits

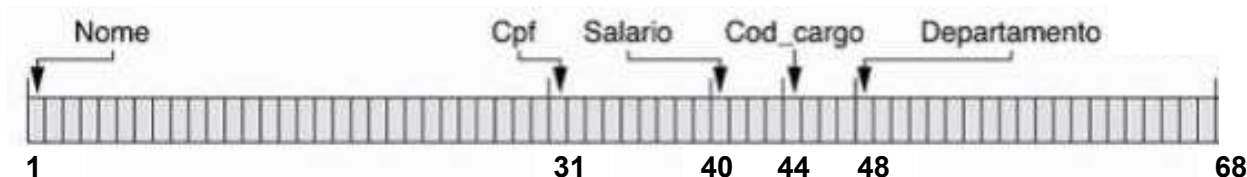
Armazenamento de registros em um SGBD relacional

- SGBD pode armazenar registros como um sucessão de bytes

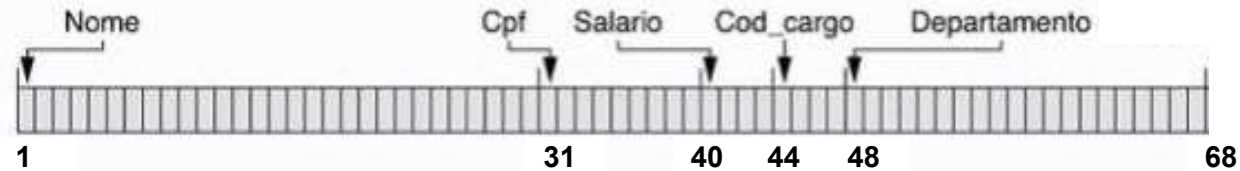
Funcionario				
nome	cpf	salario	cod_cargo	depto
Carlos	777.888.999-00	3100	7	Vendas
Jussara	111.222.333-44	2000	4	Estoque
Maria	000.555.777-11	4000	6	RH

Funcionario
+ nome: char(30)
+ cpf: char(9) , PK
+ salario: int
+ cod_cargo: int
+ depto: char(20)

Cada registro desta tabela pode ser armazenado em disco como uma sequência de atributos:

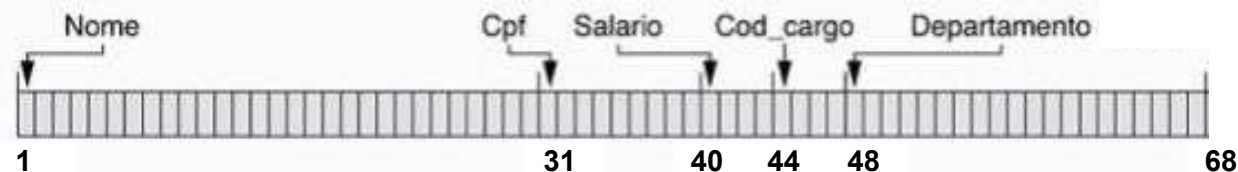


Armazenamento de registros em um SGBD relacional



Sabemos quantos bytes cada atributo necessita. Logo, podemos calcular que cada registro da tabela *Funcionario* ocupa 68 bytes de espaço da memória secundária.

Armazenamento de registros em um SGBD relacional



Sabemos quantos bytes cada atributo necessita. Logo, podemos calcular que cada registro da tabela *Funcionario* ocupa 68 bytes de espaço da memória secundária.

Mas e se a tabela *Funcionario* possuir um atributo de tamanho variável?
Como armazenar um registro como esse?

Armazenamento de registros com atributos de tamanho variável

Considere a tabela *Funcionario* com os seguintes atributos:

Note que *nome* é `varchar(30)`. Ou seja, o nome do funcionário pode conter até 30 caracteres. De maneira similar temos *depto*.

Funcionario
+ nome: varchar(30)
+ cpf: char(9) , PK
+ salario: int
+ cod_cargo: int
+ depto: varchar(20)

Armazenamento de registros com atributos de tamanho variável

Considere a tabela *Funcionario* com os seguintes atributos:

Note que *nome* é `varchar(30)`. Ou seja, o nome do funcionário pode conter até 30 caracteres. De maneira similar temos *depto*.

Funcionario
+ nome: varchar(30)
+ cpf: char(9), PK
+ salario: int
+ cod_cargo: int
+ depto: varchar(20)

Caracteres especiais (ex: *NULL*, *\0*) delimitam os valores dos atributos de tamanho variável, representando assim o último byte armazenado em disco:

Nome	Cpf	Salario	Cod_cargo	Departamento
Silva, João	12345678966	XXXX	XXXX	Computação
1	12	21	25	29

Caracteres separadores

Podemos armazenar registros com atributos de tamanho variável usando vários formatos diferentes

Formato 1 (*somente valor do atributo*):

Nome		Cpf	Salario	Cod_cargo	Departamento
Silva, João		12345678966	XXXX	XXXX	Computação
1	12	21	25	29	

■ Caracteres separadores

Funcionario

- + nome: **varchar(30)**
- + cpf: **char(9), PK**
- + salario: **int**
- + cod_cargo: **int**
- + depto: **varchar(20)**

Funcionario
+ nome: varchar(30)
+ cpf: char(9), PK
+ salario: int
+ cod_cargo: int
+ depto: varchar(20)

Formato 2 (*pares de nome e valor do atributo*):

Nome = Silva, João █ Cpf = 12345678966 █ DEPARTAMENTO = Computação ☒

Caracteres separadores

= Separa nome de campo
do valor do campo

█ Separa campos

☒ Termina registro

Blocagem de Registros

- Dados são gravados na memória secundária em **blocos**
- Registros de uma tabela relacional precisam ser armazenados em blocos de disco
 - Um bloco de disco pode conter vários registros
 - Podemos estimar quantos registros cabem em um bloco usando o **fator de blocagem (bfr)**

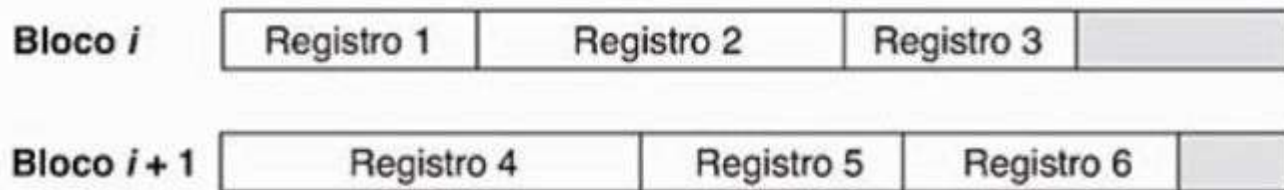
$$\text{bfr} = B / R$$

The diagram illustrates the formula $\text{bfr} = B / R$. Below the formula, there are two red-outlined boxes. The left box contains the text "Tamanho do bloco de disco (bytes)" and has a red arrow pointing upwards to the variable B in the formula. The right box contains the text "Tamanho dos registros (bytes)" and has a red arrow pointing upwards to the variable R in the formula.

Organização não espalhada

$$\mathbf{bfr = B / R}$$

Nem sempre existe uma divisão exata entre B e R.
Nestes casos, existirão espaços vazios dentro dos blocos de disco.



Organização não espalhada

$$\mathbf{bfr = B / R}$$

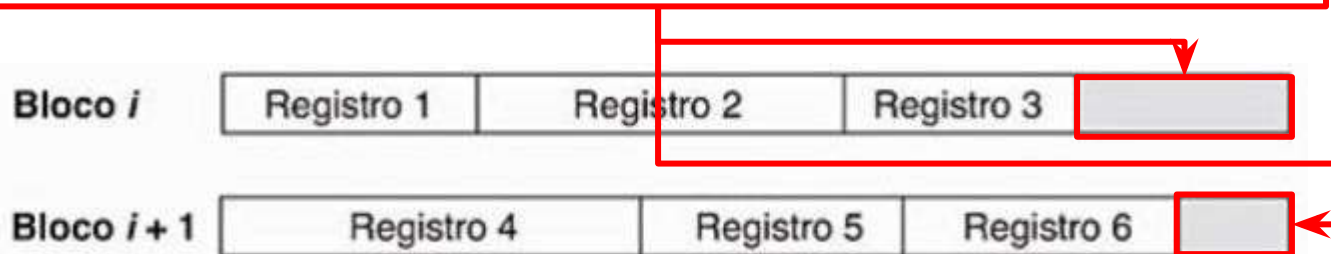
Nem sempre existe uma divisão exata entre B e R.
Nestes casos, existirão espaços vazios dentro dos blocos de disco.



Organização não espalhada

$$\mathbf{bfr = B / R}$$

Nem sempre existe uma divisão exata entre B e R.
Nestes casos, existirão espaços vazios dentro dos blocos de disco.

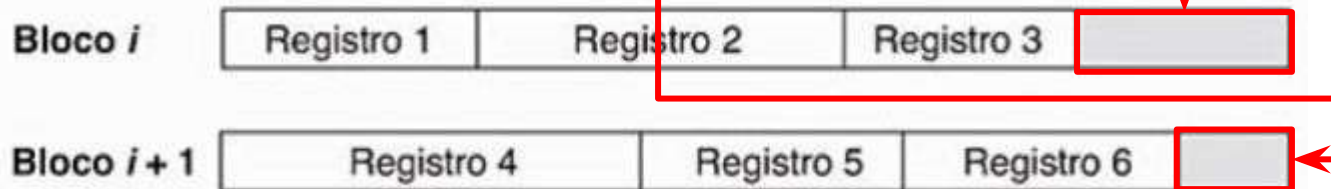


Esta situação ocorre porque os registros não podem ser divididos (eles devem ser armazenados inteiramente dentro de um bloco).

Organização não espalhada

$$\text{bfr} = B / R$$

Nem sempre existe uma divisão exata entre B e R.
Nestes casos, existirão espaços vazios dentro dos blocos de disco.



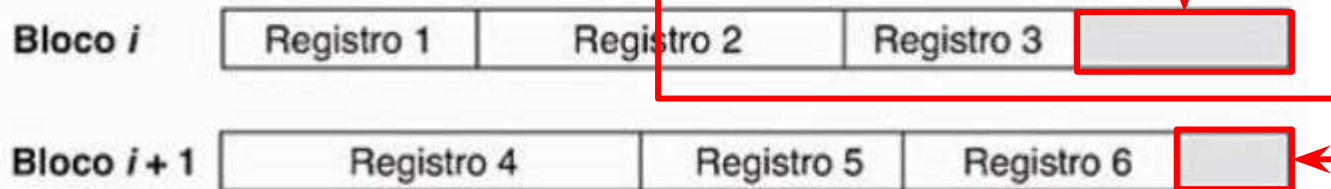
Esta organização de registros é chamada de **organização não espalhada**.

Como corrigir essas perdas de espaço em disco?

Organização não espalhada

$$\text{bfr} = B / R$$

Nem sempre existe uma divisão exata entre B e R.
Nestes casos, existirão espaços vazios dentro dos blocos de disco.



Esta organização de registros é chamada de **organização não espalhada**.

Como corrigir essas perdas de espaço em disco?

Organização espalhada

Podemos armazenar uma parte de um registro em um bloco, e o restante em outro bloco. Assim, podemos melhorar o uso do espaço!



Organização espalhada

Podemos armazenar uma parte de um registro em um bloco, e o restante em outro bloco. Assim, podemos melhorar o uso do espaço!



Um ponteiro no final do primeiro bloco aponta para o bloco que contém o restante do registro

Organização espalhada

Podemos armazenar uma parte de um registro em um bloco, e o restante em outro bloco. Assim, podemos melhorar o uso do espaço!



Esta organização de registros é chamada de **organização espalhada**.

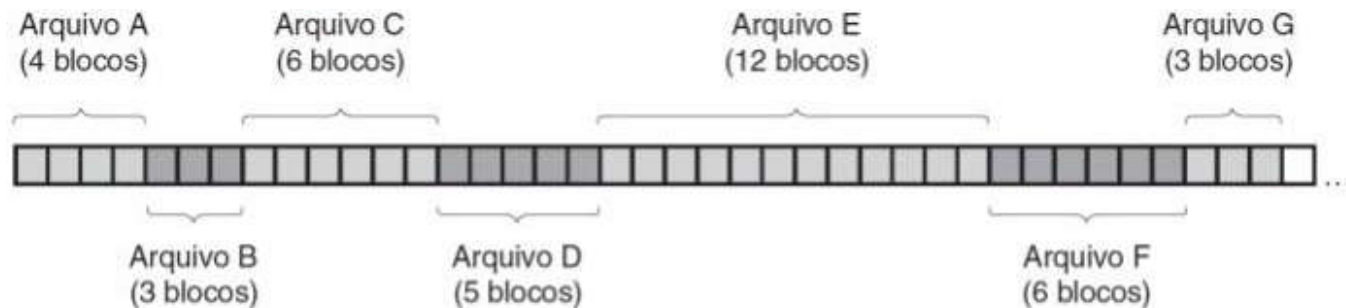
Um ponteiro no final do primeiro bloco aponta para o bloco que contém o restante do registro

Alocação de blocos em disco

- Existem várias formas de se armazenar blocos fisicamente em memórias secundárias (HDDs, SSDs, etc)
 - As formas mais utilizadas são:
 - **Alocação contígua**
 - Arquivos (registros) são alocados a blocos de disco de forma consecutiva
 - **Alocação ligada**
 - Cada bloco contém um ponteiro para o próximo bloco de arquivo

Alocação contígua

- Arquivos (registros) são alocados a blocos de disco de forma consecutiva



Vantagem:

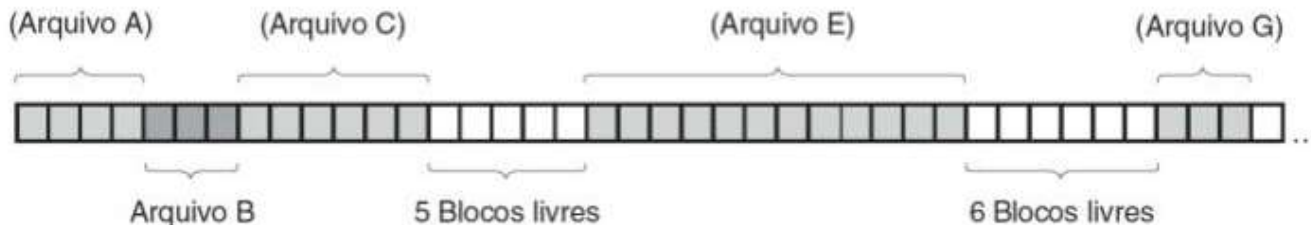
leitura rápida de dados (por conta do buffering duplo)

Desvantagem:

Dificuldade de expandir arquivos

Alocação contígua

- Arquivos (registros) são alocados a blocos de disco de forma consecutiva



Desvantagem:

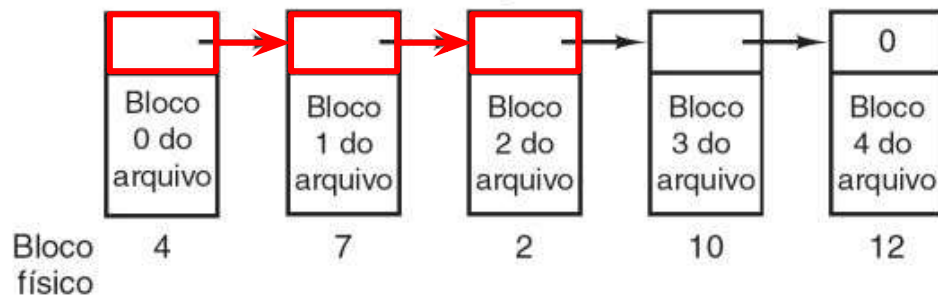
Fragmentação de disco (espaços vazios no meio do disco), o que gera problemas no gerenciamento do espaço de disco

Como faço para armazenar um arquivo H com 7 Blocos?

Seria possível armazenar H, pois o disco tem $5 + 6 = 11$ Blocos livres. Porém esses Blocos livres não estão juntos.

Alocação ligada (ou alocação encadeada)

- Cada bloco de arquivo contém um ponteiro para o próximo bloco de arquivos



Vantagem:
facilidade para
expandir arquivos

Desvantagem:
leitura de dados lenta
(leitura sequencial de
blocos)

Para ler um bloco no meio de um
arquivo, precisamos ler todos os
blocos anteriores em sequência

Referencial Bibliográfico

- KORTH, H.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistemas de bancos de dados**. 5. ed. Rio de Janeiro: Ed. Campus, 2006.
- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Ed. Campus, 2004. Tradução da 8ª edição americana.