



# **Armazenamento de Disco, Estruturas de Arquivos Básicas e Hashing**

André Luis Schwerz  
Rafael Liberato Roberto

# Tópicos

- Introdução
- Armazenamento de Dados
  - Memória primária, secundária e terciária
- Buffering de Blocos
- Registro de tamanho fixo e variável
- Arquivos de registros desordenados (Heap files)
- Arquivos de registros ordenados (Sorted files)
- Técnicas de hashing
  - Hashing interno e externo
- Acesso Paralelo em Disco usando RAID

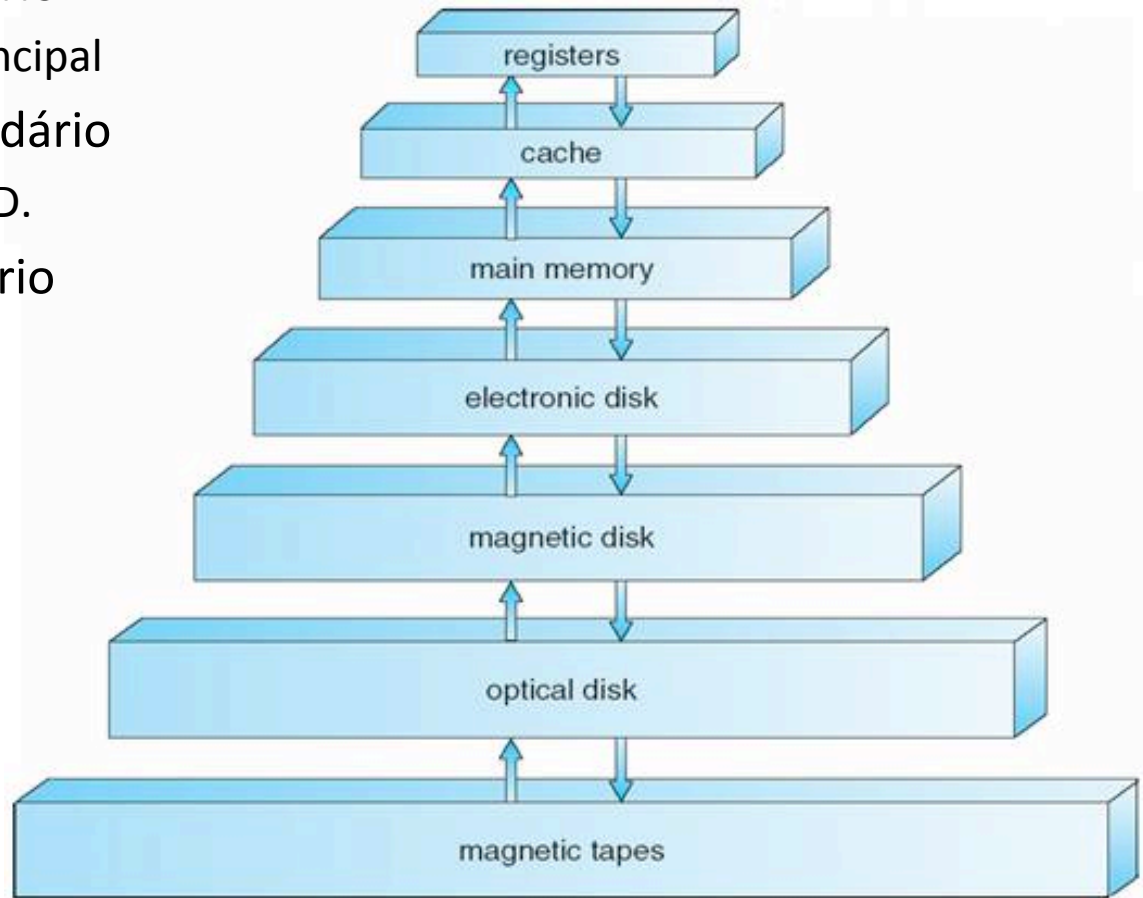
# Introdução

---

- As coleções de dados que compõe um banco de dados precisam ser de alguma forma armazenadas em algum mídia de armazenamento, para que assim, o SGBD possa recuperar, atualizar e processar esses dados de acordo com a sua necessidade.

# Armazenamento de Dados

- Hierarquia de armazenamento:
  - Armazenamento primário
    - Cache e memória principal
  - Armazenamento secundário
    - Disco rígido, CD e DVD.
  - Armazenamento terciário
    - Fitas magnéticas



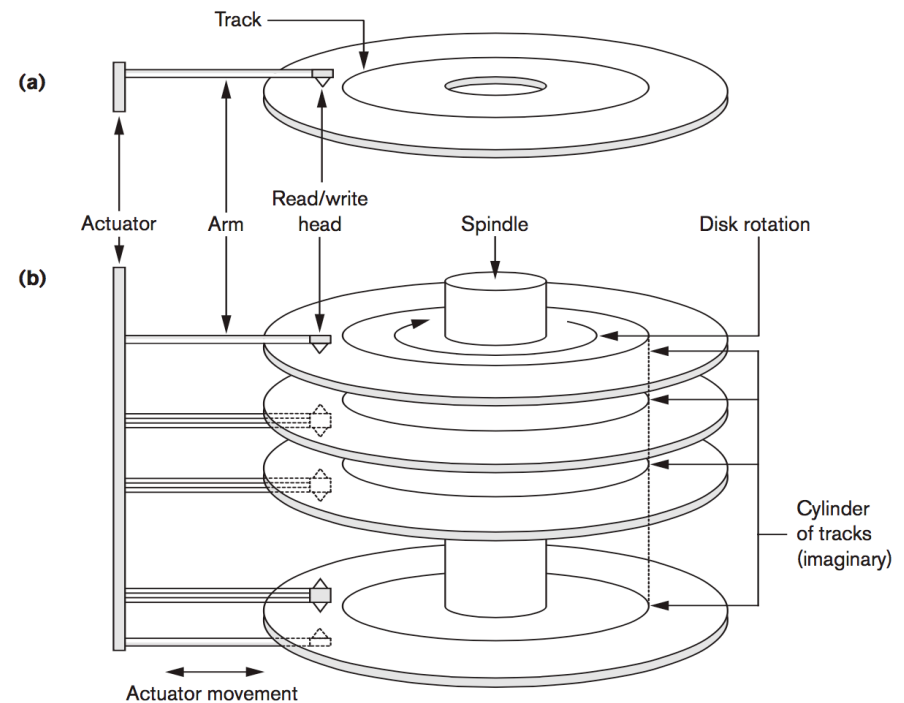
# Memória Principal

---

- Memória RAM (Random Access Memory)
  - Memória de acesso aleatório
  - Velocidade de transferência centenas de vezes superior à dos dispositivos de memória de massa
  - Extremamente simples: um minúsculo capacitor, que quando está carregado eletricamente
  - Volátil
  - Módulos de memória são divididos em linhas e colunas.
- Memória cache
  - Memórias não eram mais capazes de acompanhar a velocidade dos processadores
  - Tipo ultrarrápido de memória que serve para armazenar os dados mais frequentemente usados pelo processador
  - Extremamente caro (chega a ser algumas centenas de vezes mais cara que a memória RAM convencional)

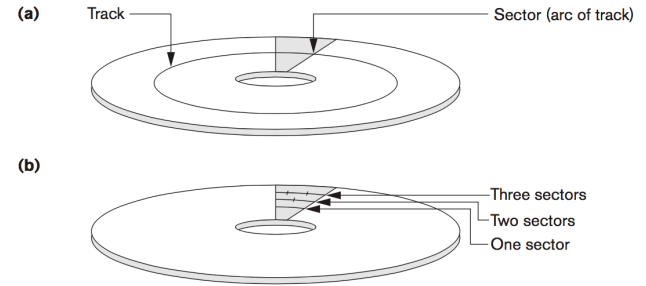
# Memória Secundária

- Armazenamento em disco
  - São utilizados para armazenamento de grande quantidade de dados
  - Não voláteis.
  - Na unidade de disco estão presentes:
    - Cabeçote de leitura/escrita,
    - Braço mecânico
    - Atuador e controladora de disco
    - Divisão física e divisão lógica



# Memória Secundária

- Divisão física (não tem como mudar)
  - Trilhas
  - Cilindros
  - Setores
- Divisão lógica (formatação)
  - **Blocos: unidade de transferência entre o disco e a memória principal**
- Operação de leitura/escrita move o cabeçote do disco para o bloco a ser transferido.
- Movimentos de rotação posicionam no setor apropriado.
- Um endereço físico de bloco de disco consiste de:
  - número do cilindro
  - número da trilha
  - número do bloco



# Memória Secundária

- Desempenho baseado na medida de três tempos:
  - Tempo de busca
  - Tempo de atraso rotacional (latência)
  - Tempo de transferência de bloco
- Tempo de pesquisa e atraso rotacional são geralmente muitos maiores que o tempo de transferência do bloco



# Memória Terciária

- Armazenamento em fitas magnéticas
  - Dispositivos de armazenamento de acesso sequencial
  - Os dados são armazenados em cartuchos de fitas magnéticas
  - Parecidas com as fitas de áudio e vídeo comuns
  - Cada byte disposto de forma transversal na fita
  - Acesso lento
  - **Utilizadas principalmente para backup**



# Buffering de Blocos

---

- Diversos buffers podem ser reservados na memória principal para acelerar a transferência de dados
- Enquanto um buffer estiver sendo lido ou escrito a CPU pode processar os dados em outro.
- Útil para processos executados concorrentemente.
- Buffering duplo: permite que leituras e escritas sejam realizadas de forma contínua em blocos consecutivos no disco, eliminando assim o tempo de pesquisa e atraso rotacional

# Registros de arquivos

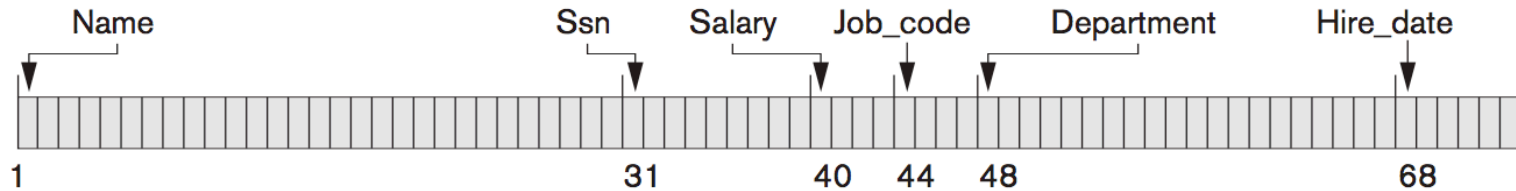
- Um arquivo é uma sequencia de registros
- Registro é uma coleção de valores de dados (itens de dados)

```
struct employee{  
    char name[30];  
    char ssn[9];  
    int salary;  
    int job_code;  
    char department[20];  
};
```

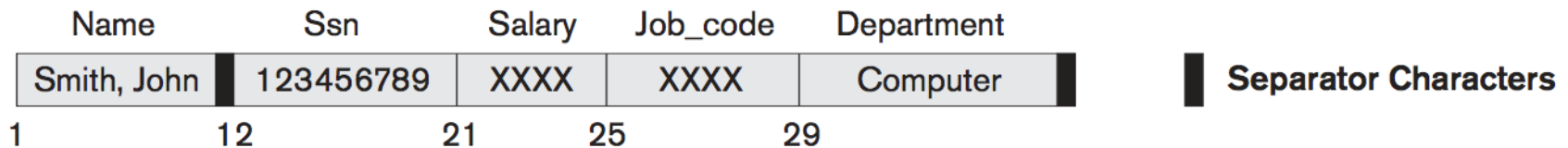
# Tipos de registros de arquivos

- Tipos de registros:

- Tamanho fixo:



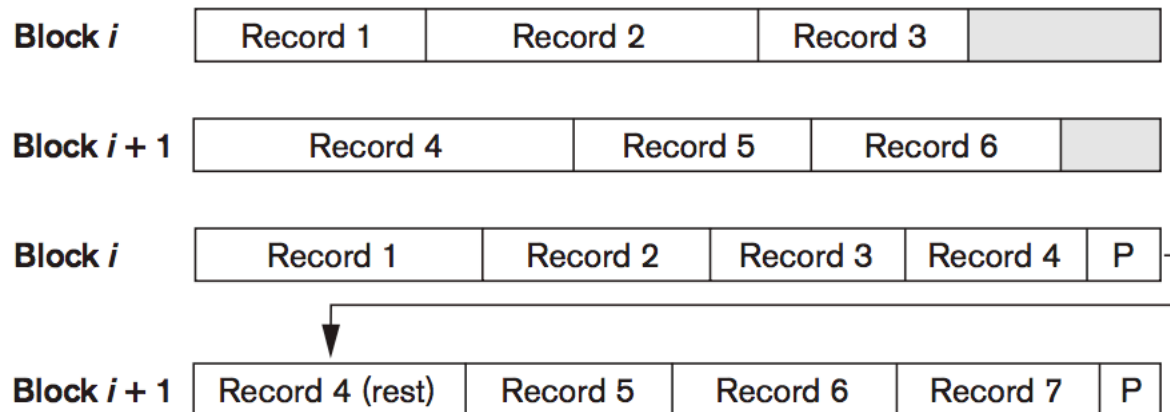
- Tamanho variável:



- Arquivos com mesmo tipo de registro e de tamanhos fixo facilitam a procura para SGBDs.
- Arquivos com registros de campos de tamanho variável:
  - Caracteres de separação ou campos de comprimento

# Blocagem de registros

- Registros são gravados em blocos.
- Registros de arquivo podem ser:
  - Não-espalhados:
    - Registro não pode passar o limite do bloco
  - Espalhados
    - Um registro pode ser armazenado em mais de um bloco



# Blocagem de registros

- Suponha:
  - Tamanho do Bloco ( $B$ ) em bytes
  - Tamanho do Registro ( $R$ ) em bytes
  - Fator de blocagem ( $bfr$ )
    - número de registros que podem ser armazenados em um bloco.
- **Se  $R \leq B$ :**
  - Fator de blocagem:
    - $bfr = \text{floor}(B/R)$
  - Espaço não usado:
    - $B - (bfr * R)$  bytes

# Registros de tamanho fixo

- Suponha a abordagem não espalhada
  - Os registros não podem atravessar as fronteiras de um bloco
- Alternativas para exclusão de um registro  $i$ :
  - Mover os registros  $i + 1, \dots, n$  para  $i, \dots, n - 1$
  - Mover o registro  $n$  para  $i$
  - Não mover registros, mas manter todos os registros vazios em uma *freelist*

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

- Excluir o registro 3 e compactar

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



# Registros de tamanho fixo

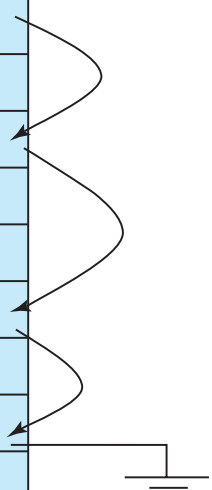
- Excluir o registro 3 e mover o último registro.

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

- Free List:
  - Armazenar o endereço do primeiro registro excluído no cabeçalho do arquivo
  - Usar este primeiro registro para armazenar o endereço do segundo registro excluído e assim por diante.

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

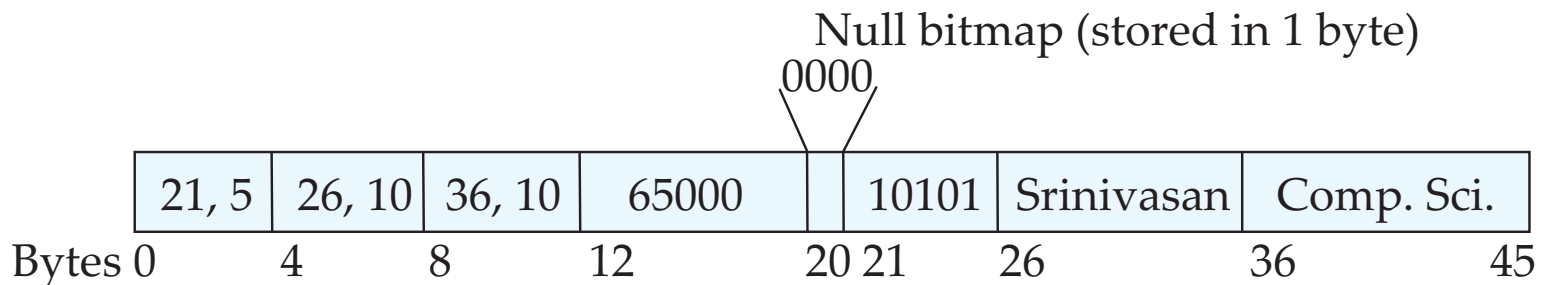


# Registros de tamanho variável

- Registros de tamanho variável surgem em sistemas de banco de dados de várias maneiras:
  - Armazenamento de vários tipos de registro em um arquivo
  - Tipos de registro que permitem tamanhos variáveis para um ou mais campos (VARCHAR)
  - Tipos de registro que permitem campos repetitivos, como arrays ou multiconjuntos
- Duas questões:
  - 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 em um bloco sejam facilmente extraídos?

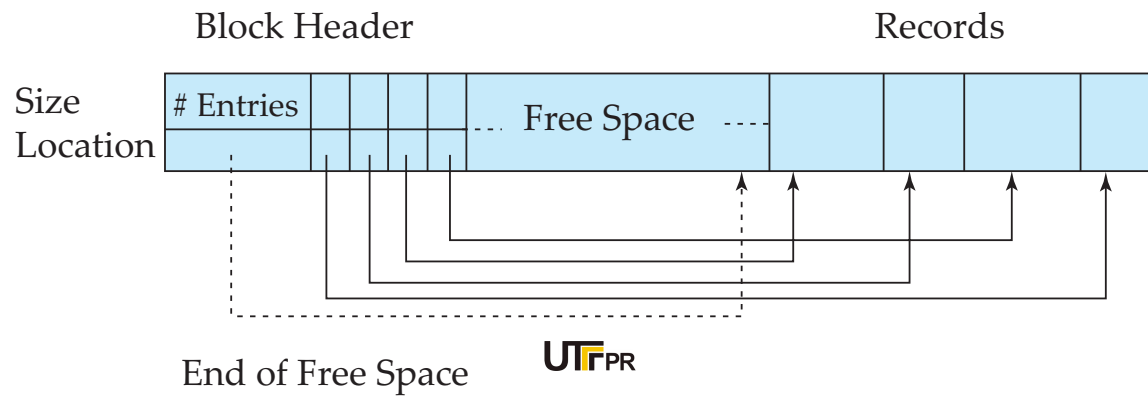
# Registros de tamanho variável

- Como representar um único registro, de modo a permitir a extração fácil dos atributos individuais?
  - Suponha que os atributos **ID**, **nome** e **nome\_depto** são string e o quarto atributo **salario** é fixo.
  - Atributos estão armazenados em ordem
  - Atributos de tamanho variáveis representados por tamanho fixo (deslocamento, tamanho), com os dados armazenados após todos atributos de tamanho fixo.
  - Valores nulos representados pelo bitmap nulo.



# Registros de tamanho variável

- Como armazenar registros de tamanho variável dentro de um bloco, de modo que os registros em um bloco sejam facilmente extraídos?
  - O cabeçalho do bloco contém:
    - Número de entradas de registros no cabeçalho
    - Final do espaço livre no bloco
    - Localização e tamanho de cada registro
  - Registros podem ser alocados de forma contígua no bloco, começando do final do bloco.
  - O espaço livre é contíguo entre a entrada final no array do cabeçalho e o primeiro registro.
  - Uso de ponteiros indiretos para permitir a movimentação dos registros dentro do bloco.



# Operações em arquivos

- As operações em um arquivo podem ser:
  - operações de recuperação
  - operações de atualização
- Operações mais comuns:
  - **Open** (abrir):
    - Prepara o arquivo para leitura ou gravação.
    - Recupera o cabeçalho do arquivo.
    - Define o ponteiro de arquivo para o início do arquivo.
  - **Reset** (reinicializar):
    - Reposiciona o ponteiro de arquivo, de um arquivo aberto, no seu início.
  - **Find** (encontrar):
    - Busca o primeiro registro que satisfaça a condição de pesquisa, transfere o bloco que tem a condição de pesquisa para um buffer de memória principal e faz o ponteiro de arquivo apontar para o registro no buffer, tornando-o o registro atual.

# Operações em arquivos

- **Read** (ler):
  - copia o registro atual do buffer para uma variável do programa de usuário.
- **FindNext** (encontrar o próximo):
  - procura o próximo registro no arquivo que satisfaça a condição de pesquisa, transferindo o bloco que contém aquele registro para o buffer da memória principal.
- **Delete** (excluir):
  - exclui o registro atual e no final atualiza o arquivo de disco para refletir a exclusão.
- **Modify** (modificar):
  - modifica alguns valores de campos do registro atual e no final atualiza o arquivo no disco para refletir a modificação.
- **Insert** (incluir):
  - acrescenta um novo registro no arquivo por meio da localização do bloco no qual o registro deve ser incluído, transferindo aquele bloco para o buffer da memória principal, escrevendo o registro no buffer e no final escrevendo o buffer no disco para refletir a modificação.
- **Close** (fechar):
  - finaliza o acesso ao arquivo por meio da liberação dos buffers e da execução de quaisquer outras operações de limpeza necessárias

# Organização dos registros nos arquivos

- **Heap** – um registro pode ser colocado em qualquer lugar no arquivo que existir espaço livre
- **Ordenado** – armazenar os registros em ordem sequencial baseados no valor de um chave de busca de cada registro
- **Hashing** – uma função de hash é calculada em algum atributo de cada registro; o resultado especifica no qual bloco do arquivo o registro deveria ser colocado.
- Registros de cada relação podem ser armazenados em arquivos separados. Em uma **multitable clustering file organization**, registros de diferentes relações podem ser armazenados em um mesmo arquivo.
  - Motivação: armazenar registros relacionados em um mesmo bloco para minimizar E/S.



# Arquivos de registros desordenados (Heap files)

---

- Também chamados de arquivos de pilha
- Novos registros são inseridos no final do arquivo
- Para procurar um registro é necessário uma busca linear por meio dos registros
- Inserção de registros é bem eficiente
- Ler os registros em ordem de acordo com um campo em particular exige a ordenação prévia dos registros do arquivo.

# Arquivos de registros ordenados (Sorted files)

---

- Também chamados de arquivo sequencial
- Registros de arquivos são mantidos ordenados por de acordo com um valor um certo campo
- Inserção é custosa
- É comum manter um arquivo de *overflow* para agilizar as inserções
- É possível fazer uma procura binária através do valor do campo de ordenação

# Tempos de acesso médios

- Tempos de acesso médios para um arquivo de  $b$  blocos sob as organizações de arquivos básicas

Tipo de Organização	Método de Acesso/busca	Média de blocos para acessar um registro específico
Heap (não ordenado)	Busca sequencial	$b/2$
Ordenado	Busca sequencial	$b/2$
Ordenado	Busca Binária	$\log_2 b$

# Técnicas de hashing

---

- A ideia do hashing é oferecer uma função  $h(x)$ , chamada de **função hash** que, aplicada ao valor do campo de hash de um registro, gere o endereço do bloco de disco no qual o registro está armazenado.

# Hashing interno

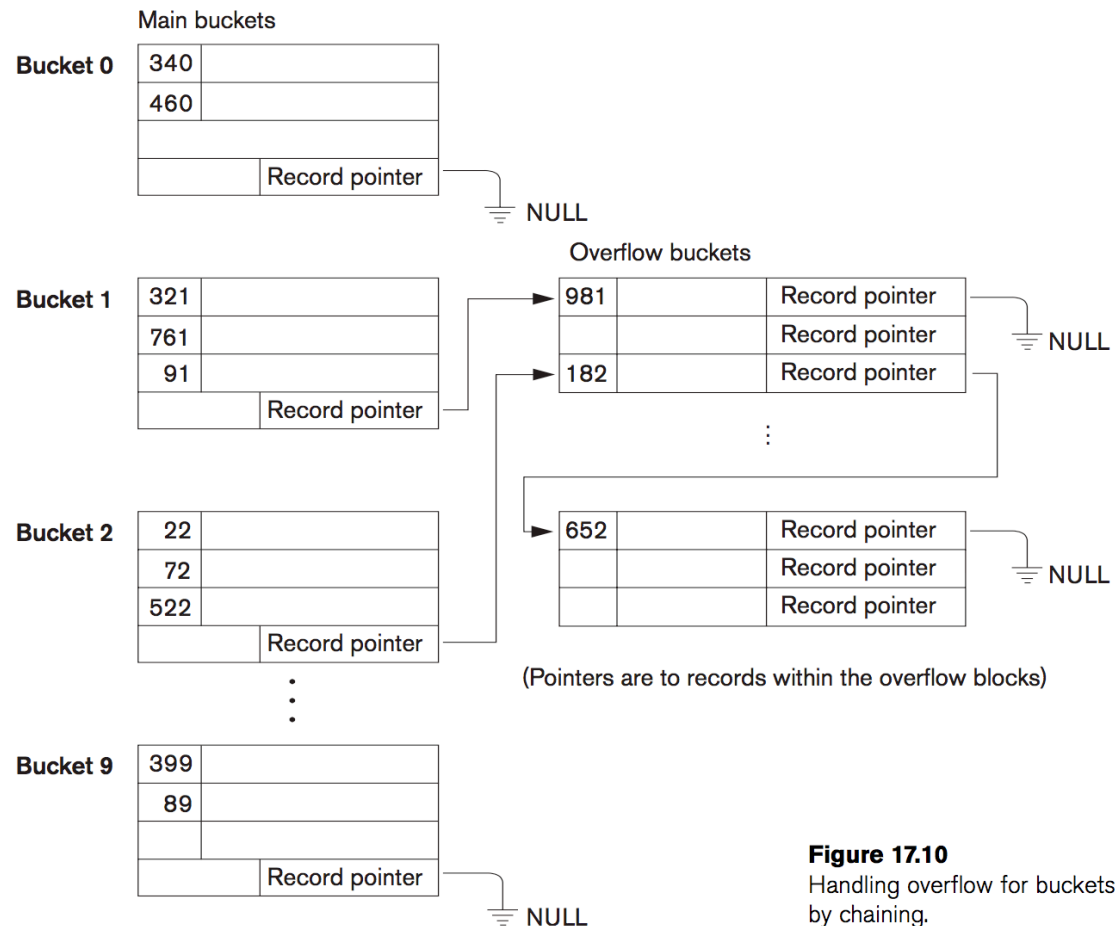
- Hashing interno é normalmente implementada através de uma **tabela hash** por meio de um vetor de registros
- Suponha que o intervalo de índice do vetor varie de 0 a M-1
- Uma função típica para isto seria a função:
  - $h(K) = K \bmod M$
  - este valor será então usado como endereço do registro
- Funções hash não garantem endereços únicos
  - Ocorrência de colisões

# Hashing Interno

- Métodos para tratar colisões:
  - Open Addressing (Endereçamento aberto)
    - A partir da posição ocupada, verificar qual posição está vazia.
  - Encadeamento (Chaining)
    - Utilizar locais de overflow
  - Hashing Múltiplo
    - Aplicar uma segunda ou, até mesmo, uma terceira função de hash.

# Hashing externo

- Chama-se hash externo quando se trata de hashing para arquivos em disco
- O espaço de endereçamento alvo é constituído de **buckets**.
- **Buckets** são grupos de blocos de disco consecutivos.
- A **função hash** mapeia uma chave a um número de bucket



**Figure 17.10**  
Handling overflow for buckets  
by chaining.

# Acesso Paralelo em Disco usando RAID

- Tecnologia de armazenamento secundário deve tomar medidas para manter o desempenho e a confiabilidade de acordo com a tecnologia dos processadores.
- O maior avanço na tecnologia de armazenamento secundário é representado pelo desenvolvimento do RAID (*Redundant Array of Independent Disk*)
- O principal objetivo do RAID é acabar com a grande diferença de performance dos discos magnéticos comparado às tecnologias da memória e dos processadores.



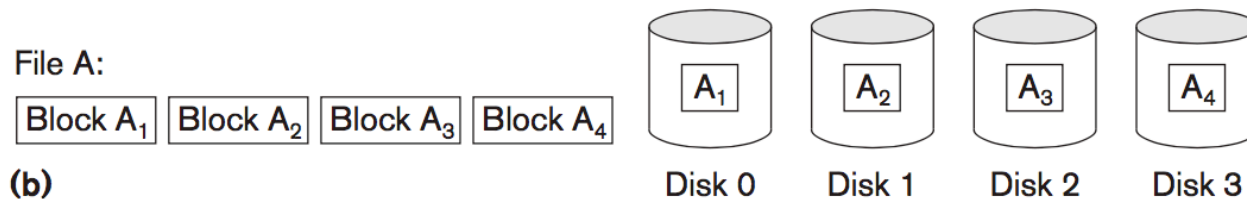
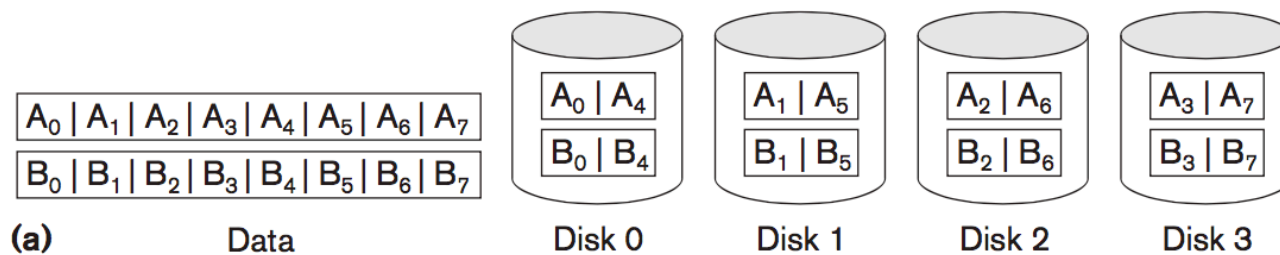
# Acesso Paralelo em Disco usando RAID

---

- A solução é usar um grande array de pequenos discos independentes atuando como um único disco lógico de maior desempenho
- Um conceito que usa o paralelismo para melhorar o desempenho de dados é chamado de ***striping***
- O *striping* de dados distribui os dados de maneira transparente por múltiplos discos como se fosse um único disco grande e rápido

# Acesso Paralelo em Disco usando RAID

- Um arquivo distribuído ou striped por quadro discos
  - A) Nível de bit
  - B) Nível de blocos



# Melhorando a confiabilidade

- *Mean Time Between Failures* – (MTBF)
  - 1 disco – 1,4 milhões de horas
  - 100 discos – 2.000 horas
  - 1000 discos – 1400 horas
- Manter uma única cópia causará uma perda de confiabilidade.
- Solução:
  - Redundância
- Desvantagens:
  - Operações de E/S adicionais.
  - Computação extra
  - Capacidade adicional para manter informação redundante.

# Melhorando a confiabilidade

- **Espelhamento** ou **sombreamento** é uma técnica para introduzir redundância.
- Em uma leitura, os dados podem ser obtidos dos espelhos com menores atrasos de fila, busca e rotacionais.
- Outra solução é o uso de **bits de paridades**.
  - Por exemplo, um disco redundante pode ser considerado como tendo a soma de todos os dados nos outros discos. Quando um disco falha, a informação perdida pode ser construída por um processo de subtração.
- Veja o **Código de Hamming** [aqui](#)

# Acesso Paralelo em Disco usando RAID

- Foram definidas diferentes organizações de RAID com base em diferentes combinações de dois fatores de granularidade de dados e padrão utilizado para calcular a informação redundante
- RAID nível 0 (Striping ou Fracionamento):
  - Dados são divididos em pequenos segmentos e distribuídos entre os discos
  - Não existe redundância
  - Melhor desempenho de gravação pois não faz o espelhamento
  - Menor desempenho de leitura comparado ao nível 1, pois não há o espelhamento.
- RAID nível 1 ("Mirroring" ou "Espelhamento"):
  - Adiciona discos paralelos aos discos principais
  - Os discos adicionados trabalham como uma cópia do original

# Acesso Paralelo em Disco usando RAID

---

- RAID nível 2:
  - Adapta o mecanismo de detecção de falhas em discos rígidos (código Hamming)
  - Todos os discos da matriz ficam sendo "monitorados" pelo mecanismo
- RAID nível 3
- RAID nível 4
- RAID nível 5
- RAID nível 6

