

# ÍNDICES

---

Sérgio Mergen

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Introdução


- Tabelas muito grandes são mantidas no disco
  - Não cabem na memória primária
- Dada uma consulta

```
SELECT * FROM func WHERE idFunc = 1244
```

- Deve-se localizar a página onde os registros solicitados se encontram
- Em muitos casos, seria necessário acessar muitas páginas até encontrar o registro desejado

# Introdução

- Usando como exemplo a implementação da classe RecordManager (visto em uma aula passada)
  - para localizar o registro cujo id é **60**, deve-se passar por todos os blocos (páginas) anteriores



10	
20	

25	
40	

50	
<b>60</b>	

70	
80	

# Introdução

- Mecanismos de indexação são usados para acelerar o acesso aos dados.
  - Objetivo é reduzir o número de páginas de dados a serem carregadas para a memória
- **Chave de busca** – atributo(s) usado(s) para localizar registros em um arquivo.
  - Ex. chave de busca **idM**
    - Os registros de movie são localizados com base em seu id

# Introdução

- Métricas de avaliação de índices
  - Tempo de acesso
  - Tempo de inserção
  - Tempo de remoção
  - Sobrecarga de espaço
- Tipos de acesso eficientes suportados
  - Consulta por equivalência
    - Ex. year = 2000
  - Consulta por intervalo
    - Ex. year >2000
    - Ex. year >2000 **AND** year < 2010

# Introdução

- Dois tipos básicos de índices:
  - **Índices Hash:** chaves de busca são distribuídas uniformemente em buckets usando uma função de hash.
  - **Índices ordenados:** chaves de busca são armazenadas em ordem.

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível



# Índices Hash

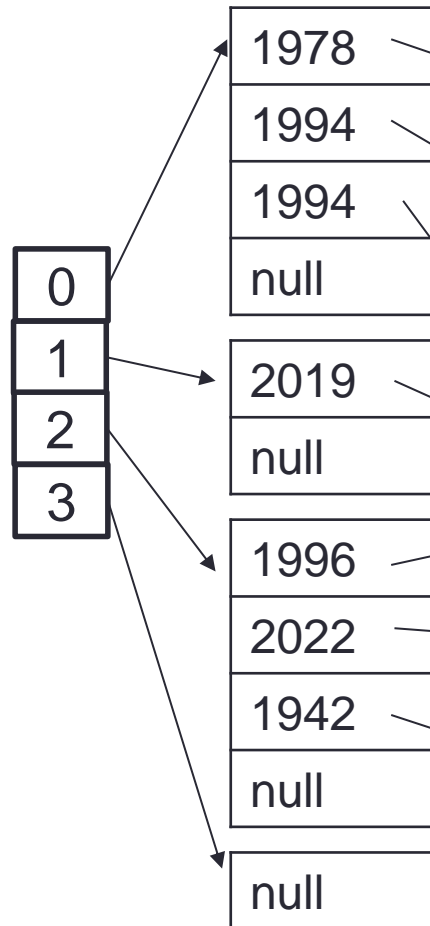
- Uma função hash  $h$ 
  - mapeia todas chaves de busca para posições de um vetor(buckets).
  - é usada para localizar registros para acesso, inserção e remoção.
- Registros com valores de chave de busca diferentes podem ser mapeados para o mesmo bucket.
- Todo o bucket deve ser varrido sequencialmente para localizar o registro.
- Custo constante (se usada uma função boa)

# Índices Hash

- Um índice hash pode ser usado de duas formas
  - Como um índice em memória
    - Nesse caso, o índice não tem relação nenhuma com a organização do arquivo
  - Como uma forma de organização de arquivo

# Índices Hash (em memória)

Hash em memória



Exemplo de função hash:

$hash(valor) = \text{dígito menos significativo} \% 4$

1	Star Wars	1978	p1
3	Forest Gump	1994	p2
4	Toy Story	1996	
6	Joker	2019	p3
7	Batman	2022	
10	Pulp Fiction	1994	p4
11	Casablanca	1942	

Arquivo de dados

# Índices Hash (para organizar o arquivo)

Exemplo de função hash:

$hash(valor) = \text{dígito menos significativo} \% 4$

*páginas sequenciais*

*Páginas de estouro*

p1

1	Star Wars	1978
3	Forest Gump	1994



10	Pulp Fiction	1994

p13

p2

6	Joker	2019

p3

4	Toy Story	1996
7	Batman	2022



11	Casablanca	1942

p21

p4


*Arquivo de dados*

# Sumário

- Introdução
- Índices Hash
- **Índices Ordenados**
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Índice Ordenado

- Em um **índice ordenado**
  - as entradas do índice são ordenadas pela chave de busca.
- Um **arquivo de índice** consiste em registros (chamados **entradas do índice**) na forma
  - Chave de busca
  - Ponteiro para o registro que possui a chave de busca

# Índice Ordenado

- Arquivos de índice são tipicamente muito menores do que o arquivo original
  - Possuem um número menor de páginas
- Possivelmente todo o arquivo de índice caiba na memória
  - O que reduz o custo quando o índice deve ser utilizado
  - Pois não é necessário se preocupar com a transferência para a memória

# Índice Ordenado

chave	ponteiro
-------	----------

1	
2	
3	
4	

6	
7	
10	
11	

14	
18	

idM	title	year
-----	-------	------

1	Star Wars	1978
2	Terminator	1984

3	Forrest Gump	1994
4	Toy Story	1996

6	Joker	2019
7	Batman	2022

10	Pulp Fiction	1994
11	Casablanca	1942

14	Godfather	1972
18	Get Out	2017

Arquivo de índice (idM)

Arquivo de dados (movie)



# Índice Ordenado

O arquivo de índice ocupa menos páginas do que o arquivo de dados

chave	ponteiro
-------	----------

1	
2	
3	
4	

6	
7	
10	
11	

14	
18	

idM	t
-----	---

1	Star wars	1978
2	Terminator	1984

3	Forrest Gump	1994
4	Toy Story	1996

6	Joker	2019
7	Batman	2022

10	Pulp Fiction	1994
11	Casablanca	1942

14	Godfather	1972
18	Get Out	2017

Arquivo de índice (idM)

Arquivo de dados (movie)

# Comparação dos índices

- Índices B+

- Desempenho bom em consultas por intervalo
- Desempenho bom em consultas por igualdade
- Por isso, é o índice preferido dos SGBDs

- Índices Hash

- Desempenho ótimo em consultas por igualdade
- Desempenho péssimo em consultas por intervalo
- A função é muito dependente do domínio de dados
- Se a função for ruim
  - Desempenho cai
- Por isso, esse índice tem pouco suporte em SGBDs

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Tipo da chave de busca

- Um índice ordenado pode ser classificado quanto ao tipo da chave de busca
  - Índices primário
  - Índices secundário

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Índice Primário

- É o índice criado sobre a chave primária de uma tabela
- Se o arquivo de dados for sequencial, a ordem dos ponteiros se alinha com a ordem dos registros
  - a leitura por ordem de chave primária é barata
- Se o arquivo de dados for uma heap, a ordem dos ponteiros e dos registros não possui nenhuma relação
  - A leitura por ordem de chave primária pode se tornar cara
    - Devido à necessidade de buscas aleatórias a partir do índice

# Índice Primário – arquivo sequencial



# Índice Primário – arquivo sequencial

chave	ponteiro	idM	title	year	
1		1	Star Wars	1978	1
2		2	Terminator	1984	2
3					
4				94	3
				96	4
6					
7				19	5
10				22	6
11					
		10	Pulp Fiction	1994	7
14		11	Casablanca	1942	8
18					
		14	Godfather	1972	9
		18	Get Out	2017	10

A leitura por ordem de chave primária em um arquivo sequencial é barata.

Basta acessar o arquivo de dados.

Sequer é preciso consultar o índice

Arquivo de índice (id)

Arquivo de dados sequencial (movie)



# Índice Primário – arquivo heap

chave	ponteiro
-------	----------

1	
2	
3	
4	

6	
7	
10	
11	

14	
18	

idM	title	year
-----	-------	------

1	Star Wars	1978
11	Casablanca	1942

4	Toy Story	1996
2	Terminator	1984

3	Forrest Gump	1994
7	Batman	2022

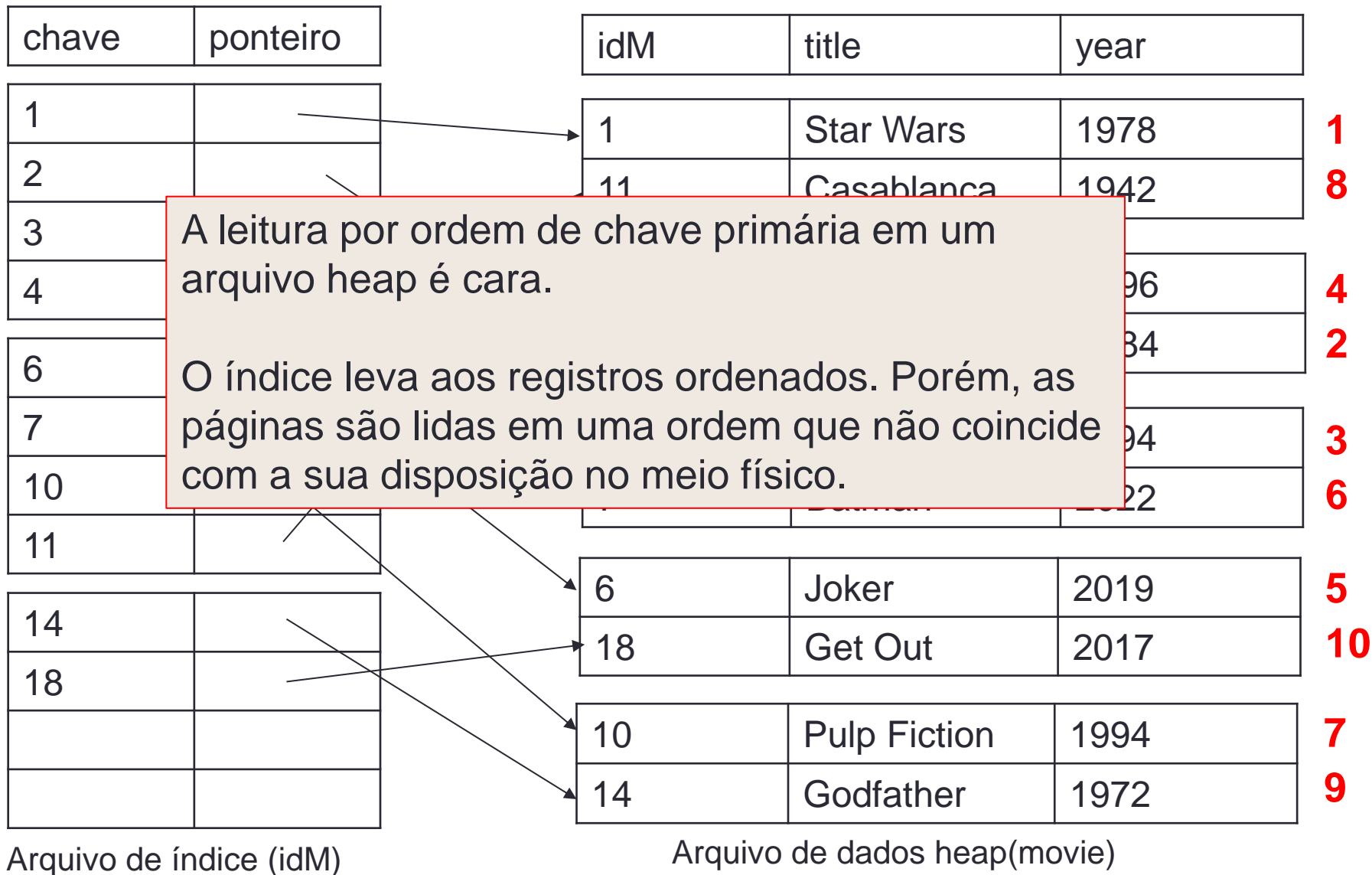
6	Joker	2019
18	Get Out	2017

10	Pulp Fiction	1994
14	Godfather	1972

Arquivo de índice (idM)

Arquivo de dados heap (movie)

# Índice Primário – arquivo heap



# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Índice Secundário

- É o índice criado sobre qualquer conjunto de colunas que não seja chave primária da tabela
- A ordem dos ponteiros e dos registros não possui nenhuma relação
  - Seja o arquivo sequencial ou heap
  - A leitura por ordem de chave primária sempre é cara
    - Envolve buscas aleatórias a partir do índice
- Nesse tipo de índice, é comum que a chave se repita.

# Índice Secundário sobre arquivo sequencial

chave	ponteiro
-------	----------

1942	
1972	
1978	
1984	

1994	
1994	
1996	
2017	

2019	
2022	

Arquivo de índice (year)

idM	title	year
-----	-------	------

1	Star Wars	1978
2	Terminator	1984

3	Forrest Gump	1994
4	Toy Story	1996

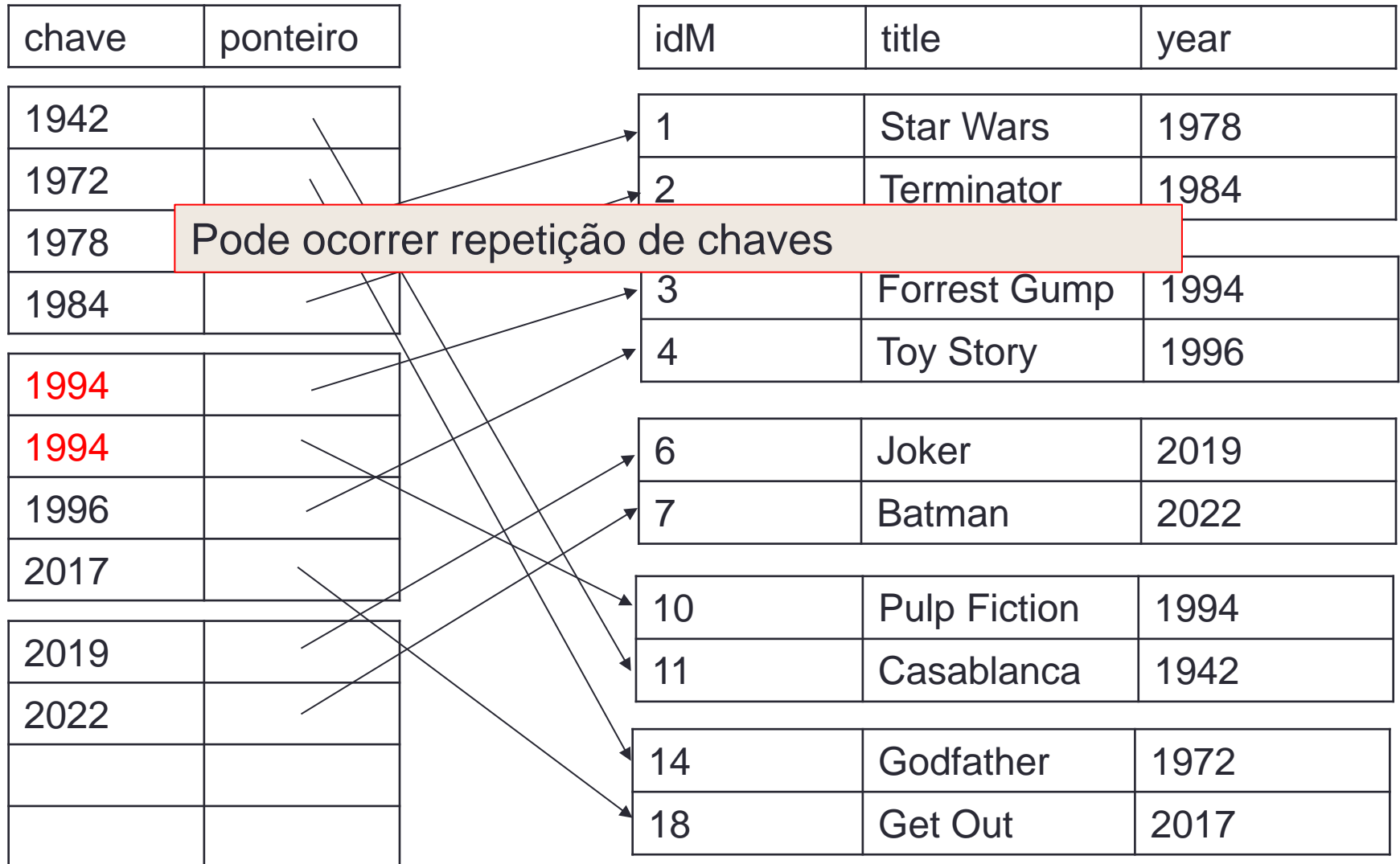
6	Joker	2019
7	Batman	2022

10	Pulp Fiction	1994
11	Casablanca	1942

14	Godfather	1972
18	Get Out	2017

Arquivo de dados sequencial (movie)

# Índice Secundário sobre arquivo sequencial



Arquivo de índice (year)

Arquivo de dados sequencial (movie)

# Índice Secundário sobre arquivo sequencial

chave	ponteiro	idM	title	year	
1942		1	Star Wars	1978	3
1972				4	4
1978					
1984				4	5
				6	7
1994					
1994				9	9
1996		7	Batman	2022	10
2017					
		10	Pulp Fiction	1994	6
2019		11	Casablanca	1942	1
2022					
		14	Godfather	1972	2
		18	Get Out	2017	8

A leitura por ordem de um índice secundário sempre é cara.

O índice leva aos registros ordenados. Porém, as páginas são lidas em uma ordem que não coincide com a sua disposição no meio físico.

Arquivo de índice (year)

Arquivo de dados sequencial (movie)

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível



# Cobertura do índice

- Os índices ordenados também podem ser classificados quanto à cobertura dos registros indexados
  - Esparsos
  - Densos

# Índice esperso

- Contém entradas no índice para apenas alguns valores de chave de busca.
  - Normalmente uma entrada por página de dados
  - A chave no índice deve ser a maior chave que é menor ou igual do que todas as chave presentes na página
- Para localizar um registro com uma chave de busca  $K$ , *deve-se*:
  - Encontrar a entrada com o maior valor de chave de busca  $< K$
  - Varrer o arquivo sequencialmente a partir do registro apontado por essa entrada

# Índice esparsos

chave	ponteiro
-------	----------

1	
3	
6	
...	

...	...
-----	-----

.  
. .  
.

idM	title	year
-----	-------	------

1	Star Wars	1978
2	Terminator	1984

3	Forrest Gump	1994
4	Toy Story	1996

6	Joker	2019
7	Batman	2022

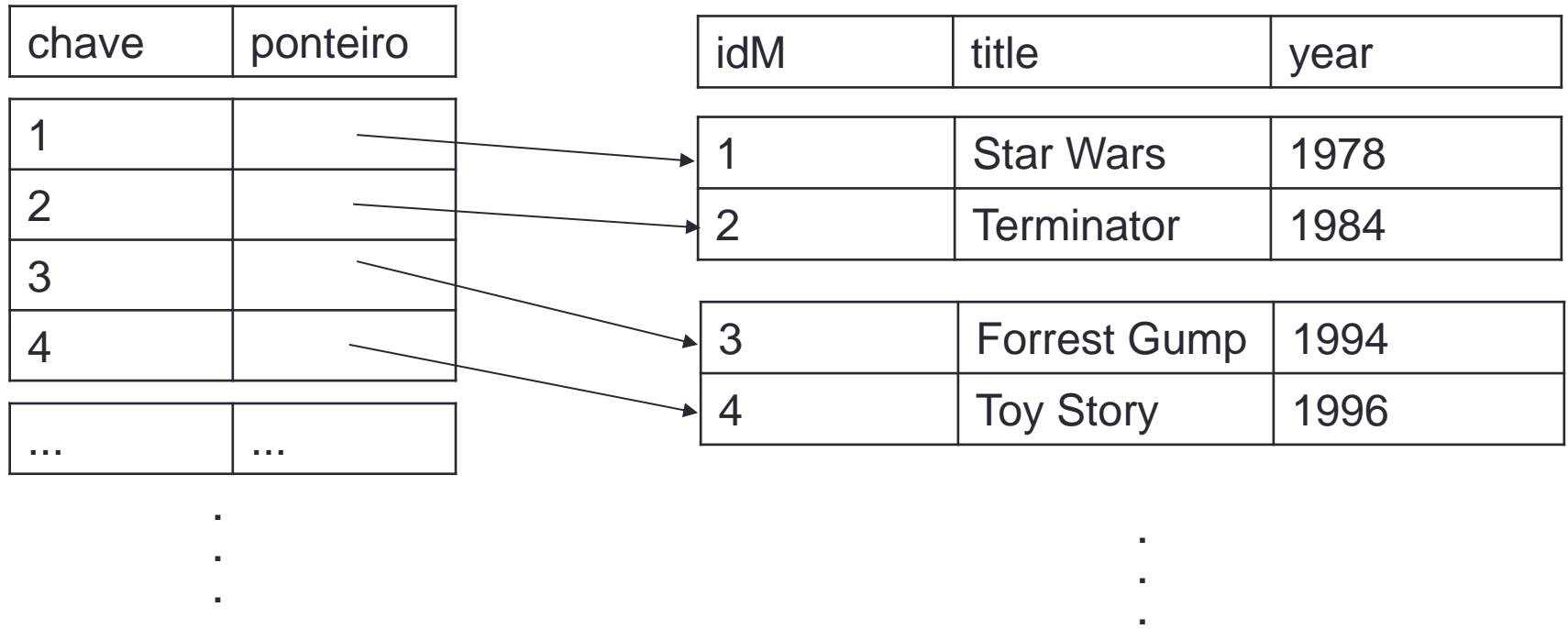
.  
. .  
.

- **Somente** aplicável quando os registros estão ordenados pela chave de busca do índice.
- Por quê?

# Índice denso

- registros do índice aparecem para todos valores de chave de busca do arquivo.
- Para localizar um registro com uma chave de busca  $K$ , *deve-se*:
  - Encontrar a entrada com o valor de chave de busca  $K$
  - Retornar o registro apontado por essa entrada

# Índice denso



- Um índice primário cujo arquivo de dado seja ordenado pela chave primária pode ser denso ou esparso
- Mas índices secundários precisam ser densos
  - Por quê?

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível

# Atualização de índices ordenados

- As atualizações dos índices significam atualizações nas páginas físicas onde os índices estão armazenados
  - A organização física e lógica dessas páginas é feita de forma similar às páginas de um arquivo de dados.
    - Criação de páginas de estouro.
    - Inserção de novas páginas na ordem sequencial
    - Distribuição de dados entre páginas
- Diferentes estratégias podem ser adotadas, dependendo da cobertura do índice (denso ou esperso) .

# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível
  - Árvores B+
  - Índices no DBest

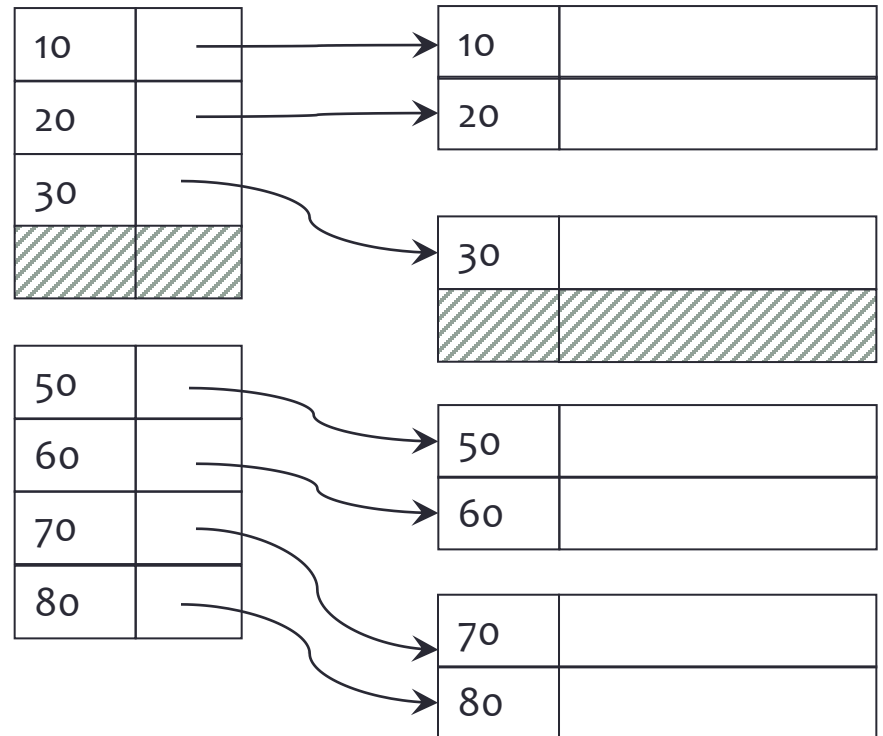
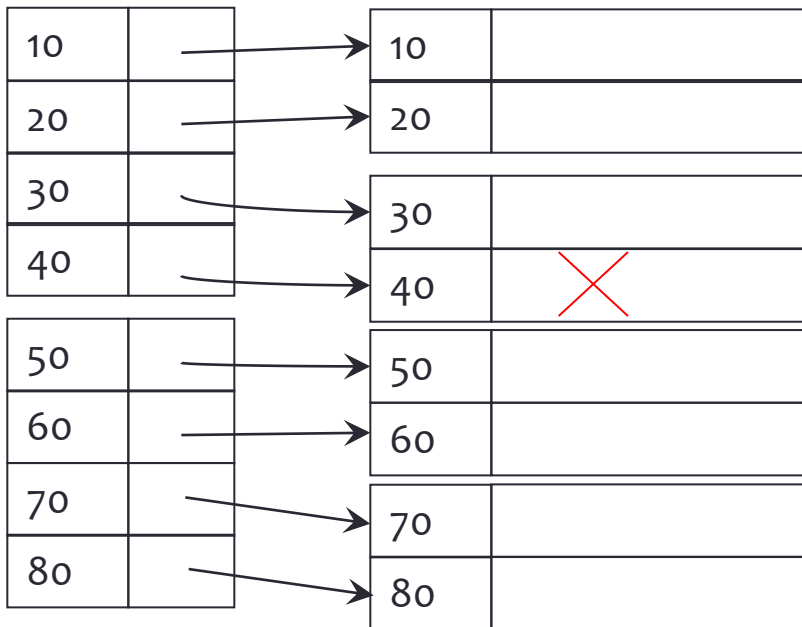


# Remoção

- Se um registro for removido de uma página
- Índices **densos**
  - Remover a entrada correspondente do índice
- Índice **esparso**
  - Mantém-se a entrada correspondente do índice
    - Mesmo se a página ficar vazia
      - estratégia simplificada.
      - A estratégia realmente adotada é mais sofisticada

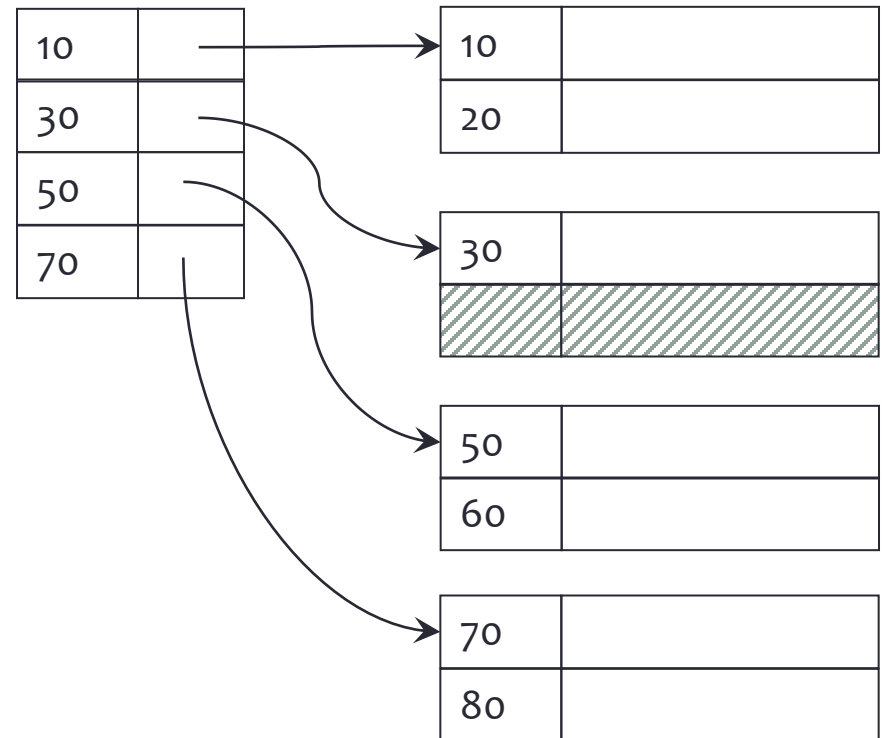
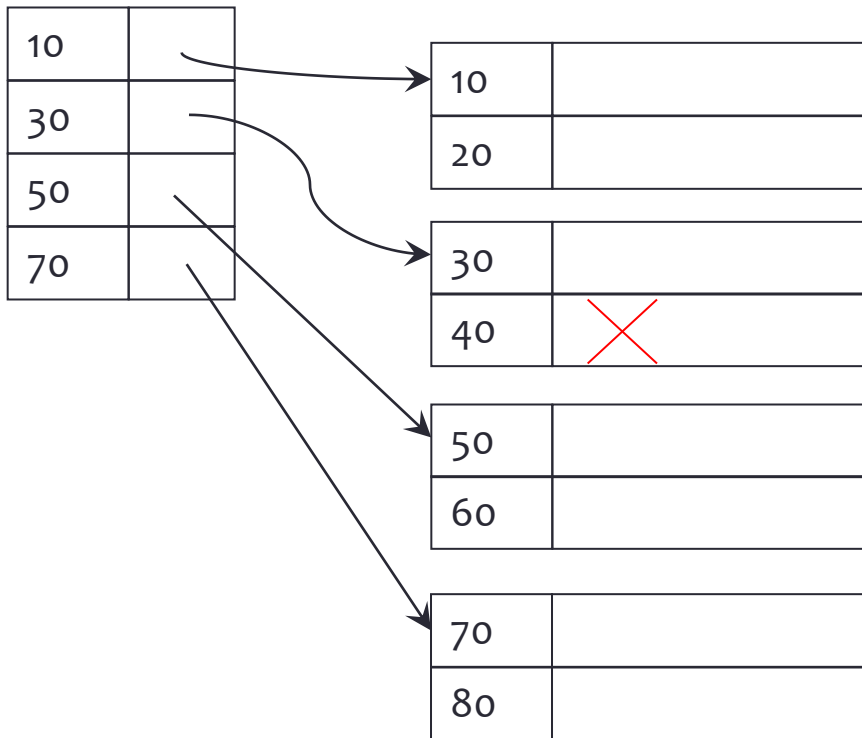
# Remoção

- Índice **denso**:
  - Eliminação de registro com chave de busca **40**



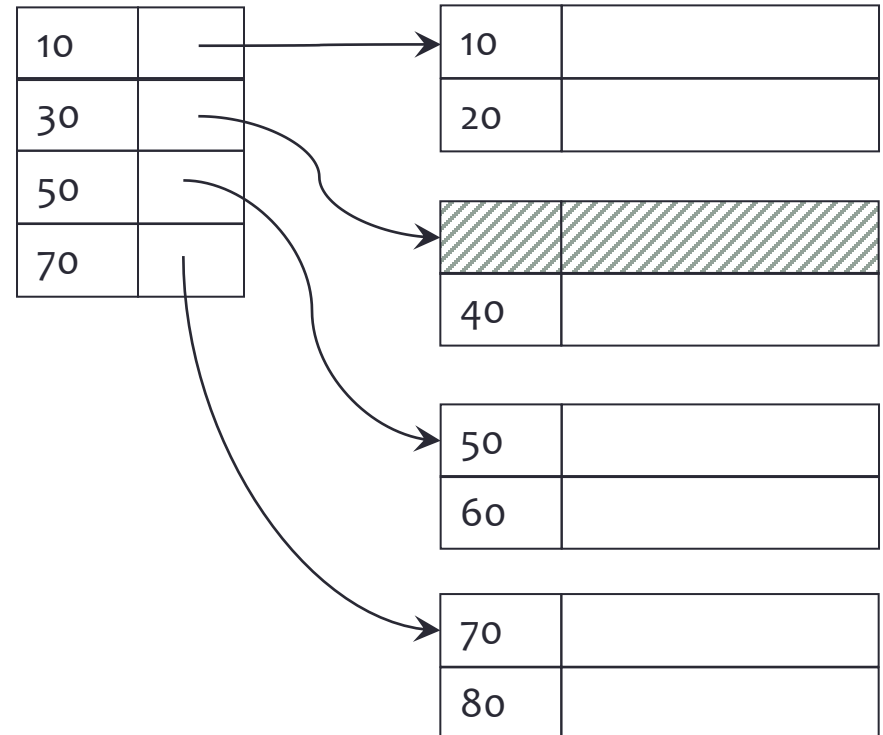
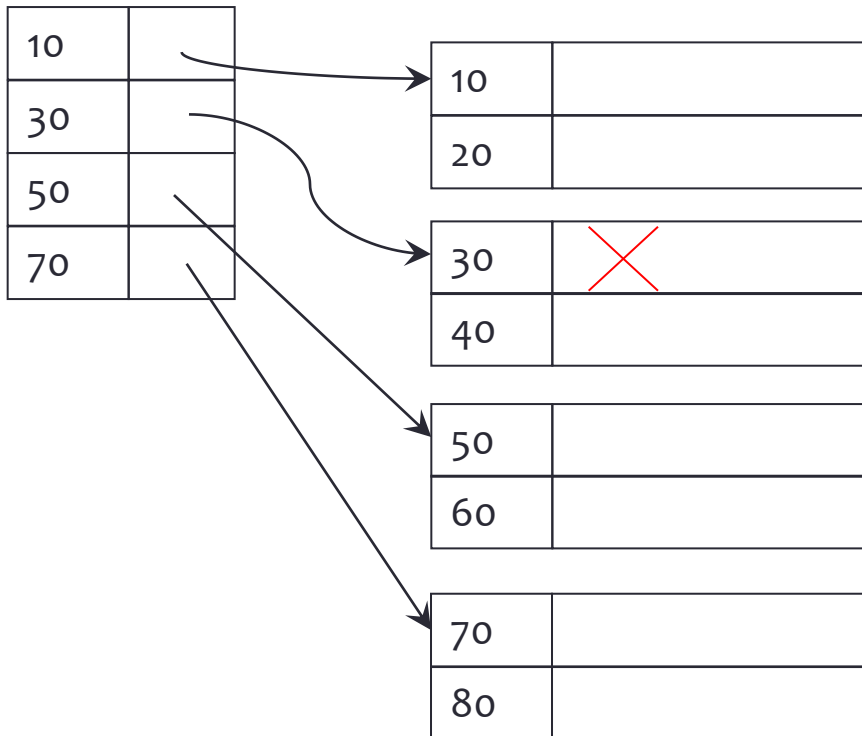
# Remoção

- Índice **esparso**:
  - Eliminação de registro com chave de busca **40**



# Remoção

- Índice **esparso**:
  - Eliminação de registro com chave de busca **30**

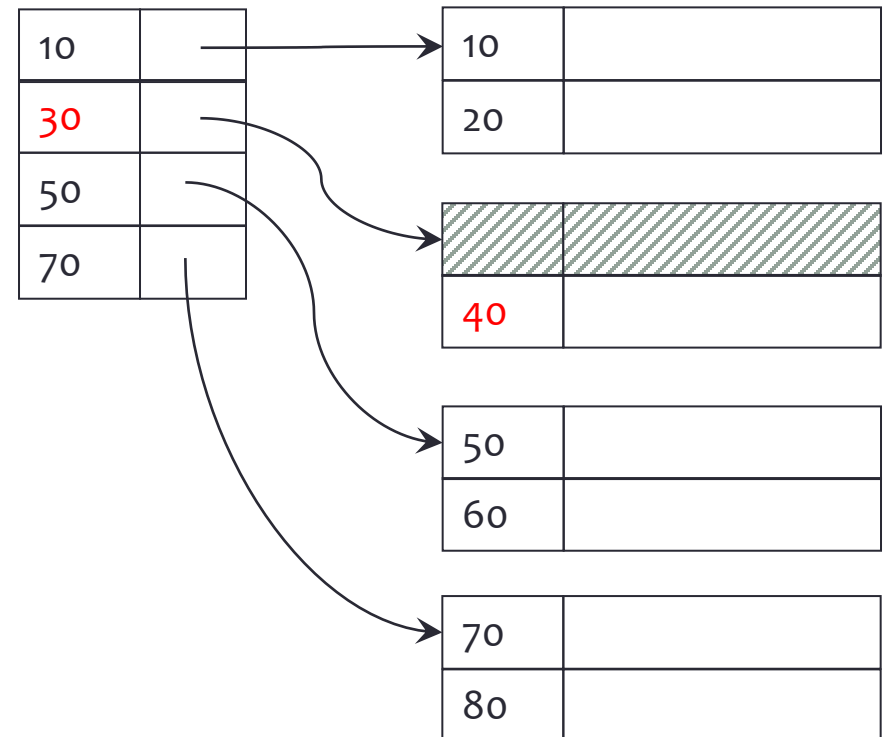
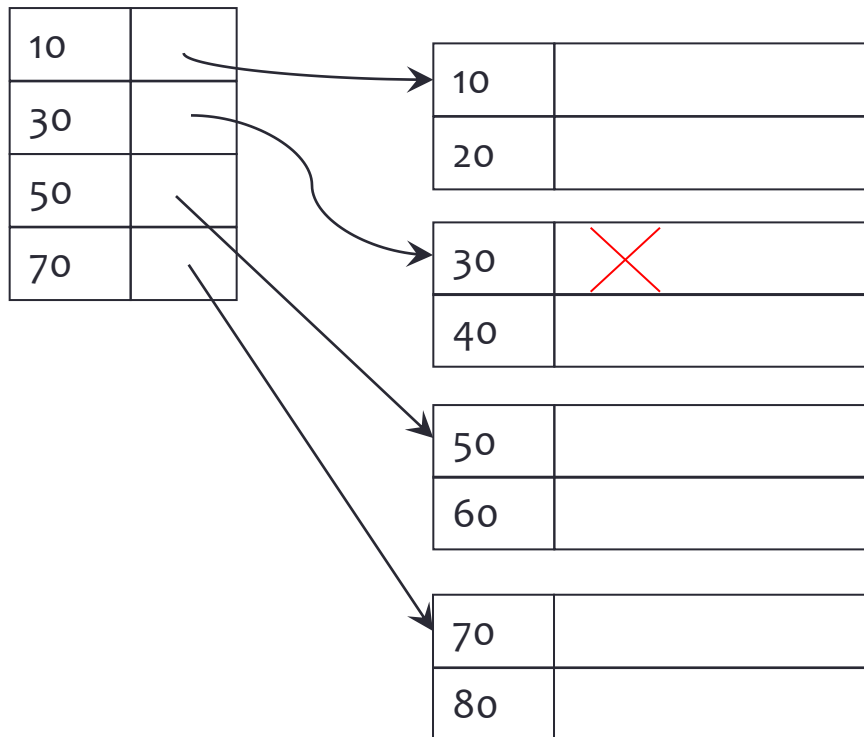


# Remoção

- Índice **esparso**:
  - Eliminação de registros

Observe que a entrada correspondente no índice não foi removida

Isso não é estritamente necessário, pois a entrada continua válida (aponta para uma página onde há registros com **chave > 30**)



# Sumário

- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - **Inserção**
  - Índices Multinível

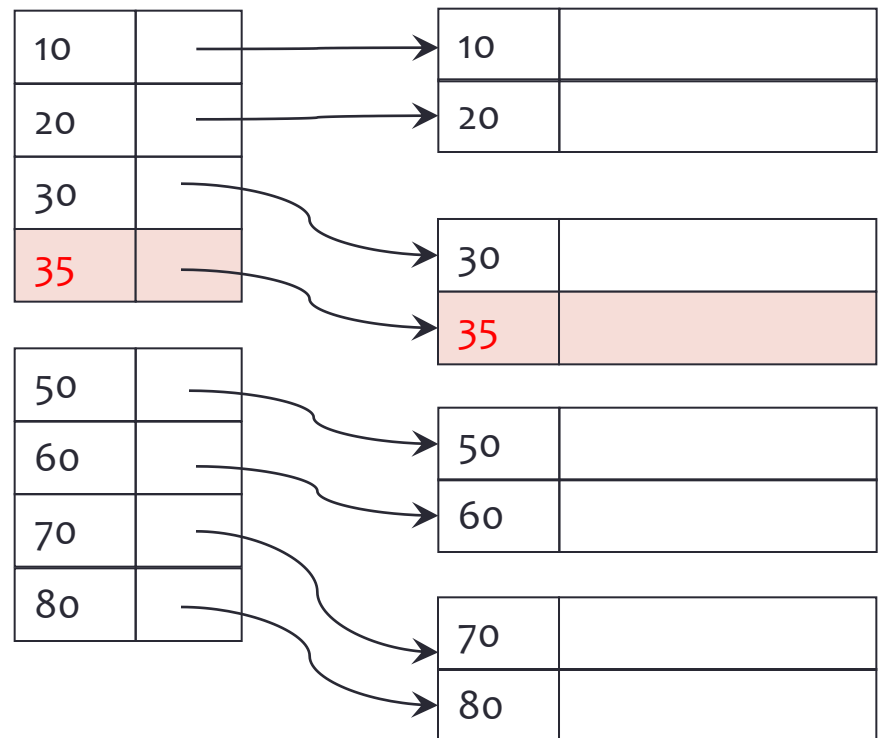
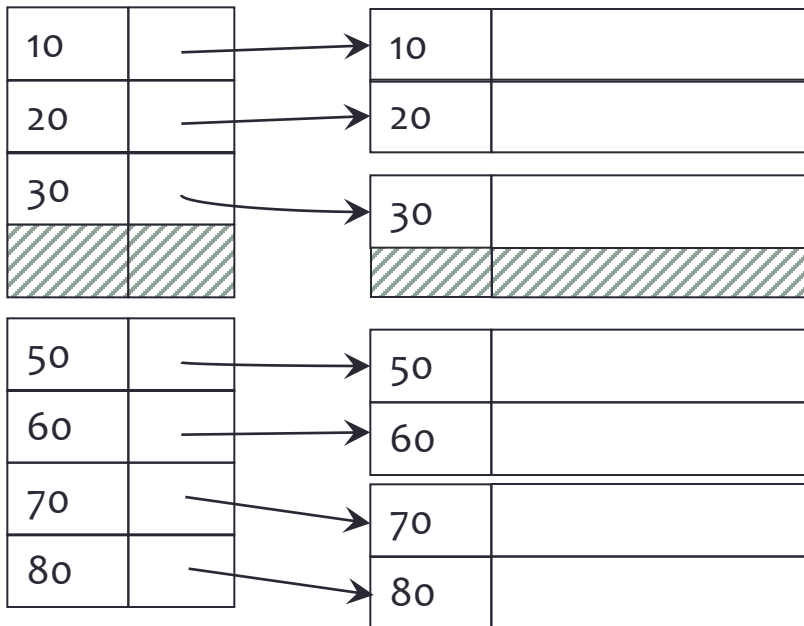
# Inserção

- Se um registro for inserido em uma página
- Índices **densos**
  - Insere-se uma entrada para o novo valor
- Índices **esparsos**
  - O índice deve possuir pelo menos um ponteiro para cada página
  - A chave no índice deve ser a maior chave que seja menor ou igual do que todas as chaves presentes na página
  - Caso o registro sendo inserido seja o menor, talvez seja necessário atualizar o índice

# Inserção

- Índice **denso**

- Inserção de registro com chave de busca **35**

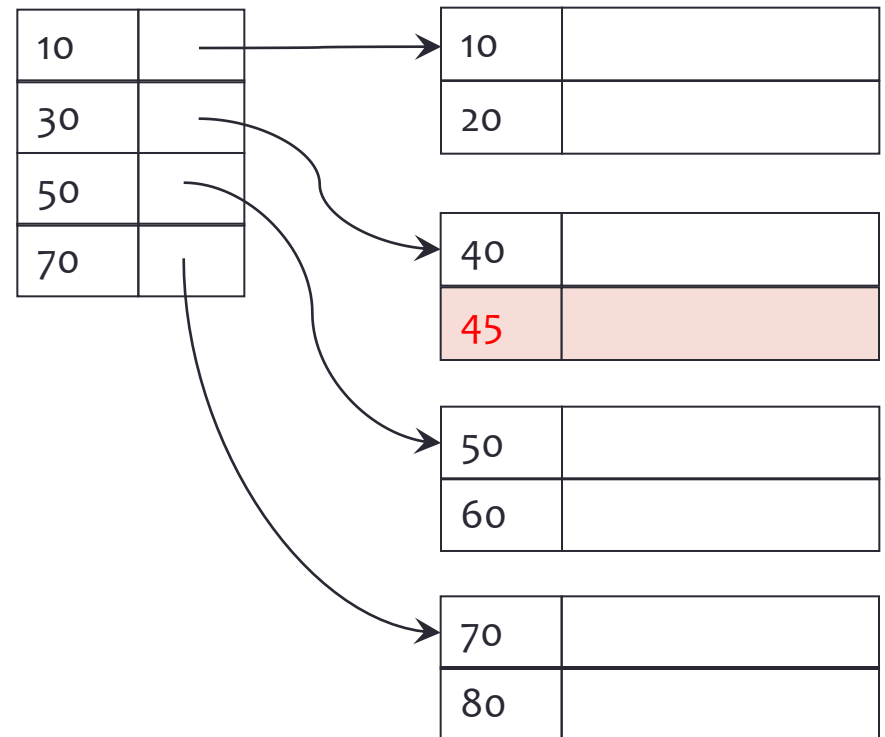
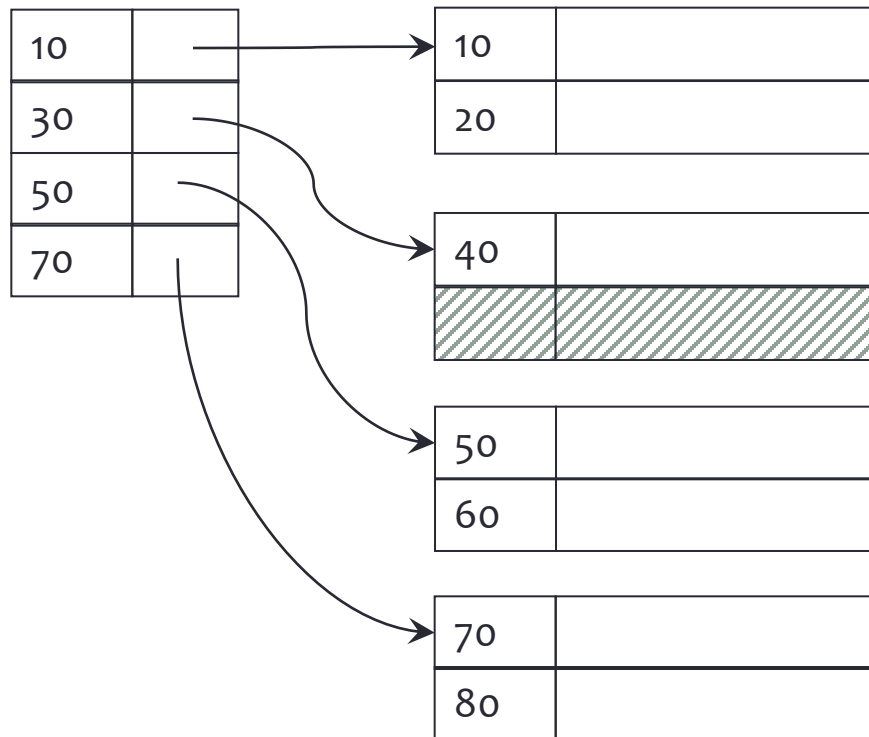




# Inserção

- Índice **esparso**

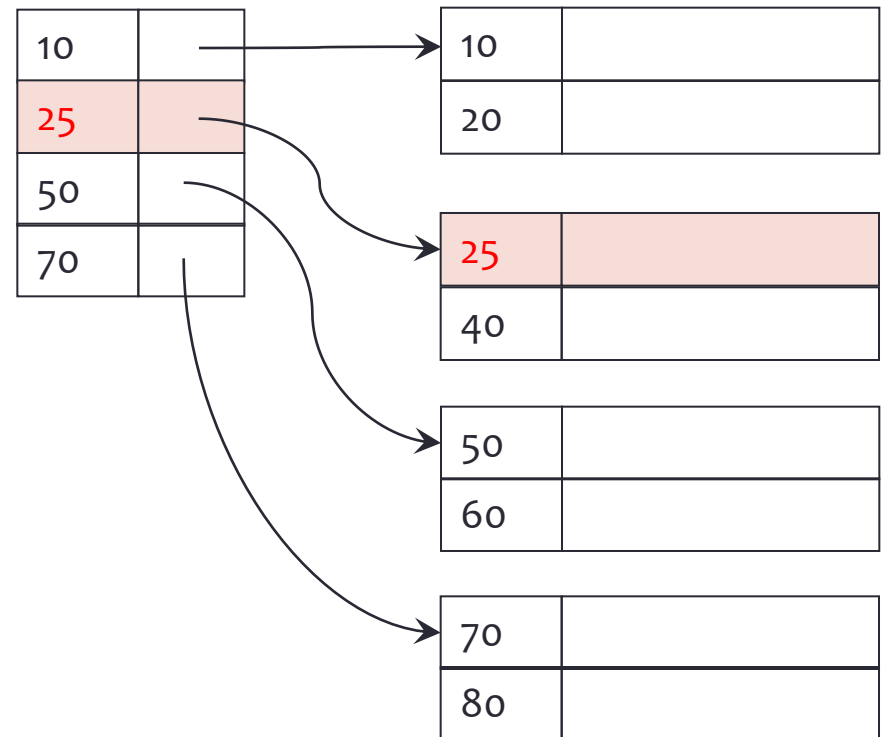
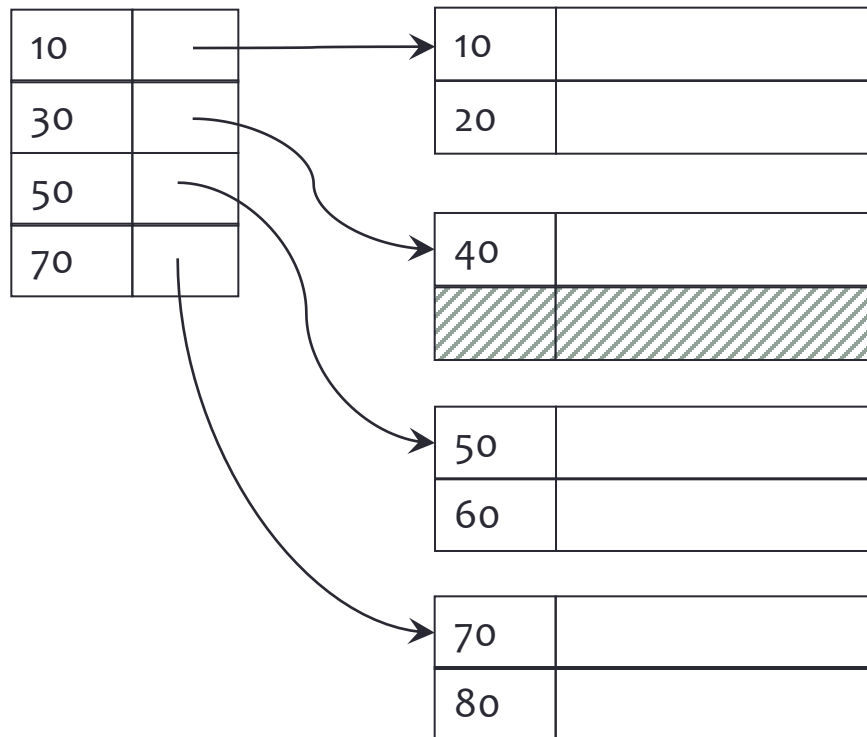
- Inserção do registro com chave de busca **45**



# Inserção

- Índice **esparso**

- Inserção do registro com chave de busca **25**

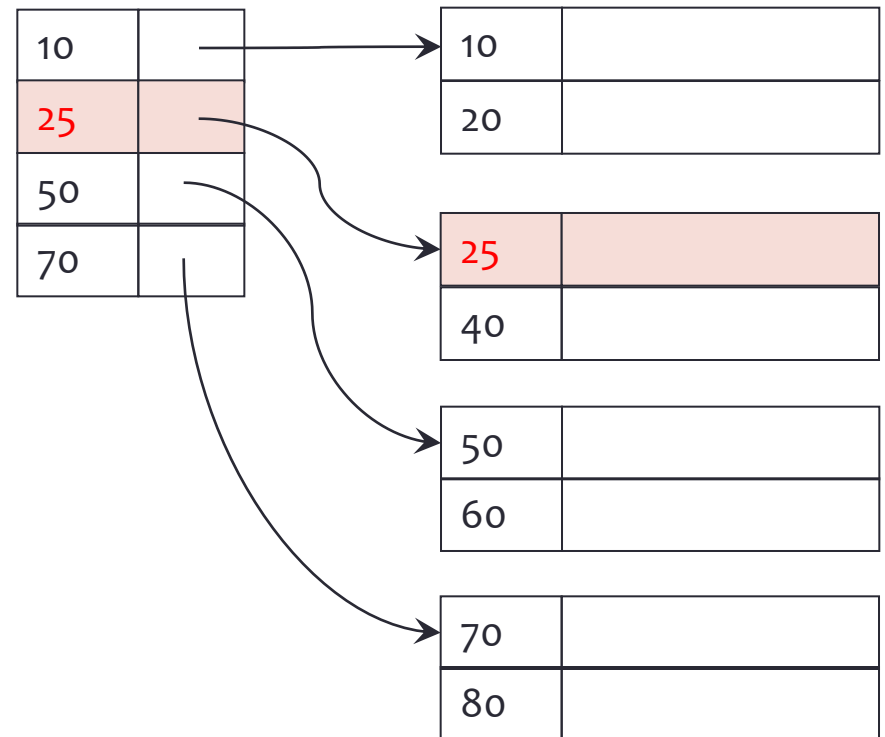
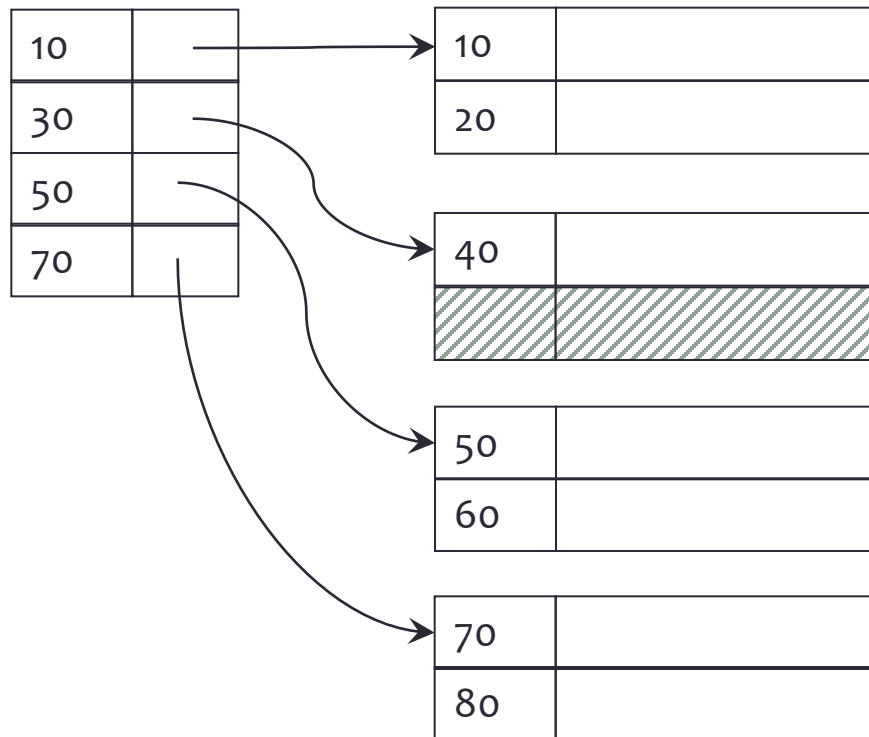


# Inserção

O índice teve que ser atualizado, uma vez que a página agora guarda uma chave menor do que 30.

- Índice **esparso**

- Inserção do registro com chave de busca **25**



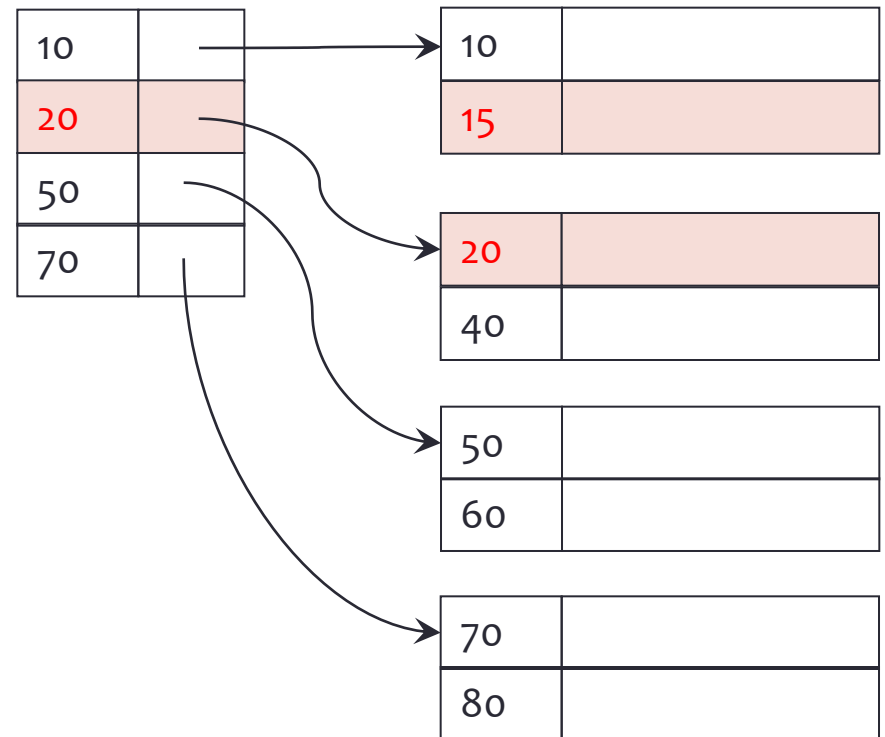
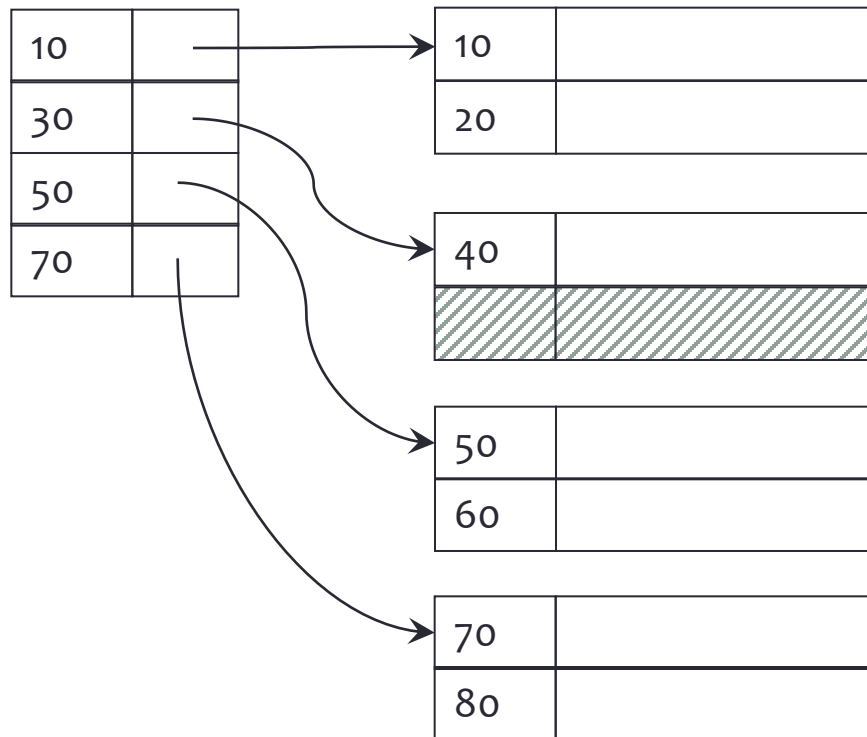
# Inserção

- Em alguns casos, a inserção pode provocar a geração de uma página de estouro
  - Seja nos dados
  - Ou nos índices densos/esparsos
- Pode-se evitar páginas de estouro através da reorganização imediata do arquivo
  - A reorganização é viável se muitas páginas tiverem que ser modificadas

# Inserção

- Índice **Esperso**

- Inserção do registro **15** – com reorganização imediata

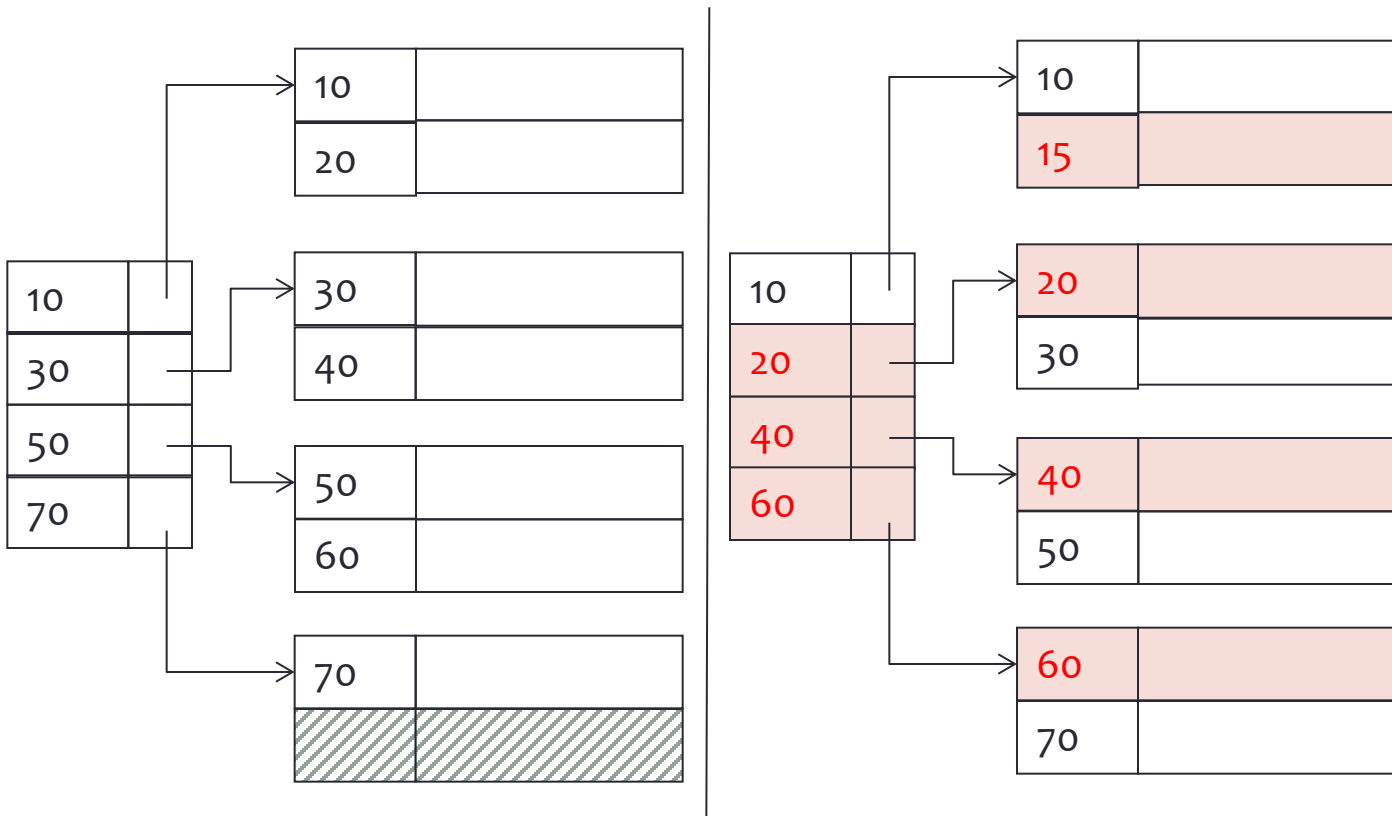


Nesse caso é VIÁVEL

# Inserção

- Índice **Esparso**

- Inserção do registro **15** – com reorganização imediata

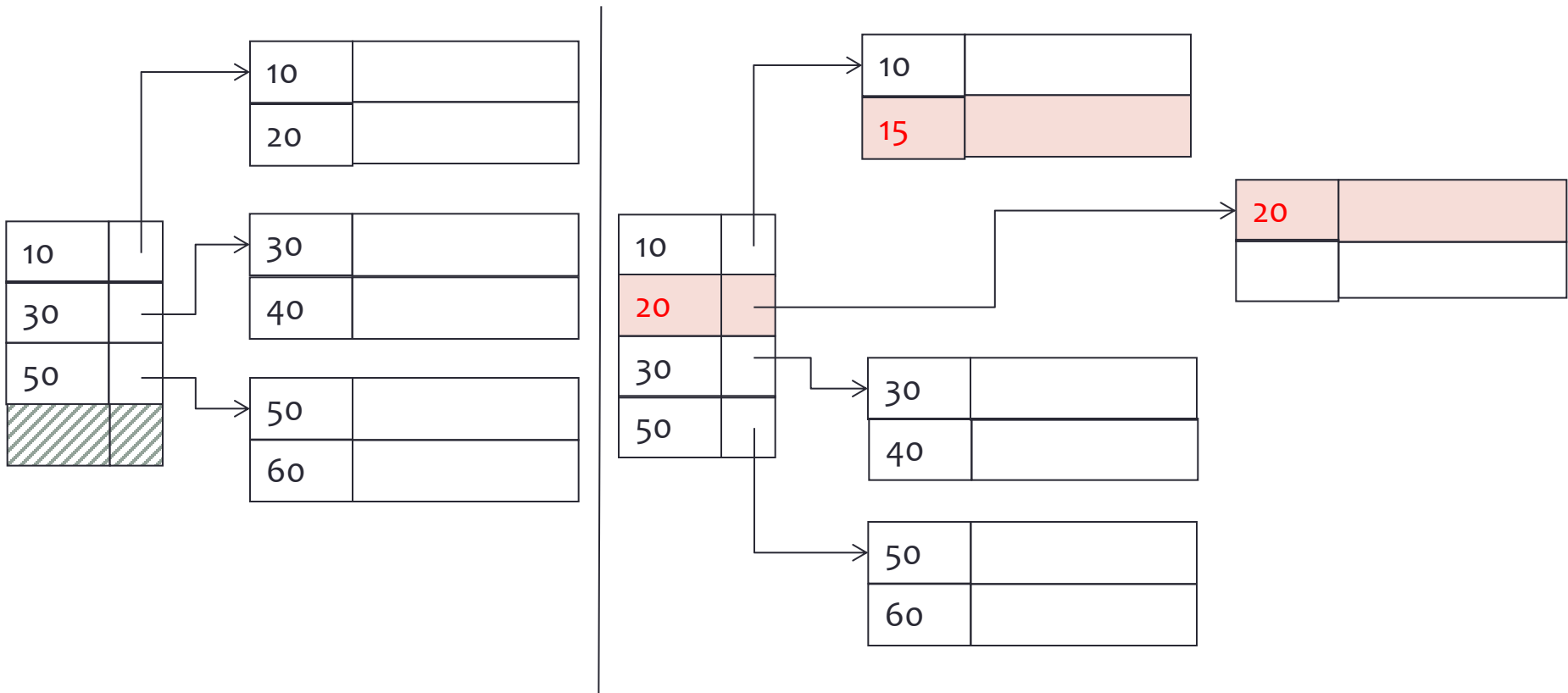


Nesse caso é MUITO CARO!

# Inserção

- Índice **Esparso**

- Inserção do registro **15** – usando páginas de estouro



# Sumário

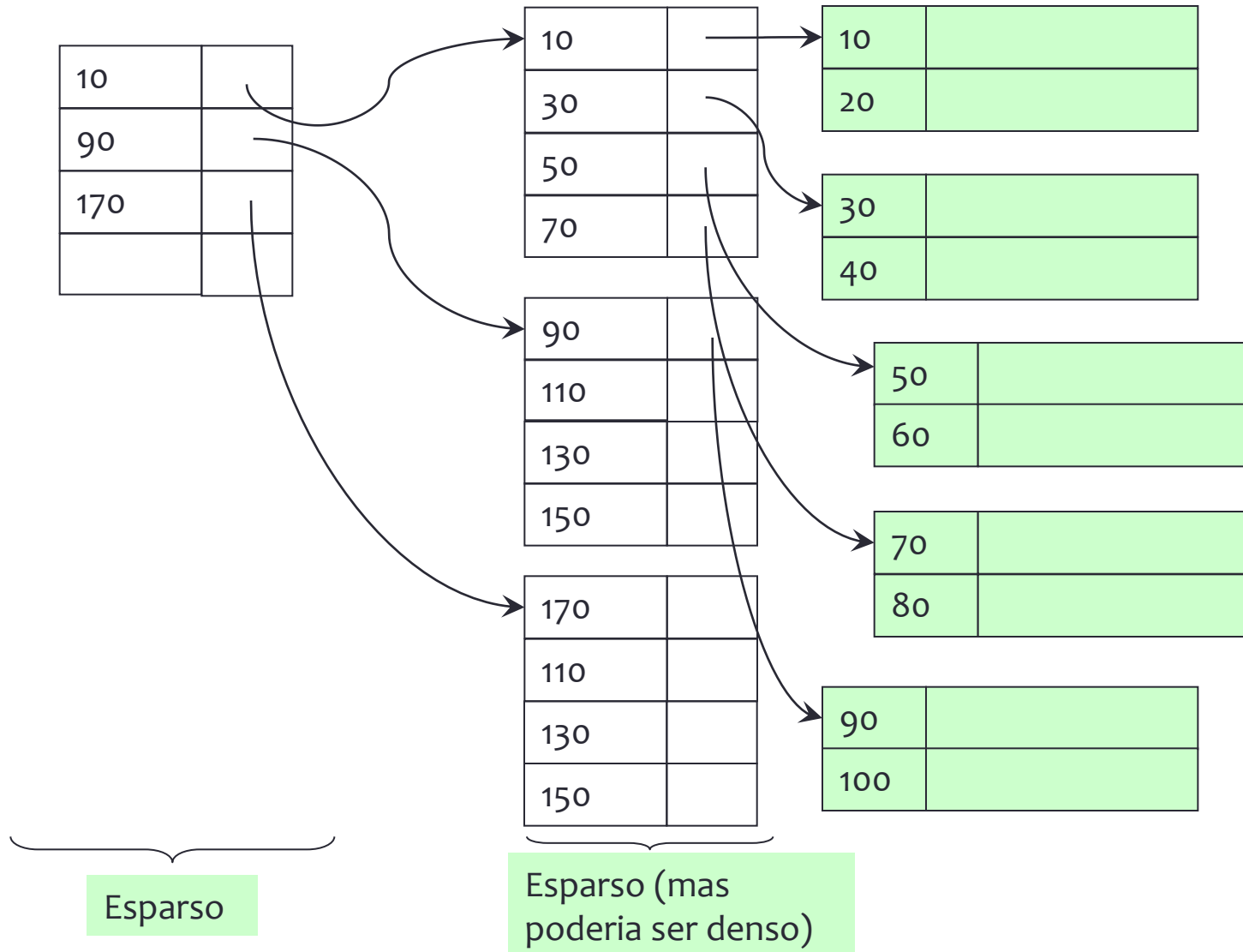
- Introdução
- Índices Hash
- Índices Ordenados
  - Tipo da chave de busca
    - Índice primário
    - Índice secundário
  - Cobertura do índice
  - Estratégias de atualização
    - Remoção
    - Inserção
  - Índices Multinível



# Índices multinível

- Se um índice não couber na memória, o acesso se torna caro.
- Solução: tratar o índice em disco como um arquivo sequencial e construir um índice esparsosobre ele.
  - Índice interno – O arquivo do índice
  - Índice externo – um índice esparsodo índice interno
- Se mesmo o índice externo for muito grande para caber na memória, outro nível de índice pode ser criado, e assim por diante.
- Índices em todos os níveis devem ser atualizados quando ocorrer atualizações no arquivo.

# Índices multinível



# Índices multinível

- Em índices multinível, as inserções e remoções são simples extensões dos algoritmos usados em índices de um nível só
  - Inserções e remoções são propagadas dos níveis internos para os externos

# Encerrando

- Índices oferecem benefícios substanciais na procura por registros
  - **mas** as atualizações em índices impõem uma sobrecarga
  - quando o arquivo é modificado
    - os índices sobre o arquivo também precisam ser