

Algoritmos e Estruturas de Dados III

Aula 4.1 Arq. Indexados

Prof. Hayala Curto
2022



PUC Minas

Arquivo Indexado

- Arquivos indexados são arquivos em que os registros são acessados de forma aleatória. Para que a busca seja eficiente, esses arquivos contam com estruturas de dados adicional de apoio (chamadas índices).
- Assim, a busca não é realizada diretamente no arquivo de dados, mas no tal índice, que retornará quais registros atendem ao critério de busca e onde eles estão localizadas no arquivo de dados.
- Um arquivo indexado é um arquivo que possui um ou mais índices que permitem acesso aleatório a um registro, dada uma determinada chave.

Arquivo Indexado

Um arquivo pode ter vários índices, cada um baseado em um atributo (ou conjunto de atributos) diferente.

Em um sistema de gerenciamento de clientes, é possível termos, por exemplo, um índice baseado no ID de cliente, outro baseado no nome do cliente e ainda um terceiro baseado no e-mail do cliente.

Arquivo Indexado

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	7	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	13	Zelda	Nintendo	300,00
253	15	The Sims 4	PC	99,00

Arquivo Indexado pela chave primária

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	15	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	7	Zelda	Nintendo	300,00
253	13	The Sims 4	PC	99,00

Pos.	ID	Pos.
0	2	4
12	3	60
24	7	182
36	8	135
48	13	253
60	15	90

Arquivo Indexado pelo título

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	15	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	7	Zelda	Nintendo	300,00
253	13	The Sims 4	PC	99,00

Pos.	Título	Pos.
0	Fortnite	135
12	GTA V	60
24	Mario	90
36	Minecraft	4
48	The Sims 4	253
60	Zelda	182

Tipos de Índices



Tipos de índices

- Primários ou secundários
- Diretos ou indiretos
- Densos ou esparsos

Índices primários ou secundários

- **Índices primários:** seguem a mesma ordem do arquivo de dados
- **Índices secundários:** não seguem a mesma ordem do arquivo de dados

Índices primários ou secundários

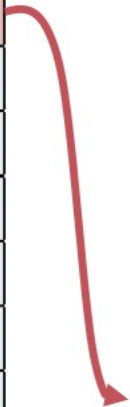
- **Índices primários: seguem a mesma ordem do arquivo de dados**
- Índices secundários: não seguem a mesma ordem do arquivo de dados

	ID	End		ID	Dados
0	10	4		4	10
12	20	60		60	20
24	30	96		96	30
36	40	137	→	137	40
48	50	182		182	50
60	60	223		223	60
72	70	252		252	70
84	80	360		360	80
96	90	415		415	90
108	100	484		484	100

Índices primários ou secundários

- Índices primários: seguem a mesma ordem do arquivo de dados
- **Índices secundários: não seguem a mesma ordem do arquivo de dados**

	ID	End		ID	Dados
0	10	415		4	20
12	20	4		60	50
24	30	96		96	30
36	40	484		137	90
48	50	60		182	100
60	60	360		223	80
72	70	252		252	70
84	80	223		360	60
96	90	137		415	10
108	100	182		484	40




Índices diretos ou indiretos

- **Índices diretos:** apontam diretamente para a posição do registro no arquivo de dados
- **Índices indiretos:** apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)

Índices diretos ou indiretos

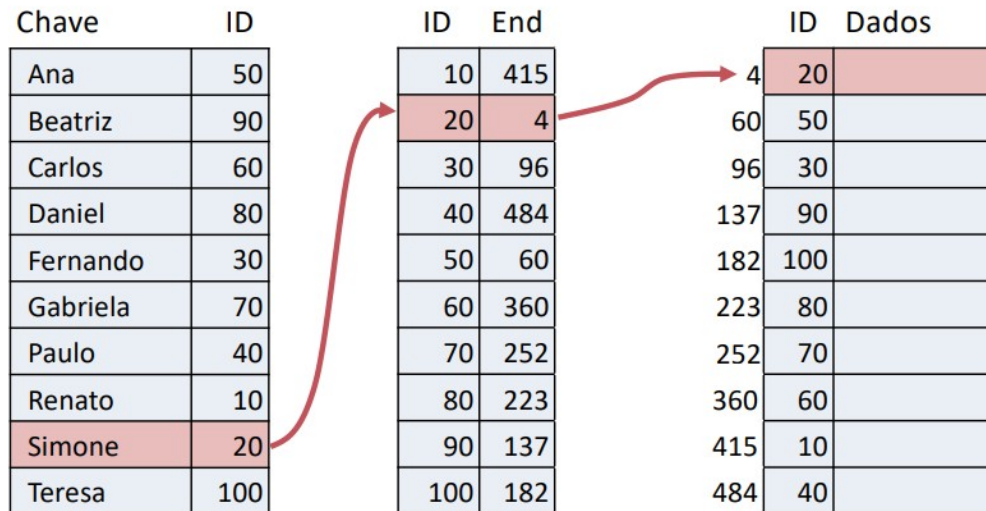
- **Índices diretos:** apontam diretamente para a posição do registro no arquivo de dados
- Índices indiretos: apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)

	ID	End		ID	Dados
0	10	415		4	20
12	20	4		60	50
24	30	96		96	30
36	40	484		137	90
48	50	60		182	100
60	60	360		223	80
72	70	252		252	70
84	80	223		360	60
96	90	137		415	10
108	100	182		484	40



Índices diretos ou indiretos

- Índices diretos: apontam diretamente para a posição do registro no arquivo de dados
- **Índices indiretos: apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)**



Índices densos ou esparsos

- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- **Índices esparsos:** possuem entradas para apenas alguns registros

Índices densos ou esparsos

- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- Índices esparsos: possuem entradas para apenas alguns registros

0	10	4
12	20	60
24	30	96
36	40	137
48	50	182
60	60	223
72	70	252
84	80	360
96	90	415
108	100	484

4	10	
60	20	
96	30	
137	40	
182	50	
223	60	
252	70	
360	80	
415	90	
484	100	

Índices densos ou esparsos

- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- **Índices esparsos:** possuem entradas para apenas alguns registros

0	10	4	→	4	10	
12	40	182	↘	60	20	
24	80	415	↘	96	30	
				137	40	
				182	50	
				223	60	
				252	70	
				360	80	
				415	90	
				484	100	

Operações em Arquivos Indexados CRUD



- As operações realizadas em arquivos indexados são as mesmas realizadas nos arquivos sequenciais.
- A principal diferença aqui, porém, é que as buscas por quaisquer entidades não serão feitas sequencialmente no arquivo dados, mas serão feitas em um dos índices.
 - Para que isso funcione corretamente, os índices devem sempre estar atualizados,

Interface do CRUD

- $ID \leftarrow \text{arquivo.create}(\text{*novo_objeto*})$
- $\text{objeto} \leftarrow \text{arquivo.read}(ID)$
- $\text{ok} \leftarrow \text{arquivo.update}(\text{*objeto_atualizado*})$
- $\text{ok} \leftarrow \text{arquivo.delete}(ID)$

Observação: A interface do CRUD não deve oferecer acesso direto aos índices usados

CREATE

```
01: algoritmo create(objeto)
02:   mover o ponteiro para início do arquivo (cabeçalho)
03:   ler últimoID
04:   objeto.ID  $\leftarrow$  últimoID + 1
05:   mover o ponteiro para início do arquivo
06:   escrever objeto.ID
07:   criar registro para o objeto
08:   mover para o fim do arquivo
09:   pos  $\leftarrow$  posição do ponteiro
10:   escrever registro
11:   inserir o par (objeto.ID, pos) no índice
12: fim-algoritmo
```

READ

```
01: algoritmo read(ID)
02:   pos ← buscar o ID no índice
03:   se pos ≠ -1
04:     mover ponteiro para pos
05:     ler registro
06:     se registro.lapide ≠ '*' // checagem dupla
07:       então extrair objeto do registro
08:       retornar objeto e terminar
09:     fim-se
10:   fim-se
11:   retornar objeto vazio // null
12: fim-algoritmo
```

DELETE

```
01: algoritmo delete(ID)
02:   pos ← buscar o ID no índice
03:   se pos ≠ -1
04:     então mover ponteiro para pos
05:       ler registro
06:       se registro.lapide ≠ '*'
07:         então extrair objeto do registro
08:           mover para pos
09:           escrever lápide como excluído
10:           remover o ID do índice
11:           retornar verdadeiro e terminar
12:         fim-se
13:       fim-se
14:     retornar falso
15: fim-algoritmo
```

UPDATE

Upd

```
01: algoritmo update(novoObjeto)
02:   pos ← buscar o ID no índice
03:   se pos ≠ -1
04:     então mover ponteiro para pos
05:     ler registro
06:     se registro.lapide ≠ '*'
07:       então extrair objeto do registro
08:       criar novoRegistro para novoObjeto
09:       se novoRegistro.tamanho ≤ registro.tamanho
10:         então mover para pos
11:           escrever novoRegistro (manter ind.tam.)
12:         senão mover para pos
13:           escrever lápide como excluído
14:           mover para fim do arquivo
15:           pos ← posição do ponteiro
16:           escrever novoRegistro
17:           atualizar o endereço para o ID no índice
18:           retornar verdadeiro e terminar
19:       fim-se
20:     fim-se
21:   fim-se
22:   retornar falso
23: fim-algoritmo
```


Considerações importantes

- O algoritmo apresentado possui apenas um índice secundário (por ID), direto e denso. Se os arquivos tiverem mais índices, esse algoritmo deve ser modificado.
- Os índices podem ser gerenciados fora do CRUD, mas isso requer muita atenção extra.

Considerações sobre Relacionamentos



Relacionamento

Um relacionamento é uma associação entre duas entidades em um banco de dados

Exemplos:

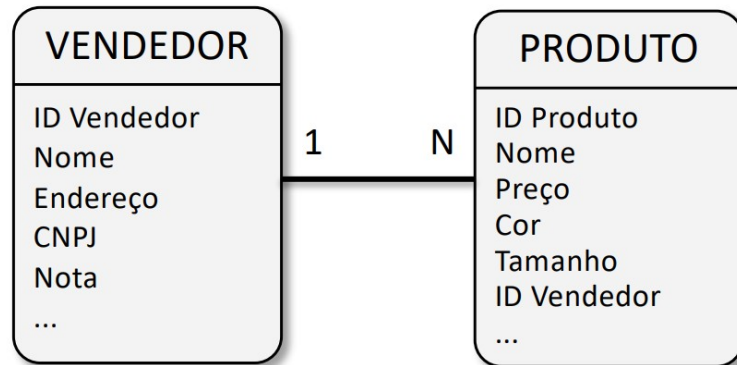
- Produtos de um vendedor
- Compromissos de um usuário
- Operações bancárias de uma conta corrente (de um correntista)

Relacionamento

Um relacionamento é uma associação entre duas entidades em um banco de dados

RELACIONAMENTO 1:N

- Cada vendedor possui N produtos
- Cada produto pertence a 1 vendedor
- O relacionamento é feito por meio de chaves estrangeiras



Tipos de relacionamento

- **Um para um (1:1)**

- Liga uma entidade de um tipo a uma outra entidade de outro tipo
- Exemplo: pessoa 1:1 habitação

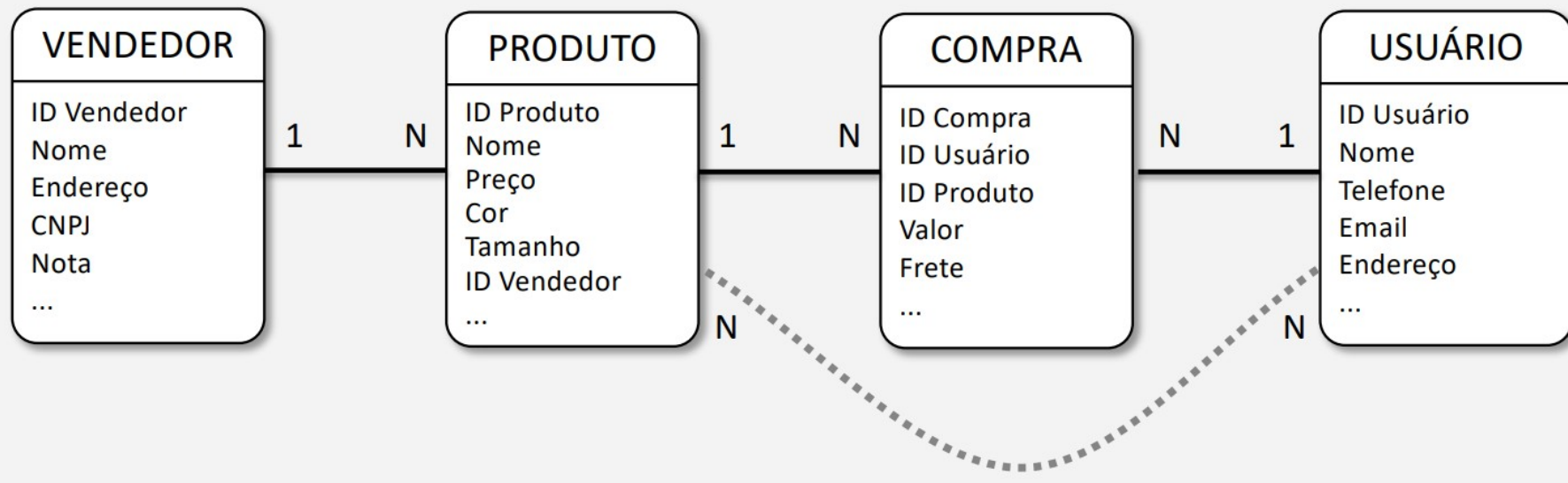
- **Um para muitos (1:N)**

- Uma entidade de um tipo está ligada a várias outras entidades de outro tipo
- Exemplo: categoria 1:N produto

- **Muitos para muitos (N:N)**

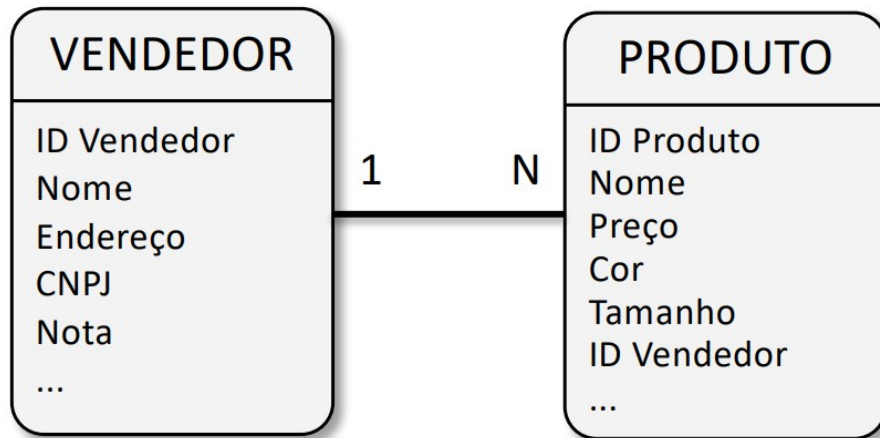
- Várias entidades de um tipo estão ligadas a várias entidades de outro tipo
- São relacionamentos que geralmente dependem de uma entidade intermediária.
- Exemplo: ator N:N filme

Relacionamentos



Observações importantes

- Quanto usamos relacionamentos, precisamos nos preocupar com a **integridade dos dados**.
- Os códigos do CRUD precisam ser alterados para garantir essa integridade.
- Muitas vezes, a **exclusão** será substituída por uma desativação.
 - ex.: vendas realizadas / produtos excluídos



Não podemos excluir um vendedor que tenha produtos cadastrados