

# Arquivos sequenciais

---

# Arquivos sequenciais

- Arquivos em que os registros são acessados na ordem em que estão armazenados.
  - Normalmente são usados quando há poucas (ou nenhuma) movimentação de registros.
- O objetivo é o acesso rápido a um grande volume de registros.

# Exemplo

Código	Nome	Email	Data Nascimento
1	Carlos Chagas	chagas@gmail.com	13/12/1977
2	Ana Maria Fonseca	ana@hotmail.com	17/04/1975
3	Sílvia Costa	silvio@gmail.com	28/01/1969
4	Joana Torres	joana@gmail.com	31/07/1982

Obs.:

1. Neste exemplo, a ordem dos registros é a ordem de inclusão
2. Código numérico, sequencial e não significativo

**Chave de ordenação**

# Chave de ordenação

- A chave de ordenação é o conjunto de critérios que estabelece a ordem dos registros
- Tipos de chave
  - Campo (ex.: CPF)
  - Combinação de campos (ex.: estado+cidade)
  - Processamento de campos (ex.: inversão de data)

# Chave de ordenação

Código	Nome	Email	Data Nascimento
2	Ana Maria Fonseca	ana@hotmail.com	17/04/1975
1	Carlos Chagas	chagas@gmail.com	13/12/1977
4	Joana Torres	joana@gmail.com	31/07/1982
3	Sílvio Costa	silvio@gmail.com	28/01/1969

Obs.:

Será necessária alguma ação para manter a ordem dos registros

# Chave de ordenação

Código	Nome	Email	Data Nascimento
1	Carlos <b>Chagas</b>	chagas@gmail.com	13/12/1977
3	Sílvio <b>Costa</b>	silvio@gmail.com	28/01/1969
2	Ana Maria <b>Fonseca</b>	ana@hotmail.com	17/04/1975
4	Joana <b>Torres</b>	joana@gmail.com	31/07/1982

Chave de ordenação
Chagas
Costa
Fonseca
Torres

# Chave de ordenação

Código	Nome	Email	Data Nascimento
3	Sílvio Costa	silvio@gmail.com	<b>28/01/1969</b>
2	Ana Maria Fonseca	ana@hotmail.com	<b>17/04/1975</b>
1	Carlos Chagas	chagas@gmail.com	<b>13/12/1977</b>
4	Joana Torres	joana@gmail.com	<b>31/07/1982</b>

Chave de ordenação
1969 01 28
1975 04 17
1977 12 13
1982 07 31



# Identificadores

# Identificadores (ou códigos)

- Características de um bom identificador
  - Numérico
    - Melhor aproveitamento dos valores dos bytes
  - Sequenciais
    - Evita o desperdício de valores numéricos
    - Sugerem a ordem de criação dos registros
  - Exclusivos
    - Identificadores não podem ser ambíguos
  - Não significativos
    - Identificadores não podem ser alterados

# Faixa numérica

- **short**

- Faixa numérica: -32.768 a 32.767

- Faixa sem sinal: 0 a 65.535

- Ao gravar no arquivo:

- ```
short codigoShort = (short)(codigoInt - 32768);
```

- Ao ler do arquivo:

- ```
int codigoInt = codigoShort + 32768;
```

- Obs.: A entrada de dados deve estar limitada à faixa

# Cabeçalho do arquivo

CÓDIGO; NOME; IDADE

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		3				1			13		'C'	'A'	'R'	'L'	'O'	'S'	' '	'C'	'H'	'A'

Último  
código  
usado

Byte	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
	'G'	'A'	'S'		25		2			9		'A'	'N'	'A'	' '	'M'	'A'	'R'	'I'	'A'

Byte	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
		31			3			13	'S'	'Í'		'L'	'V'	'I'	'O'	' '	'C'	'O'	'S'	'T'

Byte	60	61	62
	'A'		27

# **Operações em arquivos sequenciais**

# Operações

- Busca
- Inclusão
- Alteração
- Exclusão
- Listagem  
(ou qualquer outro processamento sequencial)
- Reorganização  
(ou reordenação)

# Busca

- Registros de tamanho fixo
  - Busca sequencial
  - Busca binária
- Registros de tamanho variável
  - Busca sequencial
  - Uso de índices → Arquivos indexados

# Busca sequencial em arquivo não ordenado

```
01: algoritmo Busca
02:     ler a chave de busca
03:     ir para o primeiro registro do arquivo
04:     enquanto não atingir o fim do arquivo
05:         ler o próximo registro
06:         se chave de busca = chave do registro
07:             então retornar registro e terminar
08:         fim-se
09:     fim-enquanto
10: fim-algoritmo
```



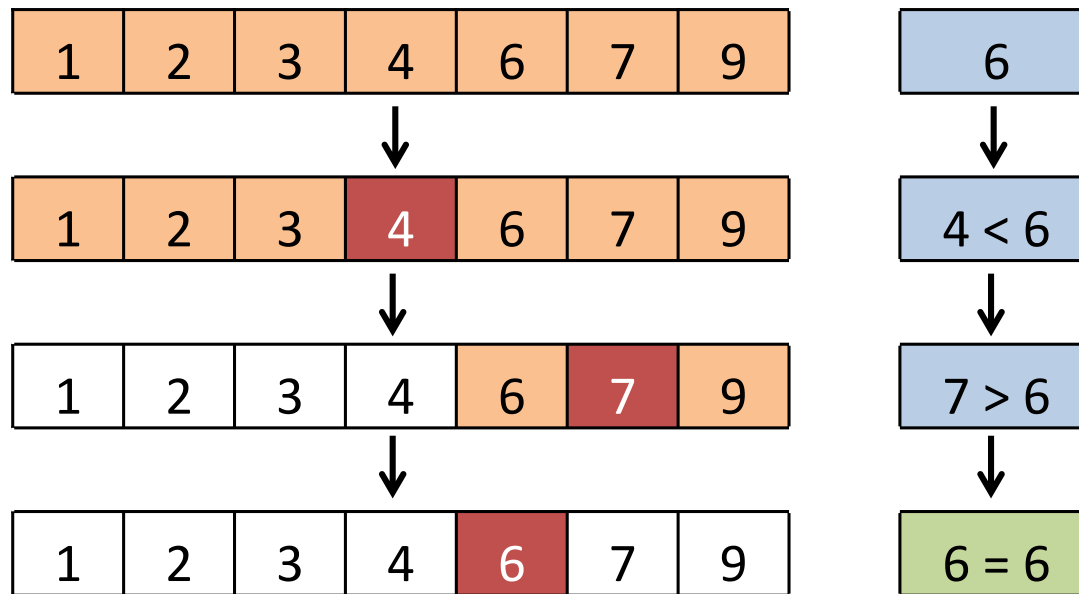
# Busca sequencial em arquivo ordenado

```
01: algoritmo Busca
02:     ler a chave de busca
03:     ir para o primeiro registro do arquivo
04:     enquanto não atingir o fim do arquivo
05:         ler o próximo registro
06:         se chave de busca = chave do registro
07:             então retornar registro e terminar
08:         fim-se
09:         se chave de busca > chave do registro
10:             então terminar
11:         fim-se
12:     fim-enquanto
13: fim-algoritmo
```

# Busca binária

```
01: algoritmo BuscaBinária
02:   ler a chave de busca
03:   POS_INICIAL ← posição do primeiro registro
04:   POS_FINAL ← posição do último registro
05:   enquanto POS_INICIAL ≤ POS_FINAL
06:     POS_MEIO ← (POS_INICIAL+POS_FINAL)/2
07:     ir para a posição POS_MEIO
08:     ler o registro
09:     se chave de busca = chave do registro
10:       então retornar registro e terminar
11:       senão se chave de busca < chave do registro
12:         então POS_FINAL <- POS_MEIO - 1
13:         senão POS_INICIAL <- POS_MEIO + 1
14:       fim-se
15:     fim-se
16:   fim-enquanto
17:   retornar não encontrado e terminar
18: fim-algoritmo
```

# Busca binária



Número máximo de comparações:  $\log_2 n$

**Inclusão**

# Inclusão

- Arquivo sem ordenação ou ordenado por código (ordem de cadastro)

```
01: algoritmo Inclusão
02:   ler o último código usado no cabeçalho do arquivo
03:   novo código ← último código + 1
04:   ir para o fim do arquivo
05:   escrever o novo registro com o novo código
06:   ir para o início do arquivo (cabeçalho)
07:   escrever o novo código
08: fim-algoritmo
```

# Inclusão

- Arquivo ordenado

```
01: algoritmo InclusãoOrdenada
02:   ler o último código usado no cabeçalho do arquivo
03:   novo código  $\leftarrow$  último código + 1
04:   ir para o primeiro registro do arquivo
05:   enquanto não atingir o fim do arquivo
06:     POSIÇÃO  $\leftarrow$  posição no arquivo
08:     ler o próximo registro
09:     se chave de busca < chave do registro
10:       então sair (do laço enquanto)
11:     fim-se
12:   fim-enquanto
```

```
13:      ir para a posição POSIÇÃO
14:      criar arquivo temporário (se possível, em 2ª mídia)
15:      enquanto não atingir o fim do arquivo principal
16:          ler um bloco de registros no arquivo principal
17:          escrever o bloco de registros no arquivo temporário
18:      fim-enquanto
19:      ir para a posição POSIÇÃO
20:      escrever o novo registro com o novo código
21:      ir para o início do arquivo temporário
22:      enquanto não atingir o fim do arquivo temporário
23:          ler um bloco de registros no arquivo temporário
24:          escrever o bloco de registros no arquivo principal
25:      fim-enquanto
26:      ir para o início do arquivo (cabeçalho)
27:      escrever o novo código
28: fim-algoritmo
```

# Inclusão

- Arquivo com chave de ordenação
  - A alternativa de "puxar" os registros para baixo, para gerar um novo espaço, não é recomendável, porque envolve muitas movimentações do HD.
  - Mas é possível trabalhar com grandes blocos de registros, usando memória ou arquivo temporário.



# Inclusão

- **Acesso concorrente**

O arquivo não pode ficar travado e, normalmente, trabalha-se com uma base replicada.

# Área de extensão

- Otimiza o processo de inclusão, quando o arquivo é muito grande (movimentação lenta) ou as reconstruções são restritas (múltiplos acessos simultâneos).
- Combina o acesso sequencial com listas encadeadas.
- Um cabeçalho pode indicar o início da área de extensão.
- Periodicamente (ou pelo tamanho da área de extensão), o arquivo também precisará ser reorganizado.

# Área de extensão

Posição	Código	Nome	Email	Data Nascimento	Próximo
0	2	Ana Maria Fonseca	ana@hotmail.com	17/04/1975	-1
1	1	Carlos Chagas	chagas@gmail.com	13/12/1977	-1
2	4	Joana Torres	joana@gmail.com	31/07/1982	101
3	3	Sílvia Costa	silvio@gmail.com	28/01/1969	-1
...	...	...	...	...	...
100	15	Rafael Oliveira	rafa@hotmail.com	06/11/1998	-1
101	8	Pedro Cardoso	pedrao@site.com.br	27/09/1987	100

# **Alteração e exclusão**

# Alteração

- Se a alteração implicar em mudança na posição do registro
  - Excluir e, em seguida, incluir o registro alterado
- Se não implicar na mudança de posição
  - Se o registro for de tamanho fixo, alterar no local
  - Se o registro for de tamanho variável, repetir procedimento da inclusão

# Exclusão

- Lápide (marca de exclusão)
  - Campo (1 byte) que indica se o registro foi excluído ou permanece válido

# Exclusão

LÁPIDE; CÓDIGO; NOME; IDADE

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
			3		' '			1			13	'C'	'A'	'R'	'L'	'O'	'S'	' '	'C'	'H'

Byte	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
	'A'	'G'	'A'	'S'		25	' '	2					9		'A'	'N'	'A'	' '	'M'	'A'	'R'

Byte	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	'I'	'A'		31	'*'			3			13	'S'		'Í'	'L'	'V'	'I'	'O'	' '	'C'

Byte	60	61	62	63	64	65
	'O'	'S'	'T'	'A'		27

# Exclusão

- Reaproveitamento de espaços de registros excluídos, antes da reorganização:
  - Registro maior – não é possível fazer o reaproveitamento do espaço
  - Registro de mesmo tamanho (do espaço) – nada a fazer
  - Registro menor – preencher o espaço restante com lixo (mas deve haver a indicação do tamanho desse espaço em algum lugar; ex.: *campo lixo*)



# Exclusão

- O arquivo deverá ser reorganizado para que os registros sejam realmente excluídos.
- A reorganização pode ser periódica ou baseada no número ou percentual de registros excluídos.
- A reorganização é feita com auxílio de um arquivo temporário. Apenas os registros válidos são copiados.

# Listagem

- Processamento sequencial de vários registros, eventualmente baseado em algum critério de seleção
- Se o processamento for feito na mesma ordem do armazenamento, basta localizar o primeiro registro do bloco e processar todos enquanto o critério de seleção for satisfeito
- Se o processamento for feito em ordem diferente do armazenamento, então todos os registros, sequencialmente, devem ser testados.