Engenharia de Computação

# Estrutura de Dados 2

## Aula 10 – Busca e Ordenação

**Prof. Muriel de Souza Godoi**
muriel@utfpr.edu.br

# Exemplos de busca

- Registros de tamanho fixo:

```
M A R I A | R U A b 1 | S A O b C A R L O S | b b b b b b b
J O A O | R U A b A | R I O b C L A R O | b b b b b b b b b
P E D R O | R U A b X V | S A O b C A R L O S | b b b b b b
A N T O N I A | R U A b X V b D E b M A I O | I B A T E | b
A N A | R U A b A U G U S T O b P A I V A | I B A T E | b b
```

1. Recupere os dados relativos ao Joao
2. Recupere os dados relativos ao Pedro

# Exemplos de busca

- Registro de tamanho variável:

```
M A R I A | R U A b 1 | S A O b C A R L O S | J O A O | R U
A b A | R I O b C L A R O | P E D R O | R U A b X V | S A O
b C A R L O S | A N T O N I A | R U A b X V b D E b M A I O
| I B A T E | A N A | R U A b A U G U S T O b P A I V A | I
B A T E |
```

1. Recupere os dados relativos ao Joao
2. Recupere os dados relativos ao Pedro

# Ordenação

- Facilita a busca
- Pode ajudar a diminuir o número de acessos a disco
- Busca sequencial:
  - recupera cada registro do arquivo, verificando se os valores dos atributos satisfazem à condição de seleção
- Busca binária:
  - recupera registros quando a condição de seleção envolve uma comparação de igualdade no atributo que determina a ordenação do arquivo

# Custo : Comparações

$$\text{Custo}_{\text{busca\_sequencial}} = n$$

- n: número de registros que são comparados
- todos os registros são varridos (pior caso)
- complexidade: **O(n)**

$$\text{Custo}_{\text{busca\_binária}} = \log_2 (n) + 1$$

- n: número de registros que são comparados
- complexidade: **O(log n)**

# Custo : Acessos a disco

$$\text{Custo}_{\text{busca\_sequencial}} = b$$

- b: número de blocos que contêm os registros
- todos os blocos são varridos

$$\text{Custo}_{\text{busca\_binária}} = \log_2(b) + \lceil s/bfr \rceil - 1$$

- $\log_2(b)$: custo para localizar o primeiro registro
- $\lceil s/bfr \rceil$: blocos ocupados pelos registros que satisfazem à condição de seleção
  - s = tamanho total dos dados / bfr = tamanho do buffer
- 1: custo para recuperar o primeiro registro

# Como ordenar o arquivo

- Arquivo completo cabe em RAM
- Estratégia
  - leitura de todos os registros armazenados em disco para a RAM
  - ordenação dos registros em RAM
    - escolha do campo base para ordenação
    - uso de um método de ordenação
  - escrita de todos os registros armazenados em RAM para o disco

# Arquivo ordenado

- Registros de tamanho fixo:

```
A N A | R U A b A U G U S T O b P A I V A | I B A T E | b b
A N T O N I A | R U A b X V b D E b M A I O | I B A T E | b
J O A O | R U A b A | R I O b C L A R O | b b b b b b b b b
M A R I A | R U A b 1 | S A O b C A R L O S | b b b b b b b
P E D R O | R U A b X V | S A O b C A R L O S | b b b b b b
```

ordenação baseada em um determinado campo, usando suas chaves

# Chave (KEY)

- Está associada a um registro e permite a sua recuperação
- Chave primária
  - identifica univocamente um registro
  - não tem repetição
- Chave secundária
  - não identifica univocamente um registro
  - tem repetição

# Forma Canônica da Chave

- Uma única representação para uma determinada chave
  - Exemplo: "Ana", "ANA", ou "ana" devem indicar o mesmo registro

- Exemplo de regra para a Forma canônica
  - todos os caracteres em letras maiúsculas → ANA

# Como Ordenar o Arquivo

- Arquivo completo não cabe em RAM
- Estratégia: ordenação por chave
  - conhecida como keysorting
  - armazena e ordena em RAM somente
    - chaves para ordenação
    - RRNs ou byte offsets dos registros

# Ordenação por Chave ( Keysorting )

- **1.** Leitura completa do arquivo de dados, trazendo para a RAM a chave e o RRN (ou byte offset ) dos registros

# Ordenação por Chave ( Keysorting )

- **2.** Ordenação do vetor em RAM
  - uso de um método de ordenação

| *chave* | *RRN* |
|---|---|
| MARIA | 0 |
| JOAO | 1 |
| PEDRO | 2 |
| ANTONIA | 3 |
| ANA | 4 |

*vetor em RAM*

| *chave* | *RRN* |
|---|---|
| ANA | 4 |
| ANTONIA | 3 |
| JOAO | 1 |
| MARIA | 0 |
| PEDRO | 2 |

*vetor ordenado em RAM*

# Ordenação por Chave ( Keysorting )

- **3.** Para cada registro do vetor em RAM
  - obtém o RRN;
  - identifica o byte offset do registro em disco
    `( byte offset = RRN * tamRegistro)`
  - lê o registro do arquivo em disco;
    - arquivo de entrada desordenado
  - escreve o registro de forma ordenada em outro arquivo
    - arquivo de saída ordenado

# Ordenação por Chave ( Keysorting )

- 5. Apaga o arquivo anterior
  - Renomeia o arquivo gerado com o come do anterior
- Arquivo ordenado em disco

```
A N A | R U A b A U G U S T O b P A I V A | I B A T E | b b
A N T O N I A | R U A b X V b D E b M A I O | I B A T E | b
J O A O | R U A b A | R I O b C L A R O | b b b b b b b b b
M A R I A | R U A b 1 | S A O b C A R L O S | b b b b b b b
P E D R O | R U A b X V | S A O b C A R L O S | b b b b b b
```

# Ordenação

```
A N A | R U A b A U G U S T O b P A I V A | I B A T E | b b
A N T O N I A | R U A b X V b D E b M A I O | I B A T E | b
J O A O | R U A b A | R I O b C L A R O | b b b b b b b b b
M A R I A | R U A b 1 | S A O b C A R L O S | b b b b b b b
P E D R O | R U A b X V | S A O b C A R L O S | b b b b b b
```

- Perguntas
  - e se a busca for feita por outro campo que não seja o campo ordenado?
  - o que acontece quando BEATRIZ é inserida?

# Pensando em Índices

*Por que realizar a tarefa custosa de escrever em disco a versão ordenada do arquivo?*

- Solução melhor ....
  - grava-se a ordenação da chave em um novo arquivo (**arquivo de índice**)
  - realiza-se busca binária no arquivo de índice, e recupera-se o RRN ou byte offset
  - realiza-se acesso direto no arquivo original (arquivo de dados)