

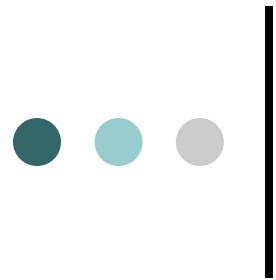
# Capítulo 6

Indexação



# Programa

- Introdução
- Operações básicas sobre arquivos
- Armazenagem secundária
- Conceitos básicos sobre estrutura de arquivo
- Organizando arquivos para desempenho
- **Indexação**
- Processamento co-seqüencial e ordenação
- B-Tree e outras organizações em árvores
- B+Tree e acesso seqüencial indexado
- Hashing
- Hashing estendido



# Objetivos

- Introduzir os conceitos básicos de indexação.
- Apresentar a utilização de índices simples para promover rápido acesso a registros.
- Investigar as aplicações do uso de indexação para manutenção de arquivos.
- Investigar o acesso a registros usando mais de um índice.



# Visão Geral do Capítulo

1. O que é um índice?
2. Índices simples com inserção temporal.
3. Operações básicas de indexação.
4. Índices muito grandes na memória.
5. Acesso por múltiplas chaves.
6. Recuperação por combinação de chaves secundárias.
7. Listas invertidas.
8. Índices seletivos.

# ● ● ● | O Que é Um Índice?

- Índices são chaves e campos de referência associados a um (ou mais) registro(s).
- Um índice é uma maneira usada para encontrar informação.
  - É um mecanismo de acesso a arquivos.
- Índice simples (ou lineares)
  - Arranjos (arrays) simples contendo:
    - chave
    - campo de referência.

# ● ● ● | O Que é Um Índice?

- **Índices simples** utilizam arrays simples.
- Um índice nos permite impor uma ordem a um arquivo sem rearranjá-lo.
- Índices também nos proporcionam múltiplos caminhos de acesso a um arquivo – **índices múltiplos** (um catálogo de uma biblioteca fornecendo buscas por autor, título, tema, etc.)
- Um índice pode permitir acesso via chave a registros de tamanho variável.

# ● ● ● | O Que é Um Índice?

- Exemplo: cadastro de musicas
  - Registros de tamanho variável
  - Chave primária = Label da gravadora + código (LABEL ID)

17	LON   2312   Symphony N S
62	RCA   2626   Quartet in C sharp   ...
117	WAR   23699   Adagio   ...
152	ANG   3795   Violin Concerto   ...

Address of  
Record

- Podemos ordenar o arquivo e fazer uma pesquisa binária?
  - Não, pois não é possível pular para o registro do meio no arquivo, uma vez que arquivos com **registros de tamanho variável não permitem acesso direto pela PRR.**

# ● ● ● | O Que é Um Índice?

## ○ Índices Simples:

- Registros de tamanho variável
- Chave primária = Label da gravadora + código (LABEL ID)
- Campo de referência = o endereço do 1o byte do registro

Index .

key	Reference field
ANG3795	152
LON2312	17
RCA2626	62
WAR23699	117

17	LON   2312   Symphony N.S   ...
62	RCA   2626   Quartet in C sharp   ...
117	WAR   23699   Adagio   ...
152	ANG   3795   Violin Concerto   ...

Address of Record

- O “Index” é ordenado em memória, e os registros aparecem no arquivo na ordem que foram incluídos.

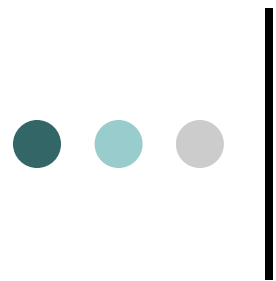


# ● ● ● | O Que é Um Índice?

- No exemplo anterior, como procurar por uma determinada LABEL ID?
  - Pesquisa Binária (na memória principal) no arquivo de índices (Index): encontra a “LABEL ID” desejada, que vai nos fornecer o campo de referência.
  - Busca diretamente no arquivo principal, o registro na posição indicada pelo campo de referência.

# ● ● ● | O Que é Um Índice?

- PROCEDURE busca\_registro(CHAVE)
  - Encontra a posição da CHAVE no arquivo de índices (provavelmente via pesquisa binária);
  - Calcula o BYTE\_OFFSET do registro correspondente no arquivo de dados;
  - Utiliza o SEEK() e o BYTE\_OFFSET para mover para o registro;
  - Lê o registro do arquivo de dados.



# O Que é Um Índice?

- **INDEXAÇÃO:**

- É uma alternativa à ordenação do arquivo:

- quando se deseja organizar um arquivo que deve ser pesquisado por chaves.
  - Usa-se um índice por tipo de chave de acesso.
  - Os índices têm registros de tamanho fixo.

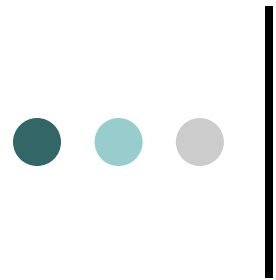
- **Permite:**

- pesquisa binária no arquivo índice.
  - Com arquivo de dados tendo chave e registro de tamanho fixo ou variável (nesse caso usa-se PRR em lugar de NRR).
  - Acesso direto no arquivo de dados.
- leitura sequencial do arquivo índice
  - buscando os registros no arquivo de dados por ordem de chave.

# ● ● ● | O Que é Um Índice?

- A estrutura do arquivo de índices é bem simples:
  - É um arquivo com registros de tamanho fixo.
  - Cada registro tem 2 campos de tamanho fixo:
    - Um campo chave
    - Um campo de referência, contendo o byte-offset.
  - Existe um registro no arquivo de índices para cada registro no arquivo de dados.
- No arquivo de dados, os registros permanecem na ordem em que foram inseridos (= inserção temporal)

O arquivo de índices é consideravelmente mais fácil de trabalhar do que o arquivo de dados, pois seus registros tem tamanho fixo, e ele é muito menor do que o arquivo de dados.



# O Que é Um Índice?

- Questões que devem ser analisadas:
  - Como fazer um “Index” persistente? (ou seja, como armazenar os índices num arquivo quando não estão na memória principal?)
  - Como garantir que o “Index” seja preciso? (Principalmente quando há muitas inserções, remoções e atualizações).

# ● ● ● | Índices simples com inserção temporal

- Os registros são inseridos no arquivo de dados na sequência em que são gerados:
  - registros são gravados na sequência temporal.
- Utilizar dois arquivos:
  - arquivo de índices:
    - registro: <chave> <referência>
      - tamanho fixo.
        - limita o tamanho da chave.
    - arquivo de dados.

O uso de um campo fixo pequeno para a chave pode causar problemas se a chave for truncada.

# ● ● ● | Índices simples com inserção temporal

## ○ Índice

<chave> <referência>

<chave> ::= chave primária (secundária).

<referência> ::= PRR (chave primária).

## ○ Inserção no arquivo de dados (ou básico)

- A cada inclusão de registro no arquivo básico, inclua <chave> e <referência> no arquivo índice (ordenadamente)

## ○ Eliminação

- A cada retirada de registro no arquivo básico, elimine a entrada correspondente no arquivo índice.

## ○ Recuperação

- Procedimento *busca-registro* (já visto no slide anterior).

# ● ● ● | Índices simples com inserção temporal

## ○ Vantagens:

- adição rápida se o arquivo de índices pode ser mantido na memória.
- a pesquisa binária pode ser feita na memória.
- recuperação:
  - requer uma única leitura para recuperar o registro no arquivo básico:
    - dada a chave, use pesquisa binária para recuperar a PRR.
    - posicione no registro com um único *seek* e o leia.

Por enquanto vamos assumir que o arquivo de índices é lido da memória secundária e armazenado na memória principal num array chamado INDEX[ ]. Mais tarde vamos considerar o caso em que o arquivo de índices é muito grande para ser armazenado na memória principal.





# Operações Básicas de Indexação

- a) Criar arquivo de índice e de dados.
- b) Carregar o arquivo de índice para a memória.
- c) Regravar arquivo de índice depois de usá-lo.
- d) Incluir registros:
  - 1. no arquivo de dados e no de índice.
- e) Excluir registros:
  - 1. no arquivo de dados e no de índice.
- f) Atualizar registros:
  - 1. Com mudança de chave primária.
  - 2. Sem mudança de chave primária.



# Operações Básicas de Indexação

- a) Criar arquivo de índice e de dados:
- os arquivos de dados e de índices são criados inicialmente como arquivos vazios.
  - estes arquivos serão posteriormente carregados com seus dados.



# Operações Básicas de Indexação

- b) Carregar arquivo de índice para a memória:
- assuma que o arquivo índice caiba todo na memória em INDEX[].
  - leia arquivo índice:
    - leia o cabeçalho;
    - verifique se a data do cabeçalho do índice é a do arquivo de dados.
    - se o arquivo índice não for válido, gere um arquivo válido.
    - leia os registros do arquivo índice e coloque-os no vetor INDEX[].
      - a leitura é rápida:
        - os registros do arquivo índice são pequenos.
        - grande quantidade deles podem estar num mesmo bloco.
        - a leitura é sequencial, há pouco movimento de braço.

# Operações Básicas de Indexação

- c) Regravar arquivo de índice depois de usá-lo.
  - Operação feita quando o INDEX[ ] é alterado.

```
procedure regrava-índice() {  
    se existe alteração no vetor INDEX[ ] então {  
        abra um novo arquivo de índices, índice_arq.  
        escreva o registro cabeçalho para índice_arq.  
        // inclua a data de atualização  
        grave o INDEX[] da memória no arquivo índice_arq.  
        feche o arquivo índice-arq.  
    }  
}
```

# ● ● ● | Operações Básicas de Indexação

- c) Regravar arquivo de índice depois de usá-lo.
- ATENÇÃO: Devido a falhas (de energia, por ex.) a atualização do arquivo de índices pode não ser realizada com sucesso.
- Como proteção contra este tipo de erro o programa deve ter pelo menos o seguinte:
  - Um mecanismo que permita o programa saber quando o arquivo de índices está desatualizado.
  - Se o programa detectar que o arquivo de índices está desatualizado, o programa deve acessar um procedimento que reconstrua o mesmo a partir do arquivo de dados.



# Operações Básicas de Indexação

- c) Regravar arquivo de índice depois de usá-lo.
- MECANISMO P/ DETECTAR DESATUALIZAÇÃO DO INDEX:
  - “setar” um *flag* assim que o arquivo de índices em memória for modificado. Este *flag* poderia ser gravado no registro cabeçalho do arquivo de índices em disco e limpo quando o arquivo for reescrito.
  - Todos os programas verificariam o status do *flag* antes de utilizar o arquivo de índices. Se o *flag* está “*setado*”, então o programa saberia que o arquivo de índices estaria desatualizado.
- RECONSTRUÇÃO DO ARQ DE ÍNDICES:
  - Deve acontecer automaticamente, antes de qualquer tentativa de acesso ao arquivo de índices.

# Operações Básicas de Indexação

## d) Incluir Registros

- Incluir registro no arquivo básico:
  - através da PRR do registro (*byte\_offset*).
  - no final do arquivo ou em algum espaço, atualizando a LED se for o caso.
- Atualizar o vetor INDEX (na memória):
  - incluir a chave primária e a PRR na ordem certa.
  - ativar o flag de atualização do INDEX[ ]
    - permite saber que o INDEX[ ] precisa ser regravado no arquivo em disco, ao término da aplicação.

Em memória!

Obs: O INDEX[] é um array mantido ordenado. A inserção de um novo registro no INDEX[] requer alguma reorganização do INDEX[]. Registros tem que ser shift, etc. A grande diferença aqui é que o INDEX[] é um array mantido (inteiramente) em memória. Toda a reorganização do INDEX[] pode ser feita sem nenhum acesso à arquivo.

# ● ● ● | Operações Básicas de Indexação

## e) Excluir Registros

- Eliminar o registro no arquivo de dados:
  - atualizar a LED para reaproveitar o espaço.
- Eliminar a entrada correspondente no INDEX[ ]
  - promover a reorganização do INDEX[ ]
    - mantendo-o ordenado pela chave.
  - ativar o *flag* sinalizando a atualização do vetor de INDEX.

Como o INDEX é mantido em um array em memória, a reorganização do INDEX[ ] não se torna uma operação custosa.



# ● ● ● | Operações Básicas de Indexação

## f) Atualizar Registros

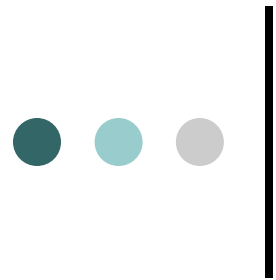
### ○ Com mudança de chave primária:

- atualizar o INDEX[ ], faz-se uma exclusão e uma inclusão.
- alterar o registro correspondente no arquivo de dados:
  - (sem maiores consequências).

### ○ Sem mudança de chave primária:

- se a alteração couber no espaço do atual registro então
  - nenhuma mudança é feita no INDEX[ ].
- caso não caiba, deve-se colocar o espaço atual na LED e incluir em uma nova posição no arquivo de dados:
  - alterar a PRR da chave correspondente no INDEX[ ].
  - ativar o flag de atualização do índice.

O usuário vê apenas como uma alteração!



## Próxima Aula...

- Índices muito grandes na memória.
- Acesso por múltiplas chaves.
- Recuperação por combinação de chaves secundárias.
- Listas invertidas.
- Índices seletivos