Armazenamento e Indexação

Capítulo 8 do livro "*Database Management Systems*", 3 ed., de R. Ramakrishnan e J. Gehrke, McGraw-Hill, 2003

Índices

MUITO importante para melhorar o tempo de processamento de uma consulta.

Considere a relação:

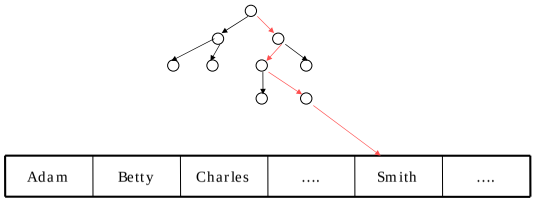
Pessoa (nome, idade, cidade)

```
SELECT *
FROM   Pessoa
WHERE  nome = "Smith"
```

O processamento sequencial do arquivo Pessoa pode levar muito tempo.

Índices

- Criação de um índice sobre o atributo nome:



Criação de Índices

Sintaxe:

```
CREATE INDEX nomeInd ON Pessoa(nome)
```


Armazenamento Externo de Dados

- Discos: recuperam páginas randômicas a um custo fixo
  - Porém, a leitura de páginas consecutivas é muito mais rápido que a leitura de páginas em ordem randômica.
  - Fitas: podem ler páginas em ordem sequencial
- Mais baratos que discos; são usados para arquivar dados.
- Organização de Arquivos: é um método para organizar registros de arquivos em um dispositivo de armazenamento externo.
  - Id de Registro (rid): é utilizado para localizar um registro
  - Índices: são estruturas de dados que permitem achar o Id de registros que possuem um determinado valor para uma chave de pesquisa.
- Arquitetura: O gerenciador de buffer executa a transferência de páginas do armazenamento externo para o buffer. Os gerenciadores de arquivo e de índice fazem chamadas ao gerenciador de buffer.

Formas de Organização de Arquivos

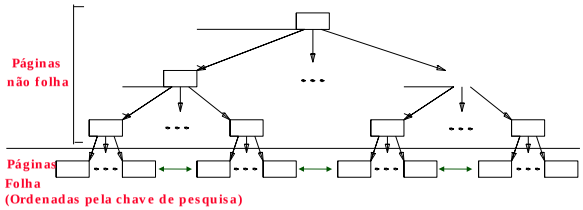
Existem diversas alternativas, que são apropriadas para algumas situações e não tão boas para outras:

- Arquivos *Heap* (ordem randômica): apropriado quando o acesso típico é a busca de todos os registros.
- Arquivos Ordenados: recomendados quando os registros devem ser recuperados em alguma ordem, ou quando somente registros em um intervalo de valores são necessários.
- Índices: estruturas de dados que organizam os registros utilizando uma árvore ou função de *hash*.
  - similar aos arquivos ordenados, os índices são recomendados para a obtenção de registros dentro de um intervalo de valores da "chave de pesquisa".
  - Atualizações são muito rápidas, comparadas a arquivos ordenados.

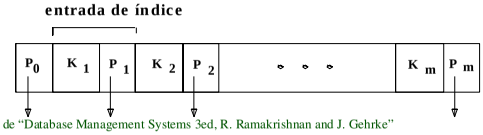
Índices

- Um *índice* sobre um arquivo agiliza a seleção de registros baseada nos *campos da chave de pesquisa* utilizados para criar o índice.
  - Qualquer atributo de uma relação pode fazer parte dos campos da chave de pesquisa do índice.
  - *Chave de pesquisa* não é o mesmo que *chave* (conjunto mínimo de atributos que identificam unicamente uma tupla da relação).
- Um índice contém uma coleção de *entradas de dados,* cada uma com um valor de chave k (representado por k*).
  - Dada uma entrada de dados k*, é possível recuperar o registro de dados com chave k com no máximo um acesso a disco.

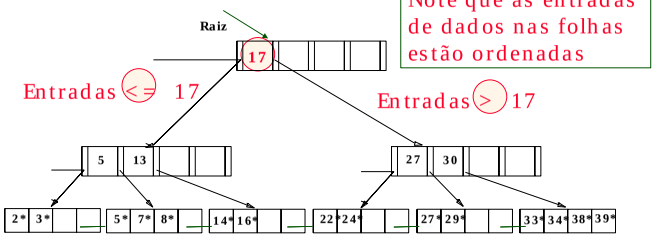
Índices com Árvore B+



- Páginas folha contêm *entradas de dados* e são encadeadas (ant, prox)
- Páginas não-folha contêm *entradas de índices;* usadas somente para direcionar a busca:



Exemplo de Árvore B+



- Encontrar: 28*, 29*, todos >15* e < 30*
- Inserção/remoção: Procura a entrada de dados na folha e a atualiza. Às vezes é necessário ajustar a página pai.
  - É possível que os ajustes se propaguem até a raiz.

Índices baseados em Hash

- Bom para seleções que utilizam o teste de igualdade.
- O índice é uma coleção de *buckets*.
  - Um Bucket = *página primária* mais zero ou mais *páginas de overflow*.
  - Os buckets contêm entradas de dados.
- *Função de Hash* **h**: **h**(*r*) = bucket no qual a entrada de dados do registro *r* deve estar contido.
  - *Não há necessidade de "entradas de índice" nesta estrutura de dados.*

Entrada de dados *k** no Índice

- Na entrada de dados k* podemos armazenar:
  - O registro de dados com chave de busca **k,** ou
  - **<k**, rid do registro de dado com chave de busca **k>**, ou
  - **<k**, lista de rids dos registros com chave de busca **k>**
- A escolha da alternativa é ortogonal à técnica de indexação utilizada para localizar as entradas de dados com uma determinada chave de busca **k**.
  - Exemplos de estruturas de indexação: árvores B+, tabelas hash

As Formas Alternativas de Entradas de Dados

- Alternativa 1:
  - Neste caso, a própria estrutura de índice corresponde a organização de arquivos utilizada para armazenar os registros de dados.
  - No máximo um índice pode utilizar a alternativa 1 para um mesmo conjunto de registros de dados. Caso contrário, haverá duplicação de registros, que tem como consequência redundância de armazenamento e pode causar inconsistências).
  - Se o tamanho do registro de dados é grande, o número de páginas contendo entradas de dados é grande. Isto implica que o tamanho das informações auxiliares no índice também pode ser grande.

de Dados (Cont.)

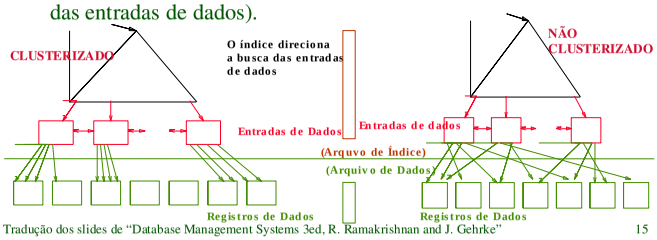
- Alternativas 2 e 3:
  - As entradas de dados em geral são bem menores que os registros de dados. Portanto, esta alternativa é melhor se os registros de dados são grandes, principalmente quando a chave de pesquisa é pequena.
  - A Alternativa 3 é mais compacta que a Alternativa 2, mas utiliza entradas de dados de tamanho variável, mesmo quando a chave de pesquisa é de tamanho fixo.

Classificação de Índices

- *Primário* vs. *secundário*: se a chave de pesquisa é uma chave primária, então o índice é primário.f
  - *índice Unique*: a chave de pesquisa contém uma chave candidata.
- *"Clusterizado"* vs. "não clusterizado": se a ordem dos registros de dados for igual ou próxima a ordem das entradas de dados do índice, então o índice é "clusterizado".
  - Alternativa 1 implica que o índice é clusterizado;
  - Um arquivo pode estar clusterizado no máximo por uma chave de pesquisa.
  - A clusterização afeta MUITO o custo de recuperação dos registros de dados através dos índices .

Índices Clusterizados

- Suponha que a Alternativa 2 seja usada para as entradas de dados e os registros de dados estão armazenados em um arquivo "Heap".
  - Para criar um índice clusterizado, ordana-se o arquivo de dados (com espaço livre nas páginas para inserções futuras).
  - Páginas de overflow podem ser necessárias para inserções (portanto, a ordem dos registros é próxima, mas não idêntica a das entradas de dados).



Analisando a Carga de Trabalho

- Para cada consulta, analisar:
  - Quais relações são utilizadas?
  - Quais os atributos retornados pela consulta?
  - Quais os atributos estão nas condições de junção e seleção? Quão seletivas são estas condições?
- Para cada atualização, analisar:
  - Quais atributos estão nas condições de seleção e junção? Quão seletivas são estas condições?
  - O tipo de atualização (INSERT/DELETE/UPDATE), e os atributos que são atualizados.

A Escolha dos Índices

- Quantos índices criar?
  - Quais as relações que devem ter índices?
  - Quais os atributos que devem formar a chave de pesquisa?
  - Devemos criar diversos índices?
- Para cada índice, que tipo de índice ele deve ser?
  - Clusterizado?
  - Hash ou árvore?

A Escolha dos Índices (Cont.)

- Considere as consultas mais importantes e os planos de consulta utilizando os índices existentes. Verifique se é possível obter um plano de consulta melhor se existirem índices adicionais. Em caso afirmativo, crie novos índices.
  - é necessário entender como um SGBD avalia consultas e cria planos de avaliação de consultas.
  - dicutiremos somentes consultas simples com apenas uma tabela.
- Antes de criar um índice, deve ser avaliado também o impacto da sua criação nas atualizações.
  - Trade-off: os índices podem tornar as consultas mais rápidas e atualizações mais lentas. Eles também requerem espaço em disco.

Guia para a Seleção de Índices

- Atributos na cláusula WHERE são candidatos a chaves de pesquisa.
  - Teste de igualdade: tabela hash. Ex: idade = 30
  - Teste de intervalo: árvore B+. Ex: idade entre 30 e 40
    - A clusterização ajuda nas consultas envolvendo teste de intervalo; também ajuda quando o teste é de igualdade se houver valores duplicados.
- Chaves de pesquisa com múltiplos atributos devem ser considerados se a cláusula WHERE contiver diversas condições.
  - A ordem dos atributos é importante em testes de intervalo.
  - Este tipo de índice pode permitir planos de pesquisa que utilizam somente o índice (sem recuperar os registros de dados).
    - Para planos de consulta que envolvem somente o índice, a clusterização não é importante.
- Procure escolher os índices que beneficiem o maior número de consultas possível. Como somente um índice pode ser clusterizado, escolha-o baseado nas consultas mais importantes que se beneficiarão com essa organização.

Exemplos de Índices Clusterizados

- Um índice com árvore B+ sobre E.idade pode ajudar.
  - Quão seletiva é a condição?
  - O índice é clusterizado?

```
SELECT E.numDep
FROM Emp E
WHERE E.idade>40
```

- Considere a consulta com GROUP BY.
  - se houver muitas tuplas com E.idade > 10, usar o índice sobre E.idade pode não ser muito bom.
  - índice clusterizado sobre *E.numDep* pode ser melhor.

```
SELECT E.numDep, COUNT (*)
FROM Emp E
WHERE E.idade>10
GROUP BY E.numDep
```

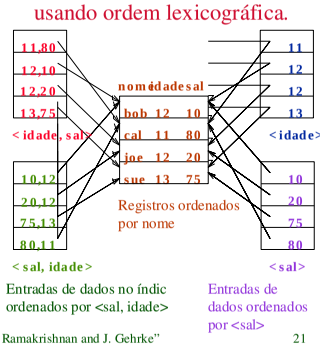
- Teste de igualdade e duplicações:
  - Clusterização em *E.hobby* é bom.

```
SELECT E.numDep
FROM Emp E
WHERE E.hobby="selos"
```


Compostas

- *Chave de Pesquisa Composta:* pesquisa em mais de um campo.
  - Teste de igualdade: Todo campo é igual a uma constante. Ex: sobre um índice sobre <sal, idade>:
    - idade=20 e sal =75
  - Teste de intervalo: Algum campo não é uma constante. Ex:
    - idade=20 and sal > 10
- As entradas de dados devem estar ordenadas pela chave de pesquisa para ajudar em testes de intervalo.
  - ordem lexicográfica

Exemplos de chaves compostas usando ordem lexicográfica.



Entradas de dados no índic ordenados por <sal, idade>

Entradas de dados ordenados por <sal>

Chave de Pesquisa Compostas

- Para obter registros da tabela Emp com *idade*=30 e *sal*=4000, um índice sobre <*idade,sal*> ou <*sal, idade*> é melhor que um índice sobre *idade* ou sobre *sal*.
- Se a condição é: 20<*idade*<30 AND 3000<*sal*<5000:
  - um índice B+ clusterizado sobre <*idade,sal*> ou <*sal,idade*> é o melhor.
- Se a condição é: *idade*=30 AND 3000<*sal*<5000:
  - um índice clusterizado sobre <*idade,sal*> é muito melhor que um índice sobre <*sal,idade*>.
- Índices compostos são maiores e precisam ser atualizados com mais frequência.

Planos de Consulta somente com Índices

- Há consultas que podem ser respondidas sem que os registros de dados de uma ou mais relações sejam recuperadas se existirem índices apropriados.

<*E.numDep*>

```
SELECT E.numDep, COUNT(*)
FROM Emp E
GROUP BY E.numDep
```

<*E.numDep,E.sal*> *Índice B+*

```
SELECT E.numDep, MIN(E.sal)
FROM Emp E
GROUP BY E.numDep
```

<*E. idade,E.sal*> ou <*E.sal, E.idade*> *Índice B+*

```
SELECT AVG(E.sal)
FROM Emp E
WHERE E.age=25 AND
 E.sal BETWEEN 3000 AND 5000
```


Planos com Índice (Cont.)

- Planos somente com índice são possíveis se a chave for <numDep,idade> ou <idade,numDep>
  - Qual é melhor?
  - E se considerarmos a segunda consulta?

```
SELECT E.numDep, COUNT (*)
FROM Emp E
WHERE E.idade=30
GROUP BY E.numDep
```

```
SELECT E.numDep, COUNT (*)
FROM Emp E
WHERE E.idade>30
GROUP BY E.numDep
```

Planos com Índice (Cont.)

- Planos que envolvem somente o índice também existem para consultas que utilizam mais de uma relação.

<E.numDep>

```
SELECT  D.gerente
FROM  Depto D, Emp E
WHERE  D.numDep=E.numDep
```

<E.numDep,E.idEmp>

```
SELECT  D.gerente, E.idEmp
FROM  Depto D, Emp E
WHERE  D.numDep=E.numDep
```


No Postgres

- CREATE [UNIQUE] INDEX nome-do-indice ON nome-da-tabela (lista-de-atributos) WHERE condição
- Exemplo:

```
CREATE INDEX indEmp
ON Dep (numDep);
```


No Postgres

- CLUSTER nome-do-índice ON nome-da-tabela
  - faz a clusterização dos registros de dados da tabela de acordo com a chave de pesquisa do índice
  - a clusterização só é feita nos registros atuais – atualizações posteriores não obedecem a clusterização
- CLUSTER nome-da-tabela
  - reordena os registros da tabela de acordo com a clusterização já definida
- CLUSTER

Resumo

- Existem diversas alternativas para organização de arquivos, cada uma apropriada para algumas situações.
- Se consultas com seleção são frequentes, é importante ordenar o arquivo ou criar um índice.
  - índices hash são bons para teste de igualdade.
  - arquivos ordenados e índices B+ são melhores para testes de intervalo; são também bons para teste de igualdade.
- Um índice consiste de uma coleção de entradas de dados que ajudam na localização de registros de dados que contém determinados valores para a chave de pesquisa.

Resumo (Cont.)

- As entradas de dados podem conter os próprios registros, conter pares <chave, rid>, ou conter pares <chave, lista-de-rids>.
  - A escolha da técnica de indexação é ortogonal ao formato das entradas de dados.
- Podem existir diversos índices para um mesmo arquivo com registros de dados, cada um com uma chave de pesquisa diferente.
- Os índices podem ser classificados como primários ou secundários, e podem ser clusterizados ou não.

Resumo (Cont.)

- Entender a carga de trabalho do SGBD para uma aplicação é essencial para melhorar o seu desempenho.
  - Quais são as consultas e atualizações mais importantes? Quais os atributos e relações envolvidos?
- Os índices devem ser escolhidos para melhorar o desempenho de consultas.
  - atualizações causam impacto na manutenção dos índices.
  - os índices devem ser escolhidos para melhorar o desempenho de diversas consultas, se possível.
  - criar índices que possibilitem planos de consultas que utilizam somente o índice.
  - a clusterização é extremamente importante; somente um índice por relação pode ser clusterizado.
  - a ordem dos atributos em chaves de pesquisa compostas pode ser importante.