Banco de dados II

01 – Introdução a Armazenamento e Indexação

Marcos Roberto Ribeiro

Departamento de Engenharia e Computação (DEC)
Curso de Engenharia de Computação
2024

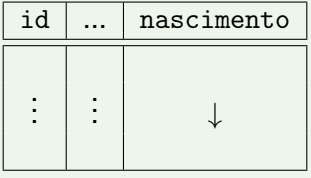
Introdução

- Basicamente, um banco de dados é um conjunto de registros
- O SGBD precisa armazenar estes registros em arquivos
- Precisamos entender como isto é feito para usar o SGBD de forma eficiente
- Dependendo de como os registros são organizados nos arquivos, podemos ter algumas operações eficientes e outras ineficientes

Exemplo – Arquivo ordenado

- Suponha que que desejamos recuperar os funcionários ordenados pela data de nascimento
- Uma alternativa é gravar os registros em arquivo ordenados pela data de nascimento

Arquivo ordenado



Exemplo – Arquivo ordenado – Prós e contras

Prós

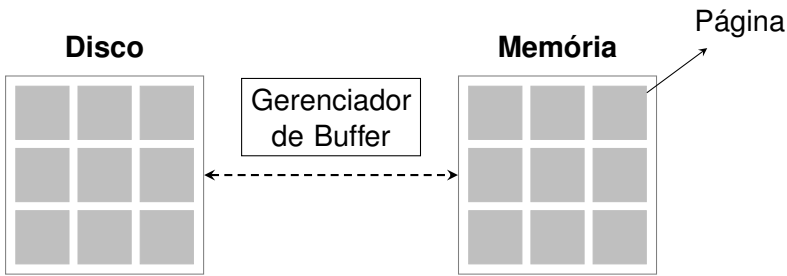
- Bom para recuperar registros pela data de nascimento

Contras

- Difícil de manter (atualizar) o arquivo
- Se houver outro tipo de consulta como: “Obter os funcionários com salários maiores do que 5.000,00”, é preciso ler todo o arquivo

- O uso da indexação auxilia na recuperação de dados sem a necessidade de mudar a estrutura de armazenamento dos registros

Armazenamento de dados



- Cada registro possui um **rid** para identificar sua página
- Leitura e gravação de dados precisa ocorrer em memória
- É necessário mover dados entre disco e memória

Tipos de armazenamento de dados

Disco (armazenamento secundário)

- Maior custo para ler ou gravar
- Maior capacidade
- Leitura sequencial de páginas mais eficiente do que a leitura aleatória (HDs tradicionais)

Memória (armazenamento primário)

- Menor custo de acesso
- Menor capacidade

Fitas e mídias óticas (armazenamento terciário)

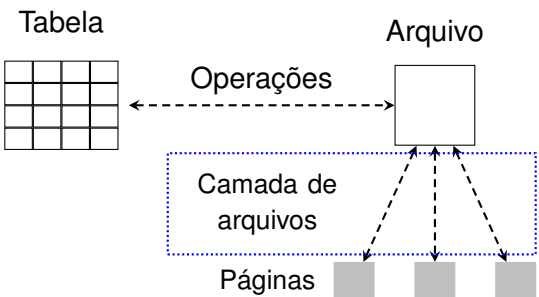
- Ideais para backup

Comparativo entre tipos de armazenamentos

Tipo de Mídia	Capacidade	Tempo de Acesso	Largura de Banda
RAM	4 GB - 1 TB	30 ns	35 GB/s
SSD	64 GB - 1 TB	50 μs	750 MB/s
HD	400 GB - 8 TB	10 ms	200 MB/s
Fitas	2 TB - 8 TB	10 s - 80 s	40-250 MB/s
Jukebox	25 TB - 2 EB	10 s - 80 s	250 MB/s - 1,2 PB/s

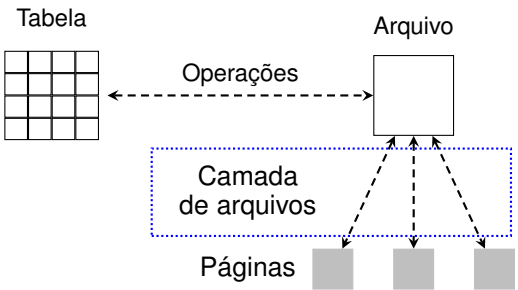
- Fonte: (ELMASRI; NAVATHE, 2018)
- Exemplo de jukebox: https://www.youtube.com/watch?v=Raz1uJM3Dpw

Organizações de arquivos e indexações



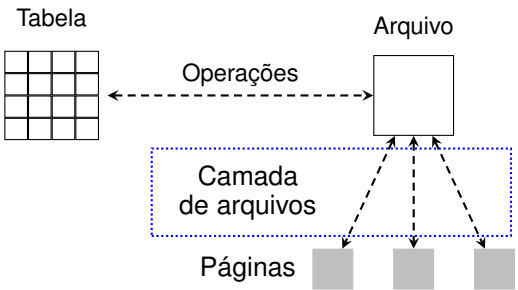
- Basicamente, uma tabela é armazenada em um arquivo pelo SGBD
- Portanto, as operações sobre a tabela precisam ser replicadas para o arquivo
- O SGBD quebra o arquivo em páginas para facilitar o gerenciamento de memória
- Por outro lado, o SGBD precisa de uma camada de arquivos que implementa as operações sobre as páginas de forma transparente

Operações sobre tabelas e arquivos



- Quais operações podem ser feitas sobre tabelas ou arquivos?
  - Inserção
  - Deleção
  - Modificação
  - Varredura

Operações sobre tabelas e arquivos



- Quais operações podem ser feitas sobre tabelas ou arquivos?
  - Inserção
  - Deleção
  - Modificação
  - Varredura

Índices

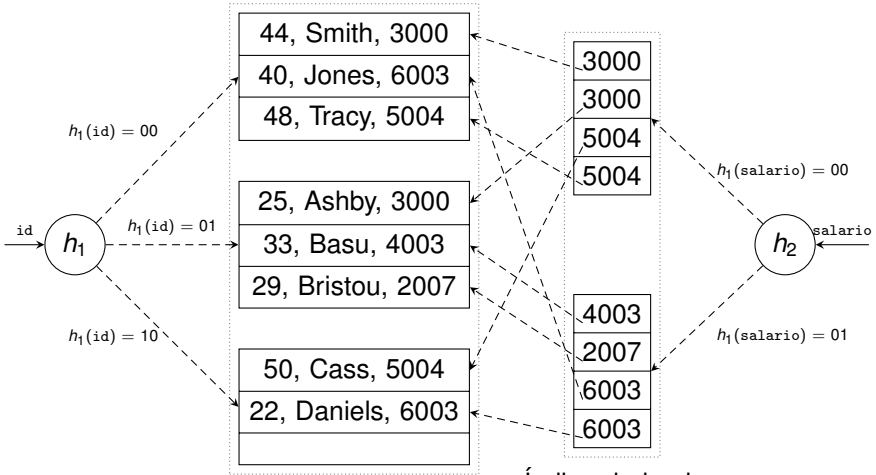
- Índices são estruturas de dados que otimizam a busca com uma **chave de pesquisa**
- As chaves de pesquisa podem ser campos ou conjuntos de campos
- As entradas de dados dos índices podem ser de três tipos:
  - O próprio registro da tabela: para **índices agrupados**[1]
  - (chave de pesquisa *k*, **rid**): para índices de valores únicos
  - (chave de pesquisa *k*, lista de **rid**): para índices com repetição de valores
- Tipos de chaves:

  Chave primária: Índice primário (não possui duplicatas)

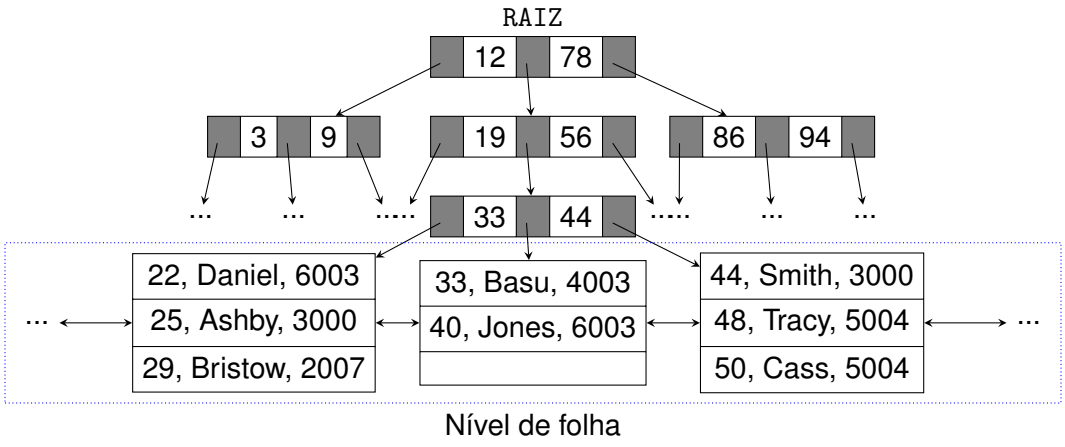
  Demais chaves: Índices secundário (pode conter duplicatas)

[1] Índice integrado com arquivo

Indexação baseada em hash



Indexação baseada em árvore



Comparação das organizações de arquivos

- Esta comparação considera um conjunto de registros de funcionários com chave de pesquisa (`nascimento`, `salario`)
- Todas as operações são especificadas sobre estes campos (`nascimento`, `salario`)
- As organizações de arquivos consideradas são:
  - Arquivo heap (ordem aleatória)
  - Arquivo ordenado por (`nascimento`, `salario`)
  - Arquivo com índice agrupado de árvore B+ sobre (`nascimento`, `salario`)
  - Arquivo com índice não agrupado de árvore B+ sobre (`nascimento`, `salario`)
  - Arquivo com índice não agrupado de hash sobre (`nascimento`, `salario`)

Comparação das organizações de arquivos

Operações consideradas

- A Varredura
- B Pesquisa (igualdade)
- C Pesquisa (intervalo)
- D Inserção de um registro
- E Exclusão de um registro

Modelo de custo

- B: Número de páginas
- R: Número de registos por página
- F: *Fan-out* (número de filhos de um nó em árvores B+)
- D: Tempo de E/S de uma página (operação mais cara)
- C: Tempo para processar um registro
- H: Tempo para computar uma função hash

Custo de arquivo heap

A Varredura: $B(D + RC) = BD$

B Pesquisa (igualdade): $B(D + RC) = BD$ (o registro pode não existir)

C Pesquisa (intervalo): $B(D + RC) = BD$

D Inserção: $2D + C$ (insere sempre no fim do arquivo)

E Exclusão: $\underbrace{B(D + RC)}_{\text{pesquisa do registro}} + \underbrace{C + D}_{\text{grava na página do registro}} = BD + D$ (considerando um registro)

Custo de arquivo ordenado

A Varredura: $B(D + RC) = BD$

B Pesquisa (igualdade): $\underbrace{D \log B}_{\substack{\text{busca}\\ \text{pela}\\ \text{página}}} + \underbrace{C \log R}_{\substack{\text{busca}\\ \text{pelo}\\ \text{registro}}} = D \log B$

C Pesquisa (intervalo): $D \log B + C \log R + \underbrace{\quad ? \quad}_{\substack{\text{páginas}\\ \text{adicionais}}} = D \log B + ?$

D Inserção: $\underbrace{D \log B + C \log R}_{\text{pesquisa}} + \underbrace{B(D + CR)}_{\substack{\text{realocação dos}\\ \text{registros}\\ \text{seguintes}}} = BD + D$

E Exclusão: mesmo caso da inserção

Custo de arquivo com índice árvore B+ agrupado

- Normalmente, as árvores B+ possuem páginas com 67% de ocupação, com isto o número de páginas é $1.5B$

A Varredura: $1.5B(D + RC) = 1.5BD$

B Pesquisa (igualdade): $\underbrace{D \log_F 1.5B}_{\text{busca pela folha}} + \underbrace{C \log R}_{\text{busca binária na folha}} = D \log_F 1.5B$

C Pesquisa (intervalo): $D \log_F 1.5B + C \log R + \underbrace{?}_{\text{páginas adicionais}} = D \log_F 1.5B + ?$

D Inserção: $D \log_F 1.5B + C \log R + D = D \log_F 1.5B + D$ (em geral, o registro cabe na folha)

E Exclusão: mesmo caso da inserção

Custo de arquivo com índice árvore B+ não agrupado I

- Consideramos páginas com 67% de ocupação e entradas de índices dom 10% do tamanho dos registros, temos:
  - $0.1 \times 1.5B = 0.15B$ páginas
  - $10 \times 0.67R = 6.7R$ registros / página

A Varredura: $\underbrace{0.15B(D + 6.7R)}_{\text{leitura de todas as entradas do índice}} + \underbrace{R(D + C)}_{\text{processamento dos registros}} = 0.15BRD^2$

- Melhor não usar o índice $B(D + RC)$

B Pesquisa (igualdade): $\underbrace{D \log_F 0.15B}_{\text{busca pela folha}} + \underbrace{C \log 6.7R}_{\text{busca na folha}} + \underbrace{D}_{\text{leitura do arquivo}} = D \log_F 0.15B + D$

Custo de arquivo com índice árvore B+ não agrupado II

C Pesquisa (intervalo): $D \log_F 0.15B + C \log 6.7R + D \times \underbrace{?}_{\text{número de registros a serem retornados}} = D \log_F 0.15B + D \times ?$

D Inserção: $\underbrace{2D + C}_{\text{gravação no arquivo}} + \underbrace{D \log_F 0.15B + C \log 6.7R}_{\text{busca pela folha no índice}} + \underbrace{D}_{\text{grava página no índice}} = 2D + D \log_F 0.15B + D$

E Exclusão: mesmo caso da inserção

Custo de arquivo com índice Hash não agrupado I

- Consideramos páginas com 67% de ocupação, hash sem overflow, páginas com 80% de ocupação e uma página adicional / bucket, temos:
  - $01.25 \times 0.1B = 0.125B$ páginas
  - $10 \times 0.8R = 8R$ registros / página

A Varredura: $B(D + RC)$ (mesmo caso do índice não agrupado de árvore B+)

B Pesquisa (igualdade): $\underbrace{H}_{\substack{\text{identificação} \\ \text{da página}}} + \underbrace{D}_{\substack{\text{leitura da} \\ \text{página}}} + \underbrace{8RC}_{\substack{\text{varredura da} \\ \text{página}}} + \underbrace{D}_{\substack{\text{leitura do} \\ \text{arquivo}}} = 2D$

C Pesquisa (intervalo): $B(D + RC)$ (o índice hash não suporta pesquisa por intervalo)

D Inserção: $\underbrace{2D + C}_{\substack{\text{gravação no} \\ \text{arquivo}}} + \underbrace{H}_{\substack{\text{identificação} \\ \text{da página no} \\ \text{índice}}} + \underbrace{2D + C}_{\substack{\text{leitura e} \\ \text{alteração da} \\ \text{página}}} = 4D$

Custo de arquivo com índice Hash não agrupado II

- E Exclusão: $\underbrace{H + 2D + 8RC}_{\text{pesquisa pelo registro}} + \underbrace{2D}_{\text{leitura e alteração da página}} = 4D$

Comparação

Tipo de Arquivo	Varredura	Pesquisa (Igualdade)	Pesquisa (Intervalo)	Inserção	Exclusão
Heap	BD	BD	BD	$2D$	Pesquisa $+D$
Ordenado	BD	$D \log B$	$D \log B$ + # páginas	Pesquisa $+BD$	Pesquisa $+BD$
Agrupado	$1.5BD$	$D \log 1.5B$	$D \log 1.5B$ + # páginas	Pesquisa $+D$	Pesquisa $+D$
Árvore B+ não agrupado	BD	$D \log 0.15B$	$D \log 0.15B$+ $D \times$ # registros	Pesquisa $+3D$	Pesquisa $+2D$
Hash não agrupado	BD	$2D$	BD	$4D$	Pesquisa $+2D$

Vantagens e desvantagens

- Arquivo heap
  - (+) Inserção rápida (sempre no final)
  - (-) Pesquisa e exclusão lentos
- Arquivo ordenado
  - (+) Pesquisa rápida
  - (-) Inserção e exclusão lentos
- Arquivo com índice agrupado de árvore B+
  - (+) Inserção e exclusão eficientes, pesquisa rápida
  - (-) Pequeno overhead
- Arquivo com índice não agrupado de árvore B+
  - (+) Pesquisa, inserção e exclusão eficientes
  - (-) Pesquisa por intervalo pode se tornar lenta
- Arquivo com índice não agrupado de hash
  - (+) Eficiente em pesquisa e atualização
  - (-) Não suporta pesquisa por intervalo

Índices e sintonização de desempenho

- Índices podem ser criados para tornar consultas mais eficientes
- Porém, certas alterações nos dados podem causar atualizações de vários índices
- Isso pode levar a problemas de desempenho no sistema
- O ideal é haver um equilíbrio, ou seja, os índices são ferramentas poderosas que devem ser usados com cautela

Avaliações somente de índice

- Quando todos os campos buscados em uma consulta fazem parte da chave do índice, tal consulta pode ser avaliada usando apenas o índice
- Não é necessário buscar os dados no arquivo
- Normalmente, o índice é menor do que o arquivo, isso faz com que a consulta seja avaliada mais rapidamente

Chaves de pesquisas compostas

- As chaves de pesquisas compostas possuem mais de um atributo
- Tais chaves podem suportar faixas maiores de consultas
- Além disto, elas aumentam as chances de *avaliações somente de índice*
- Por outro lado, o índice precisa ser atualizado se ocorrer qualquer alteração em um dos campos que compõem a chave
- Outra desvantagem é o espaço a mais ocupado pelo índice

Índices no PostgreSQL

- A criação de índices no PostgreSQL é com a instrução **CREATE INDEX**

```
CREATE INDEX [nome] ON tabela [ USING método]
(coluna_1, ..., coluna_n);
```

- Os principais métodos são:

  btree: Árvore B (método padrão, único que suporta unicidade)

  hash: Hash (não funciona para múltiplas colunas)

- Mais detalhes no manual do PostgreSQL (POSTGRESQL, 2023)

Exemplo prático – banco de dados I

```
1  DROP DATABASE IF EXISTS index_test;
2
3  CREATE DATABASE index_test;
4
5  \connect index_test;
6
7  CREATE TABLE contacts(
8     contact_id BIGSERIAL PRIMARY KEY,
9     contact_name VARCHAR(5) NOT NULL,
10    company_name VARCHAR(5),
11    phone VARCHAR(11),
12    email VARCHAR(100),
13    details TEXT
14 );
15
16 CREATE FUNCTION gen_string(INT)
17 RETURNS TEXT AS $$
```

Exemplo prático – banco de dados II

```
18 SELECT string_agg (substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',
19                    ceil (random() * 26)::integer, 1), '') AS letter
20 FROM   generate_series(1, $1);
21 $$ LANGUAGE SQL;
22
23 CREATE FUNCTION gen_number(INT)
24 RETURNS TEXT AS $$
25 SELECT string_agg (substr('0123456789',
26                    ceil (random() * 10)::integer, 1), '') AS digit
27 FROM   generate_series(1, $1);
28 $$ LANGUAGE SQL;
29
30 INSERT INTO contacts(contact_name, company_name, phone, email, details)
31 SELECT gen_string(5), gen_string(5), gen_number(11),
32        gen_string(30)||'@'||gen_string(30)||'.'||gen_string(30), gen_string(1000)
33 FROM GENERATE_SERIES(1, 100000);
```

Exemplo prático - consultas I

```
-- Disable notices
\set QUIET 1
-- Show queries
\set ECHO none
-- Discard queries output
\o /dev/null

-- Connect to database
\connect index_test;
-- Drop existing INDEX
DROP INDEX IF EXISTS contacts_contact_name_idx;
DROP INDEX IF EXISTS contacts_company_name_idx;
DROP INDEX IF EXISTS contacts_phone_idx;
DROP INDEX IF EXISTS contacts_email_idx;
DELETE FROM contacts WHERE contact_id > 100000;

-- Display timing
```

Exemplo prático - consultas II

```
18 \timing on
19 -- Show queries
20 \set ECHO queries
21
22 SELECT contact_id, contact_name FROM contacts LIMIT 1;
23
24 \echo ''
25 SELECT contact_id, contact_name FROM contacts ORDER BY contact_id LIMIT 1;
26
27 \echo ''
28 SELECT contact_id, contact_name FROM contacts ORDER BY contact_name LIMIT 1;
29
30 \timing off
31
32 \echo ''
33 -- Create index
34 CREATE INDEX contacts_contact_name_idx ON contacts (contact_name);
```

Exemplo prático - consultas III

```
35 CREATE INDEX contacts_company_name_idx ON contacts (company_name);
36 CREATE INDEX contacts_phone_idx ON contacts (phone);
37 CREATE INDEX contacts_email_idx ON contacts (email);
38
39 \timing on
40
41 \echo ''
42 SELECT contact_id, contact_name FROM contacts ORDER BY contact_name LIMIT 1;
43
44 \echo ''
45 INSERT INTO contacts(contact_name, company_name, phone, email, details)
46 SELECT gen_string(5), gen_string(5), gen_number(11),
   ↪  gen_string(30)||'@'||gen_string(30)||'.'||gen_string(30), gen_string(1000)
47 FROM GENERATE_SERIES(1, 10000);
48
49 \timing off
50
```

Exemplo prático - consultas IV

```
51 \echo ''
52 -- Drop indexes
53 DROP INDEX IF EXISTS contacts_contact_name_idx;
54 DROP INDEX IF EXISTS contacts_company_name_idx;
55 DROP INDEX IF EXISTS contacts_phone_idx;
56 DROP INDEX IF EXISTS contacts_email_idx;
57 \timing on
58
59 \echo ''
60 INSERT INTO contacts(contact_name, company_name, phone, email, details)
61 SELECT gen_string(5), gen_string(5), gen_number(11),
   ↪  gen_string(30)||'@'||gen_string(30)||'.'||gen_string(30), gen_string(1000)
62 FROM GENERATE_SERIES(1, 10000);
```

Exemplo prático - resultado I

```
SELECT contact_id, contact_name FROM contacts LIMIT 1;
Time: 0,805 ms

SELECT contact_id, contact_name FROM contacts ORDER BY contact_id LIMIT 1;
Time: 1,036 ms

SELECT contact_id, contact_name FROM contacts ORDER BY contact_name LIMIT 1;
Time: 65,730 ms

CREATE INDEX contacts_contact_name_idx ON contacts (contact_name);
CREATE INDEX contacts_company_name_idx ON contacts (company_name);
CREATE INDEX contacts_phone_idx ON contacts (phone);
CREATE INDEX contacts_email_idx ON contacts (email);

SELECT contact_id, contact_name FROM contacts ORDER BY contact_name LIMIT 1;
Time: 0,308 ms
```

Exemplo prático - resultado II

```
INSERT INTO contacts(contact_name, company_name, phone, email, details)
SELECT gen_string(5), gen_string(5), gen_number(11),
↪  gen_string(30)||'@'||gen_string(30)||'.'||gen_string(30), gen_string(1000)
FROM GENERATE_SERIES(1, 10000);
Time: 2312,693 ms (00:02,313)

DROP INDEX IF EXISTS contacts_contact_name_idx;
DROP INDEX IF EXISTS contacts_company_name_idx;
DROP INDEX IF EXISTS contacts_phone_idx;
DROP INDEX IF EXISTS contacts_email_idx;

INSERT INTO contacts(contact_name, company_name, phone, email, details)
SELECT gen_string(5), gen_string(5), gen_number(11),
↪  gen_string(30)||'@'||gen_string(30)||'.'||gen_string(30), gen_string(1000)
FROM GENERATE_SERIES(1, 10000);
Time: 2172,593 ms (00:02,173)
```

Referências

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Elsevier, 2004.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Addison Wesley, 2018.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados**. 3. ed. São Paulo: McGrawHill, 2008.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de bancos de dados**. 3. ed. São Paulo: Campus, 2007.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Documentation**. 2023. Disponível em: https://www.postgresql.org/docs/current/.