Sistemas de Banco de Dados

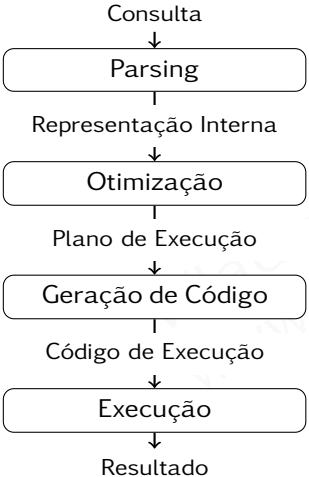
Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

PROCESSAMENTO DE CONSULTA

Processamento de Consulta

SGBDs **PROCESSAM, OTIMIZAM** e **EXECUTAM** consultas



- PARSING → análise sintática
  - VARREDURA → tokenização
  - ANÁLISE → regras gramaticais SQL
  - VALIDAÇÃO → metadados ∈ esquema
- OTIMIZAÇÃO → escolha de estratégia eficiente para execução da consulta
  - ÁRVORE DE CONSULTA → ou grafo de consulta é uma representação interna da consulta
  - PLANO DE EXECUÇÃO → estratégia de execução
- GERAÇÃO DE CÓDIGO → código compilado ou interpretado para execução

Processamento de Consulta: Tradução

Parsing → consulta é traduzida para álgebra relacional

- **Tradução** → decomposição em blocos de expressões SELECT-FROM-WHERE

```
SELECT  CPF, Nome FROM PROFESSOR WHERE Salario >
        (SELECT AVG(Salario) FROM PROFESSOR WHERE Sexo='M');
```

```
SELECT  CPF, Nome
  FROM  PROFESSOR
 WHERE  Salario > X;
```

$$\pi_{CPF,\,Nome}(\sigma_{Salario > X}(PROFESSOR))$$

```
SELECT  AVG(Salario)
  FROM  PROFESSOR
 WHERE  Sexo = 'M';
```

$$\gamma_{AVG(Salario)}(\sigma_{Sexo='M'}(PROFESSOR))$$

Processamento de Consulta: Algoritmos

Processar uma consulta envolve a escolha de algoritmos e estratégias a serem aplicados na execução de sequências de operações da álgebra relacional

- Ordenação → agregação, conjunto, junção e projeção (distinct)
- Pesquisa → junção e seleção
- Hashing → agregação, conjunto, junção, projeção e seleção

A escolha do algoritmo e estratégia adequados é feita pelo SGBD, dependendo da tecnologia de armazenamento e da organização de dados

- Memória primária livre → *hashing* em operação de junção
- Arquivo ordenado → pesquisa binária em operação de seleção

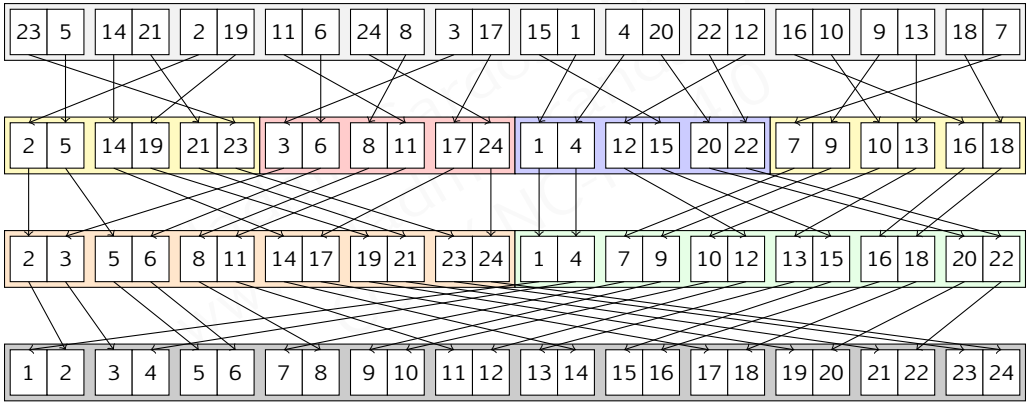
Algoritmos: Ordenação Externa

Estratégia de ordenação e intercalação (MERGE-SORT) de registros em disco

- ORDENAÇÃO → partes (*runs*) do arquivo são transferidas do disco para memória primária, ordenadas em memória primária e regravadas em disco
  - $B_M \rightarrow$ # *buffers* disponíveis em memória primária
  - $B_D \rightarrow$ # blocos do arquivo em disco
  - $R = \left\lceil \frac{B_D}{B_M} \right\rceil \rightarrow$ # *runs*
- INTERCALAÇÃO → mesclagem de *runs* ordenadas em disco
  - Grau $\rightarrow D = min((B_M - 1), R) \rightarrow$ # *runs* mescladas em cada passo
  - $S = \lceil \log_D R \rceil \rightarrow$ # passos de intercalação

Custo $O(n \log n) \rightarrow (2 \times B_D) + (2 \times B_D \times \lceil \log_D R \rceil)$

Exemplo de MERGE-SORT com 24 registros em 12 blocos e 3 *buffers*, realizando ordenação em $\lceil 12/3 \rceil = 4$ *runs* e intercalação de grau 2 em $\lceil log_2 4 \rceil = 2$ passos



Algoritmos: Ordenação Externa

Para ordenar um arquivo que ocupa 1024 blocos em disco usando 5 *buffers*:

- Ordenação → 205 *runs* ordenadas em disco
  - $R = \left\lceil \frac{1024}{5} \right\rceil = \lceil 204,8 \rceil = 205$
- Intercalação → grau 4 em 4 passos
  - $D = min((5-1), 205) = 4$
  - $S = \lceil \log_4 205 \rceil \approx \lceil 3,84 \rceil = 4$
    - Passo 1 → 205 *runs* mescladas 4 a 4
    - Passo 2 → 52 *runs* mescladas 4 a 4
    - Passo 3 → 13 *runs* mescladas 4 a 4
    - Passo 4 → 4 *runs* mescladas

Custo → $(2 \times 1024) + (2 \times 1024 \times 4) = 10.240$

Algoritmos: Projeção (π)

Estratégias envolvem eliminar registros duplicados

- Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordena-se o resultado da projeção e varre-se sequencialmente registros removendo duplicatas em registros adjacentes
- Hashing $\rightarrow O(n)$
  - Computa-se um endereço de partição (*bucket*) a partir de uma função *hash* sobre cada registro de resultado, alocando-o no *bucket* correspondente
  - Antes da alocação verifica-se se o registro já está presente no *bucket*, somente alocando-o se não estiver presente

Algoritmos: Seleção (σ)

Inúmeras estratégias possíveis, dependendo da existência de índices e da característica e complexidade da condição de seleção

- Arquivo Não Indexado
  - Pesquisa Linear $\rightarrow O(n)$
    - Recupera-se cada registro e verifica-se se valores em campos satisfazem a condição de seleção
  - Pesquisa Binária $\rightarrow O(\log n)$
    - Condição de seleção envolve comparação de $< \leq = \geq >$ em campo de ordenação
  - Hashing $\rightarrow O(1)$
    - Condição de seleção envolve comparação de $=$ em campo *hash*

Algoritmos: Seleção (σ)

- Arquivo Indexado
  - Índice Primário, de Agrupamento, Secundário ou Multinível $\rightarrow O(\log n)$
    - Condição de seleção envolve comparação de $< \leq = \geq >$ em campo de indexação
  - Índice Hash $\rightarrow O(1)$
    - Condição de seleção envolve comparação de $=$ em campo de indexação
- Seleção Conjuntiva $\rightarrow$ operador $\wedge$ na condição de seleção
  - Índice Composto $\rightarrow O(\log n)$
    - Condição de seleção envolve um subconjunto dos campos de indexação, desde que todos os campos iniciais estejam presentes

Algoritmos: Seleção (σ)

- Seleção Conjuntiva $\rightarrow$ operador $\wedge$ na condição de seleção
  - Índice Individual $\rightarrow O(\log n)$
    - Pesquisa-se no índice e verifica-se condições remanescentes da seleção
  - Índice Múltiplo $\rightarrow O(\log n)$
    - Pesquisa-se em cada índice secundário separadamente, realiza-se a intersecção de ponteiros recuperados, e verifica-se condições remanescentes da seleção
- Seleção Disjuntiva $\rightarrow$ operador $\vee$ na condição de seleção
  - Índice Múltiplo $\rightarrow O(\log n)$
    - Pesquisa-se em cada índice secundário separadamente e realiza-se a união de ponteiros recuperados. **Demanda** índice para cada campo na condição de seleção

Algoritmos: Operações de Conjunto ($\times \cup \cap -$)

Estratégias envolvem combinação de registros

- Força Bruta $\rightarrow O(n^2)$
  - Produto Cartesiano ($\times$) $\rightarrow$ combinam-se todos os registros de cada conjunto
- Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordenam-se os registros dos dois conjuntos, varrem-se os dois conjuntos simultâneamente e a operação de conjunto apropriada é efetuada
- Hashing $\rightarrow O(n)$
  - Computam-se endereços de *bucket* a partir de uma função *hash* para alocação de registros do menor conjunto
  - Computa-se a função *hash* para cada registro do outro conjunto
  - Aloca-se (união) ou desaloca-se (diferença ou intersecção) o registro no *bucket* correspondente de acordo com a operação de conjunto utilizada

Algoritmos: Agregação (γ)

Estratégias dependem da existência de índices e de campos de agrupamento

- Completa → campo de agrupamento (group by) não especificado
  - $O(n)$ → varre-se arquivo de dados ou de índice computando função
  - $O(\log n)$ → para índice B+ Tree no campo usado na função de agregação
- Particionada → campo de agrupamento (group by) especificado
  - Ordenação → $O(n \log n)$
    - Ordena-se arquivo pelo campo de agrupamento, varrendo-o e computando função de agregação para cada partição
  - Hashing → $O(n)$
    - Particiona-se o arquivo em *buckets* usando campo de agrupamento e computa-se função de agregação para cada *bucket*
  - Índice de Agrupamento → $O(n)$
    - Arquivo já particionado, bastando computar função de agregação

Algoritmos: Junção ($\bowtie$)

Estratégias dependem da existência de índices e da característica e complexidade da condição de junção

- Junção de *Loop* Aninhado $\rightarrow O(n^2)$
  - Força bruta
  - Para cada registro $r_i \in R$, recuperar cada registro $s_j \in S$
  - Combinar r_i e s_j se satisfazem a condição de junção
- Junção de *Loop* Único $\rightarrow O(n \log n)$
  - Aplicável caso haja índice em ao menos um arquivo
  - Para cada registro $r_i \in R$, onde R é o arquivo com maior custo de busca
  - Usar o índice em S para recuperar os registros que satisfazem a condição de junção, combinando-os com o registro r_i

Algoritmos: Junção ($\bowtie$)

- Junção Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordena-se arquivos por campos presentes na condição de junção
  - Varrem-se simultâneamente ambos os arquivos pelo campo de ordenação, combinando os registros que satisfazem a condição de junção
  - Registros em cada arquivo são acessados apenas uma vez
- Junção Hash $\rightarrow O(n)$
  - Para condição de junção com comparação de $=$
  - Varre-se o arquivo R de menor tamanho, aplicando uma função *hash* sobre o campo presente na condição de junção para criar *buckets* em memória
  - Para cada registro em S, use a função *hash* para encontrar registros em R que satisfazem a condição de junção, e combine-os com o registro de S

Algoritmos: Junção Externa ($\bowtie \ltimes \times$)

Estratégia envolve a combinação de estratégias de junção e de conjunto

1. Junte os arquivos R e S usando a melhor estratégia de junção
2. Use a operação DIFERENÇA para encontrar os registros do arquivo R (aberto) não presentes no resultado da junção
3. Produto cartesiano dos registros não presentes de R com registro NULO de S
4. Use a operação UNIÃO para combinar os registros das etapas 1 e 3

Custo $\rightarrow (\bowtie) + (-) + (\times) + (\cup)$

Referências Bibliográficas

[1] Elmasri, Ramez; Navathe, Sham. *Fundamentals of Database Systems*. 7ed. Pearson, 2016.

[2] Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S. *Database System Concepts*. 6ed. McGraw-Hill, 2011.

[3] Date, Christopher J. *An Introduction to Database Systems*. 8ed. Pearson, 2004.