Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

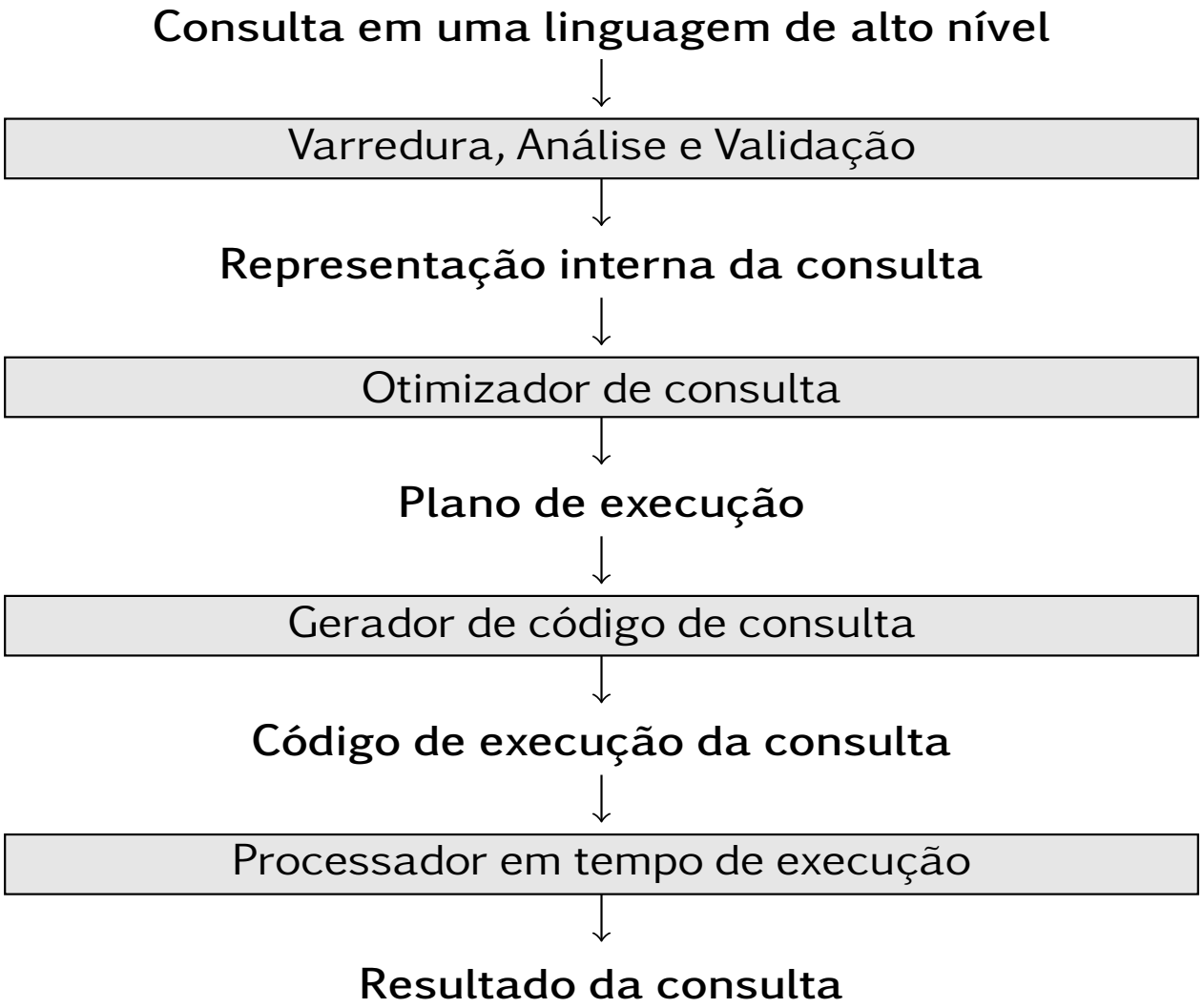
Seção 19

Processamento e Otimização de Consultas

Processamento de Consultas

- SGBDs utilizam técnicas para **processar**, **otimizar** e **executar** consultas
- Consultas precisam ser **lidas**, **analisadas** e **validadas**
  1. Leitura (Varredura) → identificação de *tokens* de consulta, como palavras-chave SQL e nomes de relações e atributos
  2. Análise → verificação sintática, determinando se a consulta segue as regras gramaticais da SQL
  3. Validação → verificação de validade de nomes de atributos e relações no esquema do BD
- Para cada consulta, é criada uma representação interna em memória, geralmente na forma de **árvore** ou **grafo**
- Plano de execução → estratégia de execução da consulta idealizada pelo SGBD para obtenção de resultados
- Otimização → processo (interno) de escolha de uma estratégia eficiente de execução

Processamento de Consultas



Processamento de Consultas

- O gerador de código de consulta produz o código, compilado ou interpretado, que será executado pelo processador em tempo de execução para produção do resultado final
- A descoberta da estratégia ótima para execução da consulta pode demandar elevado custo computacional
- Os otimizadores de consulta dos SGBDs comumente não garantem a produção do melhor plano de execução possível

Tradução de Consultas

- Para se otimizar uma consulta, ela primeiramente precisa ser traduzida para uma expressão equivalente em álgebra relacional e representada como uma árvore (ou grafo) de consulta
- A consulta precisa ser decomposta em **blocos**:
  - Unidades básicas a serem traduzidas em operadores algébricos para otimização
  - Contém uma única expressão SELECT-FROM-WHERE e, caso exista, GROUP BY e HAVING.
- Como exemplo, consideremos a consulta:

```
SELECT Unome, Pnome
FROM   FUNCIONARIO
WHERE  Salario > (SELECT MAX(Salario)
                  FROM   FUNCIONARIO
                  WHERE  Dnr=5);
```

Tradução de Consultas

- Para esse caso, temos dois blocos de consulta. O bloco de consulta externo:

```
SELECT Unome, Pnome
FROM   FUNCIONARIO
WHERE  Salario > X;
```

- Que pode ser traduzido para a seguinte expressão da álgebra relacional:

$$\pi_{Unome,Pnome}(\sigma_{Salario>X}(FUNCIONARIO))$$

Tradução de Consultas

- E o bloco de consulta interno:

```
SELECT MAX(Salario)
FROM   FUNCIONARIO
WHERE  Dnr = 5;
```

- Que pode ser traduzido para a seguinte expressão da álgebra relacional:

$$\Im_{MAX(Salario)}(\sigma_{Dnr=5}(FUNCIONARIO))$$

- Apresenta uma consulta aninhada - só precisa ser avaliado uma vez para produzir o resultado.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

ALGORITMOS PARA ORDENAÇÃO EXTERNA

Algoritmos para ordenação externa

- Ordenação dos resultados da consulta quando a cláusula ORDER BY for especificada.

Sorting (Intercalação):

- Ordenação dos resultados da consulta quando houver as cláusulas: ORDER BY, Junção, Projeção (para eliminação de duplicata - DISTINCT);
- Pode ser evitada se um índice apropriado houver no atributo de arquivo desejado.

Algoritmos para ordenação externa

Sorting - Intercalação Externa:

- Adequado a grandes arquivos de registros armazenados no disco (que não cabem totalmente na memória principal);
- Estratégia ordenação-intercalação (sort-merge);
- Exige buffer na memória principal, onde se realiza a classificação e mesclagem;
- Dividido em buffers individuais.
  - Mantém o conteúdo de exatamente um bloco de disco.

SORTING - INTERCALAÇÃO EXTERNA:

1. Fase de ordenação (classificação)
    - Classificar pequenos "subarquivos";
    - Os pedaços do arquivo são lidos para memória principal, ordenados e regravados no disco (como subarquivos temporários);
    - Tamanho de cada pedaço ($\mathbf{n_R}$) definido pelo número de blocos de arquivo ($\mathbf{b}$);
    - Número de blocos de arquivo ($\mathbf{n}$) definido pelo espaço de buffer disponível ($\mathbf{n_B}$).

SORTING - INTERCALAÇÃO EXTERNA:

1. Fase de ordenação (classificação) - Exemplo:

   - Se o número de buffers disponíveis na memória principal $n_B = 5$ **blocos** e o tamanho do arquivo $b = 1.024$ **blocos**, então $n_r = (b/n_B) = 205$ pedaços iniciais.

   - Após a ordenação, 205 pedaços ordenados são armazenados como subarquivos temporários no disco.

SORTING - INTERCALAÇÃO EXTERNA:

2 Fase de intecalação (mesclagem):

- Pedaços ordenados mesclado em passos de intercalação;
- Cada passo de intercalação pode ter uma ou mais etapas;
- **Grau de intercalação (d_M)** = número de subarquivos ordenados que podem ser mesclados em cada etapa, que contém:
  - **Bloco de buffer** - mantém um bloco de disco em cada subarquivo;
  - **Bloco de buffer adicional** - mantém um bloco de disco do resultado da intercalação.

Sorting - Intercalação Externa:

2 Fase de intecalação (mesclagem):

- $\mathbf{d_M}$ é o menor de $(\mathbf{n_B}\text{-}1)$ e $\mathbf{n_R}$.
- **Número de passos da intercalação** = $(\log_{d,M}(n_R))$
- Exemplo:
  - $\mathbf{n_B} = 5$ e $\mathbf{d_M} = 4$, de modo que os 205 pedaços iniciais ordenados seriam intercalados 4 de cada vez em cada etapa em 52 subarquivos, ao final do primeiro passo de intercalação;
  - Os 52 arquivos ordenados serão intercalados quatro de cada vez em 13 arquivos ordenados, que depois intercalados em 4 arquivos ordenados e, por fim, para um arquivo totalmente ordenado.

SORTING - INTERCALAÇÃO EXTERNA:

2 Fase de intecalação (mesclagem):

- O desempenho pode ser medido a partir do número de leituras e gravações de bloco de disco, antes que a ordenação seja concluída. Custo expresso na fórmula:

$$(2 * b) + (2 * b * (\log_{d.M} n_R)), \text{ onde}$$

- $(2 * b)$ = Número de acessos de blocos para a fase de ordenação.
- $(2 * b *$ = Número de acessos de blocos para a fase de intercalação.
- $(\log_{d.M} n_R))$ = Número de passos de intercalação.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Seleção

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere o seguinte banco de dados relacional (parte 1):

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere o seguinte banco de dados relacional (parte 2):

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere as seguintes operações aplicadas ao banco de dados apresentado no slide anterior:

OP1: $\sigma_{Cpf='12345678966'}(FUNCIONARIO)$

OP2: $\sigma_{Dnumero>5}(DEPARTAMENTO)$

OP3: $\sigma_{Dnr=5}(FUNCIONARIO)$

OP4: $\sigma_{Dnr=5\ AND\ Salario=30000\ AND\ Sexo='F'}(FUNCIONARIO)$

OP5: $\sigma_{Fcpf='12345678966'\ AND\ Pnr=10}(TRABALHA\_EM)$

- A seguir, serão apresentados os métodos para varredura de arquivos para seleção simples.

Algoritmos para operação de Seleção

1. Pesquisa Linear
   - Recupera cada registro do arquivo e testa se seus valores de atributo satisfazem a condição de seleção;
   - Diferente dos demais métodos (que dependem de um caminho de acesso apropriado no atributo usado na condição de seleção), se aplica a qualquer arquivo.
2. Pesquisa Binária
   - O mais eficiente se a condição envolver uma comparação de igualdade em um atributo chave, no qual o arquivo é ordenado;
   - Exige o arquivo seja ordenado no atributo de pesquisa.

Algoritmos para operação de Seleção

3 Usando um índice primário

- Utilizar se a condição de seleção envolver uma comparação de igualdade entre este atributo.

4 Usando uma chave hash

- Utilizar se a condição de seleção envolver uma comparação de igualdade entre esta chave.

Algoritmos para operação de Seleção

5 Usando um índice primário para recuperar vários registros

- Nos casos de condição de comparação >, >=, < ou <= em um campo chave com um índice primário (ex: Dnumero > 5):
  1. Utilizar o índice para encontrar o registro que satisfaz a igualdade (ex: Dnumero = 5);
  2. Recuperar todos os registros subsequentes no arquivo ordenado;
- Aplicáveis a consultas de intervalos.

Algoritmos para operação de Seleção

6 Usando um índice de agrupamento para recuperar vários registros

- Se a condição de seleção envolver uma comparação de igualdade em um atributo não chave com índice de agrupamento, utilizar o índice para recuperar todos os registros que satisfazem a seleção.

Algoritmos para operação de Seleção

7 Usando um índice secundário (B^+-tree em uma comparação de igualdade

- Utilizado para recuperar um único registro se o campo de índice for uma chave ou para recuperar múltiplos registros se o campo de índice não for uma chave;
- Aplicáveis a consultas de intervalos.

- A seguir, serão apresentados os métodos de pesquisa para selação complexa (ex: presença de conectivo lógico AND).

Algoritmos para operação de Seleção

8 Seleção conjuntiva usando um índice individual

- Se o atributo tiver caminho de acesso que permita os métodos (2) a (7):
  1. Utilizar esta condição para recuperação;
  2. Verificar se cada registro satisfaz as condições simples restantes da seleção.

9 Seleção conjuntiva usando índice composto

- Se dois ou mais atributos estiverem envolvidos nas condições de igualdade na seleção conjuntiva e um índice composto existe nos campos combinados, pode-se usar o índice diretamente.

Algoritmos para operação de Seleção

10 Seleção conjuntiva por untersecção de ponteiros de registro

- Os índices secundários devem estar disponíveis em mais de um dos campos envolvidos na seleção simples e devem incluir ponteiros de registros;
- A intersecção desses conjuntos de ponteiros gera os ponteiros de registro que satisfazem a condição de seleção conjuntiva;
- Se apenas uma das condições tiverem índices secundários, cada registro recuperado é testado mais de uma vez para verificar se o mesmo satisfaz a condição.

Algoritmos para operação de Seleção

- Uma condição disjuntiva (conectivo lógico OR) é complexo para se otimizar e processar. Considere a OP4′:

$$\sigma_{Dnr=5\ AND\ Salario=30000\ AND\ Sexo='F'}(FUNCIONARIO)$$

  - Neste caso, pouca otimização pode ser feita, pois os registros que satisfazem a condição são a união dos registros que satisfazem as condições individuais;
  - Se qualquer uma das condições não tiver um caminho de acesso, deve-se utilizar a pesquisa linear.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Junção

Algoritmos para operação de Junção

- Junções multivias: envolvem mais de dois arquivos.
  - O número de vias para executar junções multivias cresce muito rapidamente.
  - A seguir, serão apresentadas técnicas somente para junções de duas vias, a partir da junção no seguinte formato:

$$R \bowtie_{A=B} (S)$$

Onde A e B são atributos de junção, que devem ser atributos compatíveis por domínio de R e S, respectivamente.

Algoritmos para operação de Junção

- As operações a seguir serão utilizadas para as demais discussões:

$$\text{FUNCIONARIO} \overset{\underline{\text{OP6:}}}{\bowtie}_{Dnr=Dnumero} (DEPARTAMENTO)$$

$$\text{DEPARTAMENTO} \overset{\underline{\text{OP7:}}}{\bowtie}_{Cpf\_ger=Cpf} (FUNCIONARIO)$$

Algoritmos para operação de Junção

J1 Junção de loop aninhado

- Algoritmo padrão - não existe quaisquer caminhos de acesso especiais.

1. Para cada registro **t** em **R**, recupere cada registro **s** de **S**;
2. Testar se os dois registros satisfazem a relação **t**[A] = **s**[B].

J2 Junção de único loop

- Utiliza-se estrutura de acesso para recuperar os registros correspondentes;
- Somente se houver índice para um dos dois atributos.

1. Para um dos dois atributos, recupere cada registro **t** em **R**;
2. Usar a estrutura de acesso para recuperar diretamente os registros correspondentes **s** de **S** que satisfazem **s**[B] = **t**[A].

Algoritmos para operação de Junção

J3 Junção ordenação-intercalação

- Desejável que os registros **R** e **S** estejam fisicamente ordenados por valor dos atributos de junção A e B;
- Os dois arquivos são varridos simultâneamente, na ordem dos atributos de junção, combinando os registros;
- Pares de blocos de arquivo são copiados para buffers de memória na ordem e os registros de cada arquivo são varridos apenas uma vez cada.

Algoritmos para operação de Junção

J4 Junção de partição-hash

- Os registros de **R** e **S** são particionados em arquivos menores;
- O particionamento é feito usando a mesma função hashing **h** no atributo de junção **A** de **R** (para particionamento do arquivo **R**) e **B** de **S** (para particionamento do arquivo **S**);
- **Fase de particionamento** e **Fase de investigação**.

Algoritmos para operação de Junção

J4 Junção de partição-hash

1. Fase de particionamento
   - Cria-se hashes dos registros do menor arquivo para as diversas partições de **R**;
   - A coleção de registros com o mesmo valor de **h(A)** é colocada na mesma partição, bucket de hash em uma tabela hash na memória principal.

2. Fase de investigação
   - Cria-se o hash de cada registro do arquivo **S** usando a mesma função de hash **h(B)** para investigar o bucket apropriado;
   - O registro é então combinado com todos os correspondentes de **R** nesse bucket.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Projeção

Algoritmos para operação de Projeção

$$\text{pi}_{\text{<lista atributos>}}(R)$$

- **Se <lista atributos> incluir uma chave da relação R**
  - É simples de se implementar, pois nesse caso o resutlado da operação terá o mesmo número de tuplas que R.
- **Se <lista atributos> não incluir uma chave da relação R**
  - As tuplas duplicadas devem ser eliminadas.

Algoritmos para operação de Projeção

A duplicidade pode ser eliminada...

- Ao ordenar o resultado e depois eliminar as tuplas duplicadas (que aparecem consecutivamente após a ordenação).
- A partir do hashing:
  - A medida que cada registro gera um hash e é inserido em um bucket do arquivo hash, ele é comparado com os demais que já estão no bucket.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operações de conjunto

Algoritmos para operações de conjunto

Produto Cartesiano

- Seu resultado (R x S) inclui um registro para cada combinação de registros de R e S;
- Importante evitar esta operação e substituí-la por outras operações, como a junção, durante a otimização da consulta.

União, Intersecção e Diferença de Conjunto

- Relações compatíveis no tipo;

Algoritmos para operações de conjunto

União, Intersecção e Diferença de Conjunto

- Utilizar várias técnicas de ordenação-intercalação:
  1. As duas relações são ordenadas nos mesmos atributos;
  2. Uma única varredura por cada relação é suficiente para produzir o resultado.

- Também pode-se utilizar o hashing:
  1. Uma tabela é varrida e particionada em uma tabela hash na memória com buckets;
  2. Os registros na outra tabela são varridos um de cada vez e usados para investigar a partição apropriada.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Implementando operações de agregação

Implementando operações de agregação

- Operações de agregação (**MIN, MAX, COUNT, AVERAGE, SUM**) quando aplicadas a uma tabela inteira, podem ser calculados por uma varredura de tabela ou usando um índice apropriado.
- Exemplo:

```
SELECT MAX(Salario)
FROM   FUNCIONARIO
```

- Se houver índice de B^+-tree (crescente) em **Salario** para a relação **FUNCIONARIO**, então o otimizador pode decidir sobre o uso do índice Salario para procurar.

Implementando operações de agregação

- No uso do **GROUP BY**, o operador de agregação deve ser aplicado separadamente a cada grupo de tuplas. Logo, a tabela precisa primeiro ser particionada em subconjuntos de tuplas.
  - Cada partição com o mesmo valor para os atributos de agrupamento.
- Exemplo:

```
SELECT Dnr, AVG(Salario)
FROM FUNCIONARIO
GROUP BY Dnr;
```

Implementando operações de agregação

1. Utilizar a ordenação ou o hashing nos atributos de agrupamento para particionar os arquivos;
2. Calcular a função de agregação para as tuplas de cada grupo obtido em (1). Ou seja, no conjunto de tuplas de **FUNCIONARIO**, cada número de departamento seria agrupado em uma partição e o salário médio calculado para cada grupo.

- Se houver um índice de agrupamento nos atributos de agrupamento, os registros já estarão particionados.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Implementando operações de junção externa

Implementando operações de junção externa

- Exemplo:

```
SELECT Unome, Pnome, Dnome
FROM (FUNCIONARIO LEFT OUTER JOIN DEPARTAMENTO
      ON Dnr=Dnumero);
```

- O resultado desta consulta é uma tabela de nomes de funcionários e seus departamentos associados;
- Pode ser calculada modificando-se um dos algoritmos de junção;
- Também pode ser calculada ao executar uma combinação de operadores da álgebra relacional.

Implementando operações de junção externa

- Sequência de operações relacionadas a consulta do exemplo:

1. Calcule a JUNÇÃO das tabelas **FUNCIONARIO** e **DEPARTAMENTO**:

$$A \longleftarrow FUNCIONARIO \bowtie_{Dnr=Dnumero} (DEPARTAMENTO)$$
$$B \longleftarrow \pi_{Unome,\ Pnome,\ Dnome}(A)$$

2 Ache as tuplas de **FUNCIONARIO** que não aparecem no resultado da JUNÇÃO:

$$C \longleftarrow \pi_{Unome,\ Pnome}(FUNCIONARIO)$$
$$D \longleftarrow \pi_{Unome,\ Pnome}(B)$$
$$E \longleftarrow C - D$$

Implementando operações de junção externa

- Sequência de operações relacionadas a consulta do exemplo:

3 Preencha cada tupla em **E** com um campo Dnome NULL:

$E \longleftarrow E \ x \ NULL$

4 Aplique a operação UNION a **B** e **E** para produzir o resultado JUNÇÃO EXTERNA A ESQUERDA:

$RESULT \longleftarrow B \cup E$

- O custo, portanto, seria a soma dos custos das etapas associadas.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Combinando operações com pipelining

Combinando operações com pipelining

- Uma consulta em SQL é traduzida para uma expressão da álgebra relacional (sequência de operações relacionais);
- Se cada operação for executada uma de cada vez, é necessário gerar arquivos temporários no disco para manter os resultados;
  - Gera um overhead excessivo.
- Para reduzir a quantidade desses arquivos, gera-se um código de execução de consulta que corresponde a algoritmos para combinações de operações em uma consulta.

Combinando operações com pipelining

- A partir deste código de execução da consulta é gerado apenas um arquivo de resultado.
  - Processo chamado de **pipelining** ou **processamento baseado em fluxo**.
- Normalmente, cria-se um código de execução de consultas de maneira dinâmica para implementar múltiplas operações.
  - Combina vários algoritmos que correspondem Ãăs operações individuais.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Usando heurísticas na otimização da consulta

Usando heurísticas na otimização da consulta

Árvore de consulta

- Estrutrura de dados de árvore que corresponde a uma expressão da álgebra relacional;
- Representa as **relações de entrada** da consulta como **nós folha**;
- Representa as **operações** da álgebra relacional como **nós internos**.

Usando heurísticas na otimização da consulta

Árvore de consulta

- Sua execução consiste na execução de uma operação de nó interno sempre que seus operandos estão disponíveis e depois na substituição desse nó interno pela relação que resulta da execução da operação;

- Termina quando a operação do nó raíz é executada e produz a relação de resultado para a consulta.

Árvore de consulta - Exemplo

- A seguir, um exemplo de consulta a partir do esquema relacional EMPRESA, em álgebra relacional, SQL e em árvore de consulta.

- **Para cada projeto localizado em ′Mauá′, recuperar o número do projeto, o número do departamento de controle, o sobrenome, o endereço e a data de nascimento do gerente do departamento.**

Árvore de consulta - Exemplo

- Consulta representada em Álgebra Relacional:

$$A \longleftarrow \sigma_{Projlocal='Mau'}(PROJETO)$$
$$B \longleftarrow A \bowtie_{Dnum=Dnumero} (DEPARTAMENTO)$$
$$C \longleftarrow B \bowtie_{Cpf\_ger = Cpf} (FUNCIONARIO)$$
$$D \longleftarrow \pi_{Projnumero,\ Dnum,\ Unome,\ Endereco,\ Data\_nasc}(C)$$

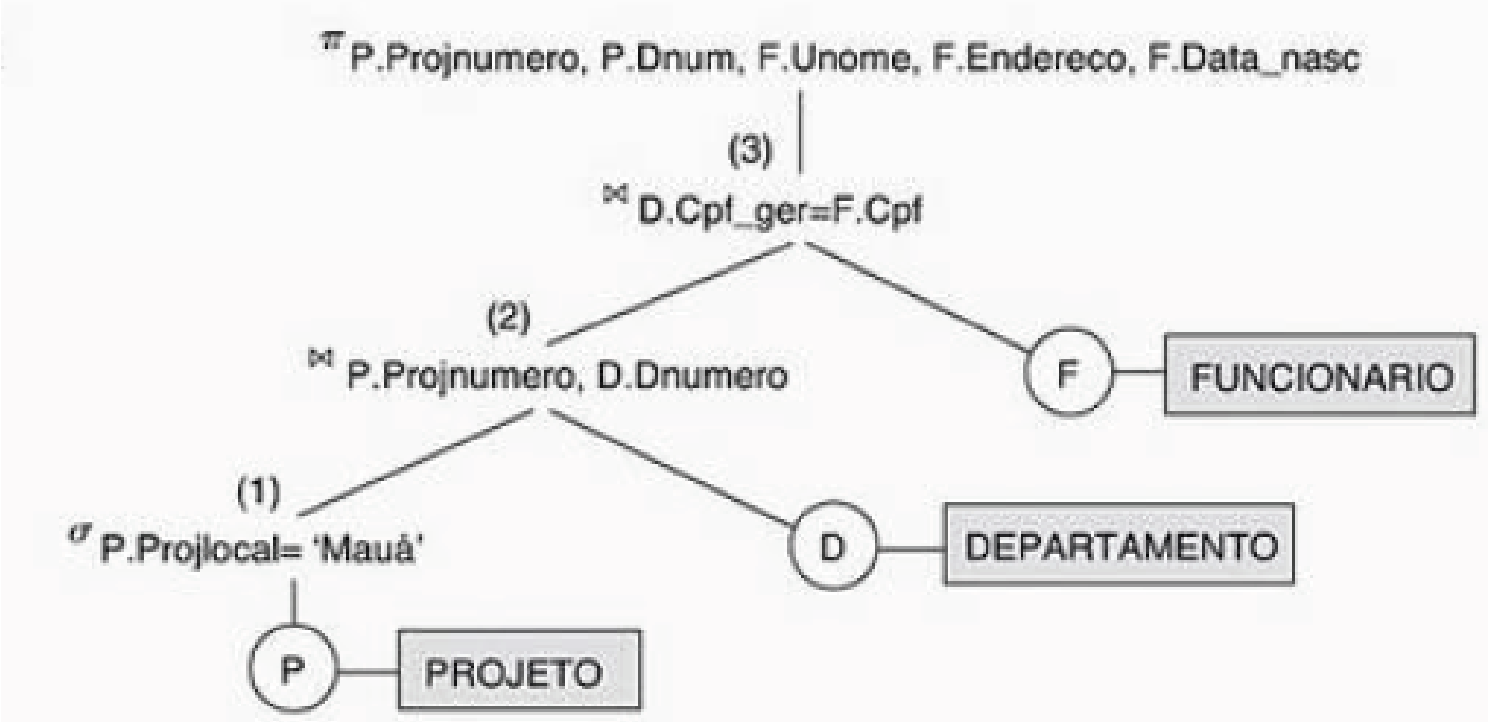
Árvore de consulta - Exemplo

- Consulta representada em SQL:

```
SELECT P.Projnumero,
       P.Dnum,
       F.Unome,
       F.Endereco,
       F.Data_nasc
FROM   PROJETO      AS P,
       DEPARTAMENTO AS D,
       FUNCIONARIO  AS F
WHERE P.Dnum      = D.Dnumero
  AND D.Cpf_ger   = F.Cpf
  AND P.Projlocal = 'Maua';
```

Árvore de consulta - Exemplo

- Consulta representada em árvore:



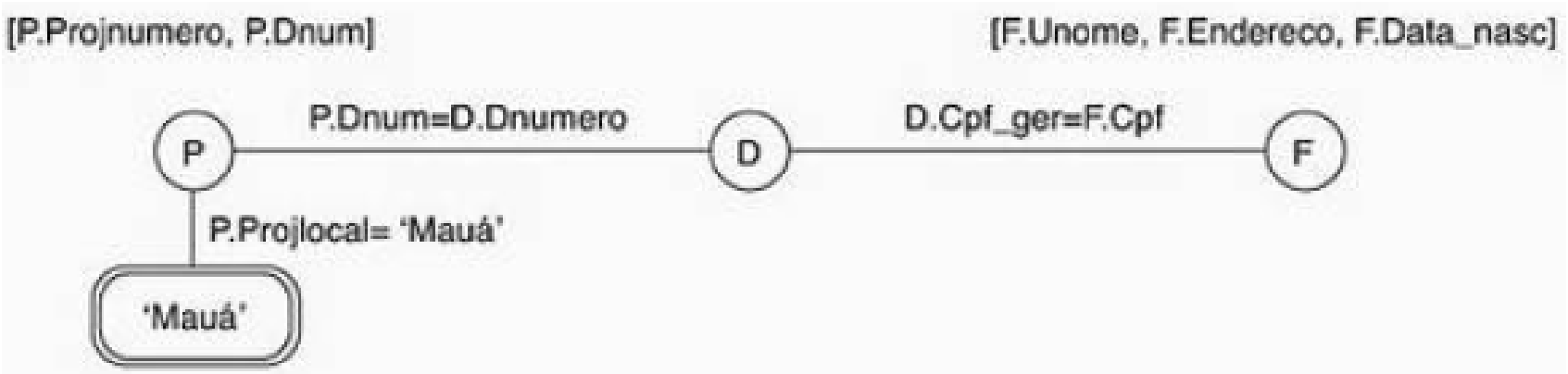
Usando heurísticas na otimização da consulta

Grafo de consulta

- As **relações de consulta** são representadas por **nós de relação**;
- Os **valores constantes** são representados por **nós de constantes**;
- As **condições de seleção** são representadas pelas **arestas**;
- Os **atributos** a serem recuperados são exibidos entre colchetes, acima das relações correspondentes;
- Não indica a ordem sobre quais operações realizar primeiro.

Grafo de consulta - Exemplo

- Consulta representada em grafo:



Usando heurísticas na otimização da consulta

- Processo de uma consulta (incluindo passos para otimização de desempenho):

  1. A varredura e o analisador gera primeiro uma estrutura de dados - **representação inicial de consulta**;

  2. A representação inicial é otimizada de acordo com regras heurísticas - **representação da consulta otimizada**;

  3. É gerado um plano de execução para executar grupos de operações com base nos caminhos de acesso disponíveis nos arquivos envolvidos.

Usando heurísticas na otimização da consulta

- A seguir, algumas técnicas de otimização que aplicam regras heurísticas para modificar a representação interna de uma consulta para melhorar seu desempenho.
  - **Principal: aplicar operações SELEÇÃO e PROJEÇÃO antes da JUNÇÃO ou outras operações binárias.**
- O otimizador de consulta heurística transforma a árvore de consulta inicial em uma árvore de consulta final equivalente, eficiente para execução.
  - Deve incluir regras para equivalência entre expressões da álgebra relacional.

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

1. Cascata de σ
   - Uma condição de seleção conjuntiva pode ser desmembrada em uma cascata (sequência) de σ como operações individuais.

2. Comutatividade de σ
   - A operação σ é comutativa:

     $\sigma_{c1}(\sigma_{c2}(R))$ equivale a $\sigma_{c2}(\sigma_{c1}(R))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

3 Cascata de π
  - Em uma cascata (sequência) de operações π, todas podem ser ignoradas, menos á última:

    $\pi_{Lista_1}(\pi_{Lista_2}(...(\pi_{Lista_n}(R))...))$ equivale a $\pi_{Lista_1}(R)$

4 Comutação de σ com π
  - Se a condição de seleção c envolve apenas os atributos A_1, ..., A_n na lista de projeção, as duas operações podem ser comutadas:

    $\pi_{A_1 . A_2 . ..., A_n}(\sigma_c(R))$ equivale a $\sigma_c(\pi_{A_1 . A_2 . ..., A_n}(R))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional:**

5 Comutatividade de $\bowtie$ (e **x**)

- A operação de junção é comutativa, assim como a operação X:

  $R \bowtie_c S$ *equivalea* $S \bowtie_c R$

  R x S equivale a S x R

6 Comutação de σ com $\bowtie$

- Se todos os atributos na condição de seleção **c** envolvem apenas os atributos de uma das relações sendo juntadas, as duas operações podem ser comutadas da seguinte forma:

  $\sigma_c(R \bowtie S)$ *equivale a* $(\sigma_c(R)) \bowtie S$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional:**

7 Comutação de π com $\bowtie$ (ou x)

- Suponha que a lista de projeção seja $L = A_1, ..., A_n, B_1, ..., B_m$, onde $A_1, ..., A_n$, são os atributos de R e $B_1, ..., B_m$ são atributos de S.

- Se a condição de junção **c** envolver apenas atributos de L, as duas operações podem ser comutadas conforme:

$$\pi_L(R \bowtie_c S) \underline{\textit{equivale a}} (\pi_{A_1, ..., A_n}(R)) \bowtie_c (\pi_{B_1, ..., B_M}(S))$$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

8 Comutatividade das operações de conjunto

- As operações de conjunto ∩ e ∪ são comutativas, mas - (diferença) não é.

9 Associatividade de ⋈, x, ∪ e ∩

- Essas quatro operações são associativas individualmente.

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

10 Comutatividade de σ com operações de conjunto

- A operação σ comuta com $\cup$, $\cap$ e -. Se θ indicar qualquer uma dessas três operações, temos:

  $\sigma_c(R\ \theta\ S)$ equivale a $(\sigma_c(R))\ \theta\ (\sigma_c(S))$

11 A operação π comuta com $\cup$

- $\pi_L(R\ \cup\ S)$ equivale a $(\pi_L(R))\ \cup\ (\pi_L(S))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

12 Convertendo uma sequência (σ, x) em $\bowtie$

- Se a condição c de um σ que segue um x corresponde a uma condição de junção, converta a sequência (σ, x) em $\bowtie$, da seguinte forma:

  $(\sigma_c(R \times S))$ equivale a $(R \bowtie_c S)$

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

1. Usando a Regra 1, quebre quaisquer operações de SELEÇÃO com condições conjuntivas em uma cascata de operações;

2. Usando as Regras 2, 4, 6 e 10, mova cada operação SELEÇÃO o mais baixo possível na árvore de consulta (que for permitido pelos atributos na condição de seleção);

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

3 Usando as Regras 5 e 9, reorganize os nós folha a partir dos critérios:

- Posicione as relações do nó folha com as operações SELEÇÃO mais restritivas;
- Garanta que a ordenação dos nós folha não cause operações de PRODUTO CARTESIANO.

4 Usando a Regra 12, combine uma operação PRODUTO CARTESIANO com uma operação SELEÇÃO subsequente na árvore para uma operação JUNÇÃO, se for o caso;

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

5 Usando as Regras 3, 4, 7 e 11, desmembre e mova as listas de atributos de projeção para baixo da árvore no máximo possível, criando novas operações de PROJEÇÃO;

6 Identifique subárvores que representam grupos de operações que podem ser executados por um único algoritmo.

Esboço de algoritmo de otimização algébrica heurística - Resumo

- Aplicar primeiro as operações que reduzem o tamanho dos resultados intermediários;
- As operações SELEÇÃO e JUNÇÃO devem ser executadas antes de outras operações semelhantes;
- Reordenação dos nós folha da árvore, enquanto evita produtos cartesianos, e do ajuste do restante da árvore.

Usando heurísticas na otimização da consulta

- O **plano de execução** inclui:
  - Informações sobre os métodos de acesso disponíveis em cada relação;
  - Algoritmos a serem usados no processamento dos operadores relacionais representados em árvore.

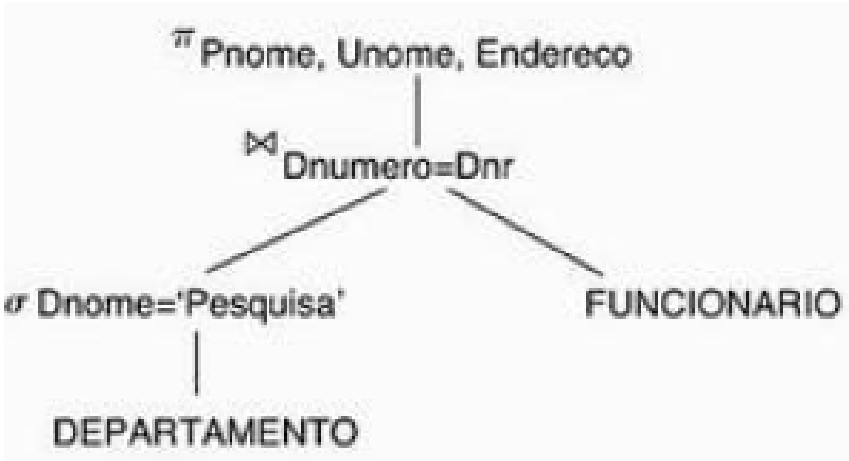
Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- Considere a consulta abaixo:

$$A \longleftarrow \sigma_{Dnome = 'Pesquisa'}(DEPARTAMENTO)$$
$$B \longleftarrow A \bowtie_{Dnumero = Dnr} (FUNCIONARIO)$$
$$C \longleftarrow \pi_{Pnome,\ Unome,\ Endereco} (B)$$

- Árvore de consulta correspondente:



Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- O otimizador escolhe:

  1. Uma busca de índice para a operação SELEÇÃO em DEPARTAMENTO;

  2. Um algoritmo de junção de único loop que percorra todos os registros no resultado da operação SELEÇÃO em DEPARTAMENTO para a operação de junção;

  3. Uma varredura do resultado de JUNÇÃO para a entrada do operador PROJEÇÃO.

Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- Pode especificar uma **avaliação materializada**
  - O resultado é armazenado como uma relação temporária, ou seja, fisicamente materializado.
- **Avaliação em pipeline**
  - A medida que as tuplas resultantes de uma operação são produzidas, são encaminhadas para a próxima operação na sequência da consulta;
  - Economia de custo por não ter de gravar os resultados imediatos em disco;
  - Não tem de ler os registros de volta para a próxima operação.

Otimização de consulta baseada em custo

- O otimizador limita o número de estratégias de execução a serem consideradas; caso contrário, muito tempo será gasto.
- Essa técnica é mais adequada para **consultas compiladas**, nas quais a otimização é feita na compilação e o código da estratégia de execução resultante é executado em tempo de execução).
- Para **consultas interpretadas**, uma otimização em escala total pode atrasar o tempo de resposta.

Componentes de custo para execução de consulta

1. **Custo de acesso ao armazenamento secundário:** Também conhecido como ***custo de E/S (entrada/saída) de disco***. O custo de procurar registros em um arquivo de disco depende do tipo de estruturas de acesso nesse arquivo. Além disso, fatores como se os blocos de arquivo estão alocados consecutivamente ou espalhados afetam o custo de acesso. Ênfase principal para banco de dados grandes.

2. **Custo de armazenamento em disco:** Custo de armazenar no disco quaisquer arquivos intermediários que sejam gerados por uma estratégia de execução para a consulta.

Componentes de custo para execução de consulta

3. **Custo de computação:** Custo de realizar operações na memória em registros dentro dos buffers de dados durante a execução da consulta. Também é conhecido como ***custo de CPU*** (unidade central de processamento). Ênfase de banco de dados menores e distribuídos.

4. **Custo de uso da memória:** Custo que diz respeito ao número de buffers da memória principal necessários durante a execução da consulta.

5. **Custo de comunicação:** Custo de envio da consulta e seus resultados do local do BD até o local ou terminal onde a consulta foi originada.

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- Tamanho de cada arquivo
- Para um arquivo cujos registros são do mesmo tipo:
  - **Número de registros (tuplas)** *(r)*
  - **Tamanho do registro (médio)** *(R)*
  - **Número de blocos do arquivo** *(b)*
- **Fator de bloco** *(bfr)*
- A ***organização de arquivo primária*** para cada arquivo

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- O **número de níveis** (x) de cada índice multinível (primário, secundário ou de agrupamento) para funções de custo que estimam o número de acessos de bloco.
- **Número de blocos de índice de primeiro nível** ($bL1$)
- **Número de valores distintos** (d) de um atributo
- **Seletividade** (sl), fração de registros que satisfazem uma condição de igualdade no atributo.

Algoritmos para processamento e otimização de consulta

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- Estimativa da **cardinalidade de seleção** ***(s = sl * r)*** de um atributo, número *médio* de registros que satisfarão uma condição de seleção de igualdade nesse atributo.
  - Para um ***atributo chave, d = r, sl = 1/r e s = 1***
  - Para um **atributo não chave**, ***d*** valores distintos distribuídos uniformemente entre os registros, *sl = (1/d), s = (r/d)*
- O otimizador de consulta precisará de valores próximos, mas não necessariamente atualizados até o último minuto desses parâmetros para uso na estimativa do custo.

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan