



Otimização de Consultas

André Luís Schwerz
Rafael Liberato Roberto



Questão

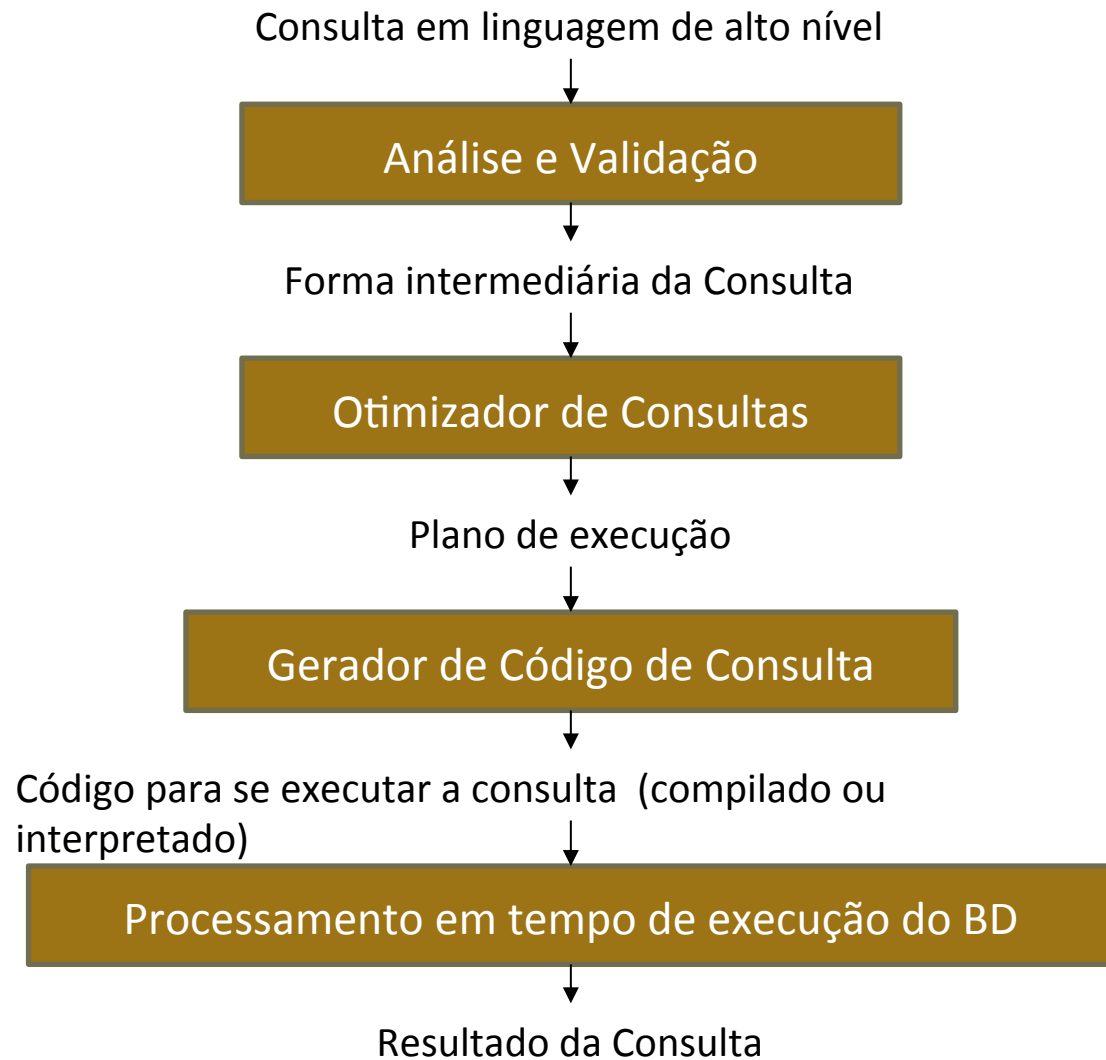
Como as consultas SQL são processadas pelo SGBD?

```
EMPREGADO (idE, nomeE, salário, RG-sup, IdDepto)
DEPARTAMENTO (idD, nomeD, idGerente, inicio-ger)
```

```
SELECT idE FROM EMPREGADO, DEPARTAMENTO
WHERE idDepto = idD AND idGerente = 1
```

```
SELECT nomeD FROM EMPREGADO, DEPARTAMENTO
WHERE idDepto = idD AND nomeE= 'Andre'
```

Introdução



Passos para otimização de consultas

- Tradução de consultas SQL – Álgebra relacional
- Algoritmos básicos para execução de consultas:
 - ordenação externa
 - seleção
 - junção
 - projeção e conjunto
 - agregação
- Heurística na otimização de consultas
- Estimativa de custos

Tradução SQL – Álgebra relacional

- Uma consulta SQL é traduzida em uma expressão algébrica estendida equivalente
- É estendida porque deve incluir os operadores de agregação:
 - MAX, MIN, SUM, AVG e COUNT
- As consultas SQL são decompostas em blocos de consulta
- Cada bloco contém uma única expressão:
 - SELECT-FROM-WHERE e cláusulas GROUP BY e HAVING
- Cada bloco é transformado em uma expressão da álgebra equivalente
- O objetivo será otimizar os blocos internamente, levando-se em consideração a ordem de execução entre eles

Tradução SQL – Álgebra relacional

Em SQL:

```
SELECT SOBRENOME, NOME FROM EMPREGADO  
WHERE SALARIO > (SELECT MAX (SALÁRIO)  
FROM EMPREGADO  
WHERE NUD=5);
```

Em blocos:

```
SELECT SOBRENOME, NOME FROM  
EMPREGADO  
WHERE SALARIO > c
```

```
SELECT MAX (SALÁRIO)  
FROM EMPREGADO  
WHERE NUD=5
```

Em Álgebra Relacional :

$$\Pi_{\text{SOBRENOME, NOME}} (\sigma_{\text{salário} > c} (\text{EMPREGADO}))$$
$$\mathcal{M}_{\text{MAX SALARIO}} (\sigma_{\text{NUD}=5} (\text{EMPREGADO}))$$

Executada apenas um vez
Conhecida como consulta
aninhada não correlacionada

Consulta aninhada correlacionada:

Variável tupla do bloco externo aparece na cláusula WHERE do bloco interno.

Ordenação Externa

- Cláusula **ORDER BY**
- A **ordenação externa** consiste em ordenar arquivos de tamanho maior que a memória principal.
- Os métodos de ordenação externa são diferentes dos de ordenação interna.
 - Os algoritmos devem diminuir o número de acesso aos blocos do disco.
 - O foco dos algoritmos para ordenação externa é reduzir o número de passadas sobre o arquivo
 - Uma boa medida de complexidade de um algoritmo de ordenação externa é o número de vezes que um bloco é lido ou escrito no disco.
- No disco, os dados ficam em um arquivo sequencial.

Ordenação por intercalação

- O método mais importante de ordenação externa é a **ordenação por intercalação**
- Intercalar significa combinar dois ou mais blocos ordenados em um único bloco ordenado.
- A intercalação é utilizada como uma operação auxiliar na ordenação.

Ordenação por intercalação

- Estratégia:
 1. Quebre o arquivo em blocos do tamanho da memória interna disponível.
 2. Ordene cada bloco na memória interna.
 3. Intercale os blocos ordenados, fazendo várias passadas sobre o arquivo.
 - A cada passada são criados arquivos ordenados cada vez maiores, até que contenha apenas um único arquivo ordenado

Exemplo

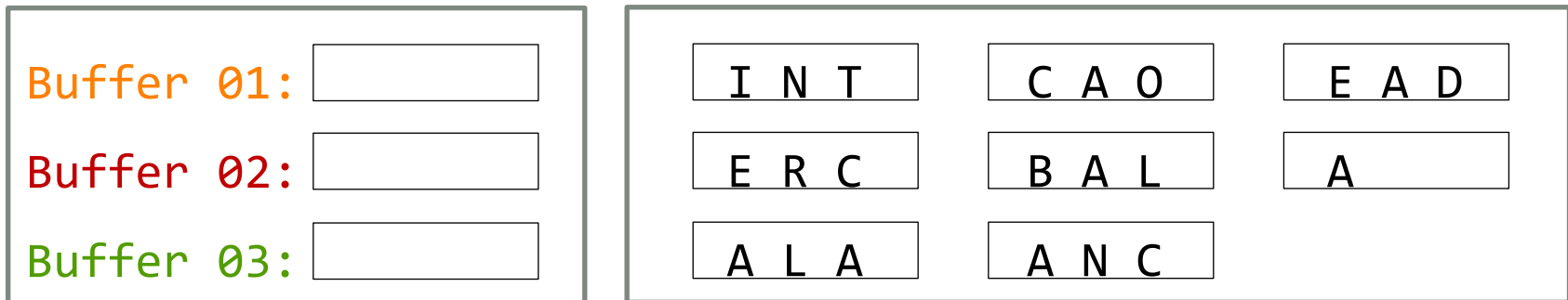
- Para exemplificação, ordene um arquivo que possui os seguintes 22 registros:

I N T E R C A L A C A O B A L A N C E A D A

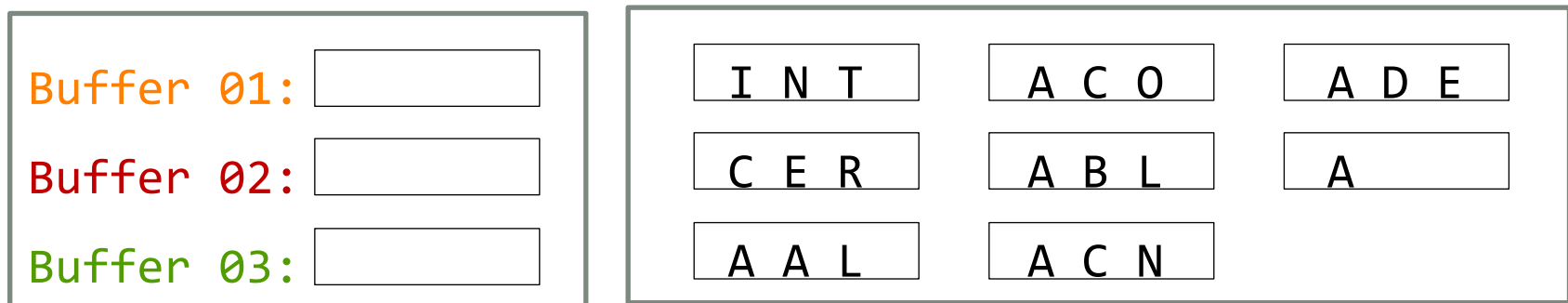
- Considere:
 - memória interna com capacidade para três itens
 - 3 buffers de 1 bloco disponíveis

Exemplo

- Fase de criação dos blocos ordenados, envolvendo:
 - quebra do arquivo em blocos do tamanho da memória principal (*buffer*):

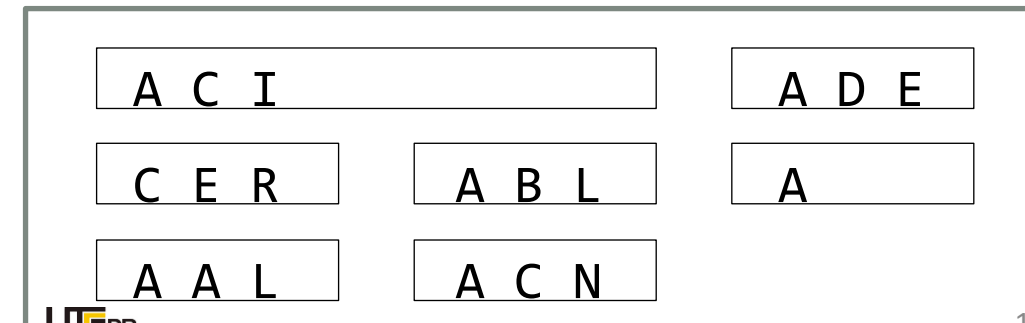
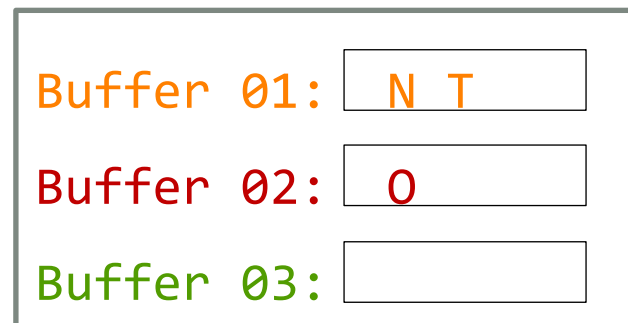
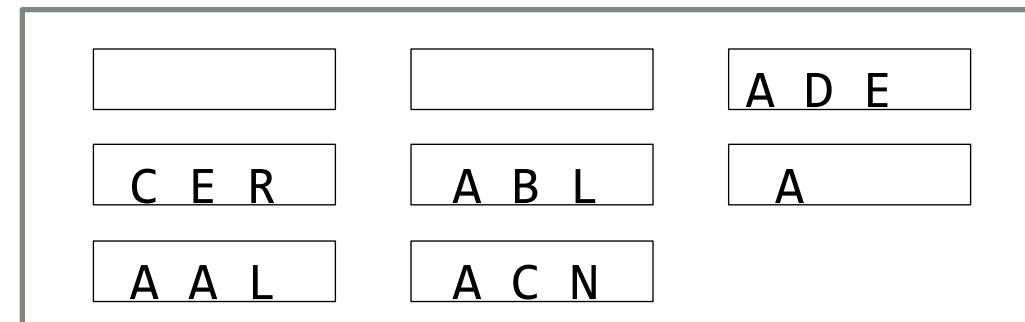
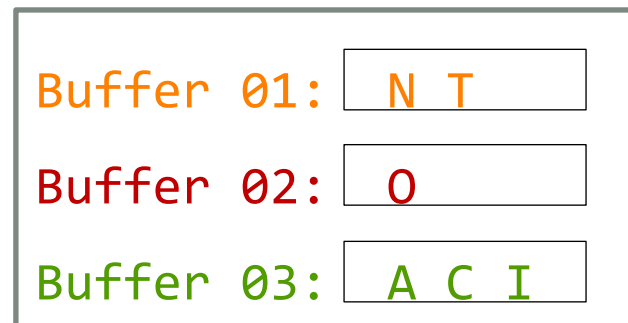
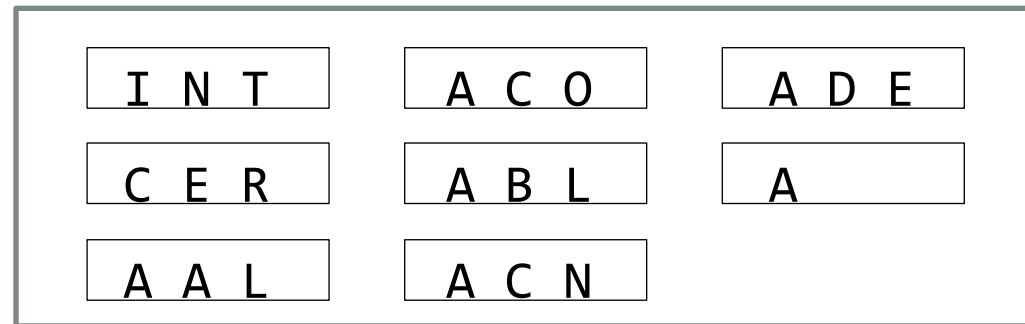
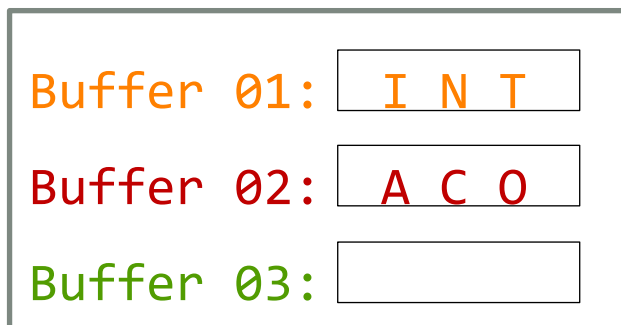


- ordenação de cada bloco na memória principal:



Exemplo

- Fase de intercalação:
 - Um bloco de buffer é usado para manter o resultado da intercalação bloco



Exemplo

Buffer 01:

Buffer 02:

Buffer 03:

A C I

A D E

C E R

A B L

A

A A L

A C N

Buffer 01:

Buffer 02:

Buffer 03:

A C I

N O T

A D E

C E R

A B L

A

A A L

A C N

A C I

N O T

A C D

E E R

A A B

L

A A A

C L N

Exemplo

A C I	N O T
A C D	E E R
A A B	L
A A A	C L N

A A C	C D E	E I N	O R T
A A A	A A B	C L L	N

A A A	A A A	A B C	C C D	E E I	L L N	N O R	T
-------	-------	-------	-------	-------	-------	-------	---

Análise da Ordenação Externa

- b = Número de Blocos
- FASE 1: Criação de Blocos
 - Complexidade = $2 * b$ (leitura e escrita)
 - Exemplo = $2 * 8 = 16$
- FASE 2: Intercalação:
 - Complexidade = $2 * b * (\log_2 b)$
 - Exemplo = $2 * 8 * (\log_2 8) = 48$
- FASE 1 + FASE 2
 - Complexidade: $2*b + 2*b*(\log_2 b)$
 - Exemplo: $16 + 48$
- Esse exemplo não leva em consideração que um buffer pode armazenar mais do um único bloco.
- Pense nisso e leia a Seção 19.2 do Livro do Navath

Algoritmos para Seleção

- Cláusula **WHERE**
- Há diferentes alternativas para localizar registros em um arquivo de disco que satisfazem uma determinada condição
- Para auxiliar nossa discussão, considere as seguintes sentenças:

OP1: $\sigma_{\text{Cpf} = '123456789'} (\text{FUNCIONARIO})$

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

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

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

OP5: $\sigma_{\text{Fcpf}='123456789' \text{ AND } \text{Pnr} = 10} (\text{TRABALHA_EM})$

Métodos de pesquisa para seleção simples

- [S1] - Pesquisa Linear

- Recupera cada registro no arquivo e testa se os valores dos atributos satisfazem a condição de seleção.
- Como os registros são agrupados em blocos de disco, cada um desses blocos é lido para um buffer da memória principal, e depois uma pesquisa pelos registros no bloco de disco é realizada na memória principal

- [S2] – Pesquisa Binária

- Pode ser usada se a condição de seleção envolver uma comparação de igualdade em um atributo chave no qual o arquivo é ordenado.
- Mais eficiente que a Pesquisa Linear [S1]
- Exemplo:

OP1: $\sigma_{\text{Cpf} = '123456789'}$ (FUNCIONARIO)

Métodos de pesquisa para seleção simples

- [S3a] – Usando um índice primário
 - Pode ser usado quando a condição de seleção envolver uma comparação de igualdade em um atributo chave com um índice primário.
- [S3b] – Usando uma chave hash
 - Pode ser usado se a condição de seleção envolver uma comparação de igualdade em um atributo chave com uma chave hash.
- Ambos [S3a] e [S3b] retornam apenas um único registro

Métodos de pesquisa para seleção simples

- [S4] Usando um índice primário para recuperar vários registros

- Se a condição de comparação for $>$, \geq , $<$ ou \leq em um campo chave com um índice primário, use o índice para encontrar o registro que satisfaz a condição de igualdade correspondente, depois recupere todos os registros subsequentes ou anteriores no arquivo (ordenado)
- Exemplo:

OP2: $\sigma_{D_{\text{numero}} > 5}$ (DEPARTAMENTO)

- [S5] Usando um índice de agrupamento para recuperar vários registros

- Pode ser usado se a condição de seleção envolver uma comparação de igualdade em um atributo não chave com um índice de agrupamento
- Exemplo:

OP3: $\sigma_{D_{\text{nr}} = 5}$ (FUNCIONARIO)

Métodos de pesquisa para seleção simples

- [S6] Usando um índice secundário (*B⁺-tree*) em uma comparação de igualdade.
 - Pode ser utilizado para recuperar um único registro se o campo de índice for uma **chave** (tiver valores únicos) ou para recuperar múltiplos registros se o campo de índice **não for uma chave**.
 - Também pode ser usado para comparações envolvendo >, >=, < e <=
- Resumindo:
 - S1 aplica-se a qualquer arquivo
 - S2 exige o arquivo seja ordenado pelo atributo da busca.
 - S3a, S4, S5 e S6 são conhecidos como **pesquisa de índice** e requerem um índice apropriado.
 - S4 e S6 podem ser usados para recuperar intervalos chamadas **consultas de intervalo**.

Métodos de pesquisa para seleção complexa

- Métodos usando quando a condição de seleção for uma **condição conjuntiva**
 - Várias condições simples compostas pelo conectivo lógico AND
 - Exemplo:

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

OP5: $\sigma_{Fcpf='123456789' \text{ AND } Pnr = 10}$ (TRABALHA_EM)

Métodos de pesquisa para seleção complexa

- [S7] Seleção conjuntiva usando um índice individual

- Se um atributo envolvido em qualquer condição simples isolada na condição de seleção conjuntiva tiver um caminho de acesso que permita o uso de um dos métodos S2 a S6, use essa condição para recuperar os registros e depois verificar se cada registro recuperado satisfaz as condições simples restantes na condição de seleção conjuntiva.
- Exemplo:

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

- [S8] Seleção conjuntiva usando um índice composto

- Usar um índice composto (ou estrutura de hash) se dois ou mais atributos que estiverem envolvidos nas condições de igualdade na condição seleção conjuntiva compõem a chave do índice.
- Exemplo:

OP5: $\sigma_{Fcpf='123456789' \text{ AND } Pnr = 10}$ (TRABALHA_EM)

Métodos de pesquisa para seleção complexa

- [S9] Seleção conjuntiva por intersecção de ponteiros
 - Se índices secundários (ou outros caminhos de acesso) estiverem disponíveis em mais de um dos campos envolvidos em condições simples na condição de seleção conjuntiva, e se os índices incluírem **os ponteiros dos registros** (em vez de ponteiros de blocos), então cada índice pode ser usado para recuperar o conjunto de ponteiros de registros que satisfaz a condição individual.
 - A **intersecção** desses conjuntos de ponteiros de registros gera os ponteiros de registros que satisfazem a condição de seleção conjuntiva, que então são usados para recuperar os registros diretamente.
- Resumindo:
 - A otimização de consulta é necessária para uma operação de seleção para **condições conjuntivas** uma vez que pode haver diferentes caminhos de acesso.
 - O otimizador deve escolher o caminho de acesso que **recupera menos registro** de maneira mais eficiente

Seletividade de uma condição

- A **seletividade (sl)** é porcentagem dos registros no arquivo a serem recuperados.
- É definida como a **razão** entre o **número de registros que satisfazem a condição** e o **número total de registros** no arquivo.
 - **SI = 0** – significa que nenhum dos registros no arquivo satisfaz a condição de seleção
 - **SI = 1** – significa que todos os registros no arquivo satisfazem a condição de seleção
- Seletividades exatas são difíceis de serem calculadas
- SGBD mantém um catalogo de informações para estimar **sl**

Condições de seleção disjuntivas


- Condição disjuntiva
 - Várias condições simples compostas pelo conectivo lógico OR
 - Muito mais complexas de serem otimizadas e processadas
 - Exemplo:

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

- Os registros que satisfazem a condição disjuntiva são a união os registros que satisfazem as condições individuais.
- Somente se houver um caminho de acesso a cada condição simples na disjunção é que uma otimização é possível.

Algoritmos para Junção

- Operações **EQUIJOINS** e **JUNÇÃO NATURAL**
- Operações de junção são as mais demoradas em uma consulta.
- Exemplos:

OP6: FUNCIONARIO  _{Dnr=Dnumero} DEPARTAMENTO

OP7: DEPARTAMENTO  _{cpf_ger=cpf} FUNCIONARIO

Métodos para implementar Junções

- [J1] Junção de loop aninhado

- Esse algoritmo de força bruta

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

- Para cada registro t em R (loop externo), recupere cada registro s em S (loop interno) e teste se a igualdade ($t[A] = s[B]$) é válida

- [J2] Junção de único loop (usando uma estrutura de acesso para recuperar os registros correspondentes)

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

- Se houver um índice (ou chave *hash*) para um dos dois atributos de junção
- Suponha o que o atributo B de S tenha um índice.
- Recupere cada registro t de R (loop no arquivo R) e depois use o índice para recuperar diretamente todos os registros correspondentes s em S que satisfazem $t[A] = s[B]$
- Exemplo:

OP6: FUNCIONARIO $\bowtie_{Dnr=Dnumero}$ DEPARTAMENTO

Métodos para implementar Junções

- [J3] – Junção Ordenação-Concatenação
 - Se os registros de R e S estiverem **fisicamente ordenados** pelos valores dos atributos A e B, uma função mais eficiente pode ser implementada.
 - Os dois arquivos são varridos simultaneamente combinando os registros que têm os mesmos valores.
 - Uma variação desse algoritmo pode utilizar um índice secundário
 - Arvore B⁺
- [J4] – Junção de partição-hash
 - Os registros de R e S são particionados em arquivos menores.
 - Fase do particionamento:
 - O particionamento de cada arquivo 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 o particionamento do arquivo S)
 - Fase de investigação:
 - Os registro de cada partição de R são comparados com os registros da partição correspondente em S.

Algoritmos para operações de Projeção

- Cláusula **SELECT** e **DISTINCT** $\pi_{\langle \text{lista de atributos} \rangle}(R)$
- Uma projeção é simples de ser implementada se o atributo chave de R estiver na lista de atributos da projeção.
 - Todos as tuplas são mantidas
 - Nenhum esforço adicional é necessário
- Caso contrário
 - As tuplas repetidas devem ser eliminadas.
 - Uso do **SELECT DISTINCT**
 - Ordenar o resultado e remover os registros duplicados
 - Também é possível usar uma função de hash e se o novo registro for direcionado para o bucket ocupado, o registro é duplicado.

Algoritmos para operações de Conjunto

- Operações UNIÃO, INTERSECÇÃO, DIFERENÇA DE CONJUNTO e PRODUTO CARTESIANO
- O produto cartesiano é a operação **mais custosa**.
 - O resultado inclui todos os registros das combinações de R e S.
 - Cada registro inclui todos os atributos.

Algoritmos para operações de Conjunto

- UNIÃO, INTERSECÇÃO, DIFERENÇA DE CONJUNTO
 - requerem relações de **tipo compatível**
 - Mesmo número de atributos com o mesmo domínio.
 - Usa variações da técnica ordenação-intercalação
 - Duas relações são ordenadas pelo mesmo atributo
 - Em seguida, uma simples varredura produz o resultado esperado.
 - Exemplo:
 - Em $R \cup S$, ao varrer e intercalar dois arquivos ordenados (R e S), podemos facilmente manter todos os registros e eliminar os duplicados.
 - Usa uma função de hash
 - O primeiro arquivo é varrido e particionado por uma função de hash
 - O segundo arquivo é varrido e a função de hash é usada para verificar se os registros estão ou não contidos nas partições.
 - Exercício:
 - Como implementar as operações de INTERSECÇÃO e DIFERENÇA DE CONJUNTO usando as duas técnicas?
 - Qual técnica é mais custosa?

Algoritmos para operações de agregação

- Operadores MIN, MAX, COUNT, AVERAGE e SUM

- Exemplo:

```
SELECT MAX(salario) FROM FUNCIONARIO;
```

- Estratégias:

- Varredura em todo arquivo
- Uso de índices (se houver)
 - A folha mais a direita de uma árvore B⁺-Tree é o elemento máximo.

- Exemplo:

```
SELECT Dno, AVG(salario) FROM FUNCIONARIO  
GROUP BY Dno;
```

- Estratégias:

- Ordenação
- Hashing
- Uso de índice de agrupamento
 - Calcular a média em cada grupo.

Algoritmos para junção externa

- Operações **LEFT JOIN**, **RIGHT JOIN** e **FULL JOIN**

- Exemplo:

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

- Implementado por variações das técnicas:
 - [J1] junção loop aninhado
 - [J2] junção de único loop

Algoritmos para junção externa

- Teoricamente, a junção externa pode ser calculada para executar uma combinação de operadores da álgebra relacional.

- Exemplo:

1. Calcule a junção interna das tabelas FUNCIONARIO e DEPARTAMENTO

$$\text{TEMP1} \leftarrow \pi_{\text{Unome}, \text{Pnome}, \text{Dnome}} (\text{FUNCIONARIO} \bowtie_{\text{Dnr}=\text{Dnumero}} \text{DEPARTAMENTO})$$

2. Ache as tuplas de FUNCIONARIO que não aparecem no resultado da junção interna

$$\text{TEMP2} \leftarrow \pi_{\text{Unome}, \text{Pnome}} (\text{FUNCIONARIO}) - \pi_{\text{Unome}, \text{Pnome}} (\text{TEMP1})$$

3. Preencha cada tupla em TEMP2 com um campo Dnome NULL

$$\text{TEMP2} \leftarrow \text{TEMP2} \times \text{NULL}$$

4. Aplique a operação de UNION em TEMP1 e TEMP2 obtendo a JUNÇÃO EXTERNA A ESQUERDA.

$$\text{RESULT} \leftarrow \text{TEMP1} \cup \text{TEMP2}$$

Notação para árvores de consulta

- Uma **árvore de consulta** é uma estrutura de dados de árvore que corresponde à uma expressão de álgebra relacional.
- Os **vértices folhas** representam as relações de entrada da consulta
- Os **vértices internos** representam as operações da álgebra relacional.
- Uma execução da árvore de consulta consiste na execução de uma operação de vértice interno assim que seus operandos estão disponíveis e depois na substituição desse vértice interno pela relação que resulta da execução da operação.
- Ordem de execução
 - Começa nos vértices folhas
 - Termina no vértice raiz.

Notação para árvores de consulta

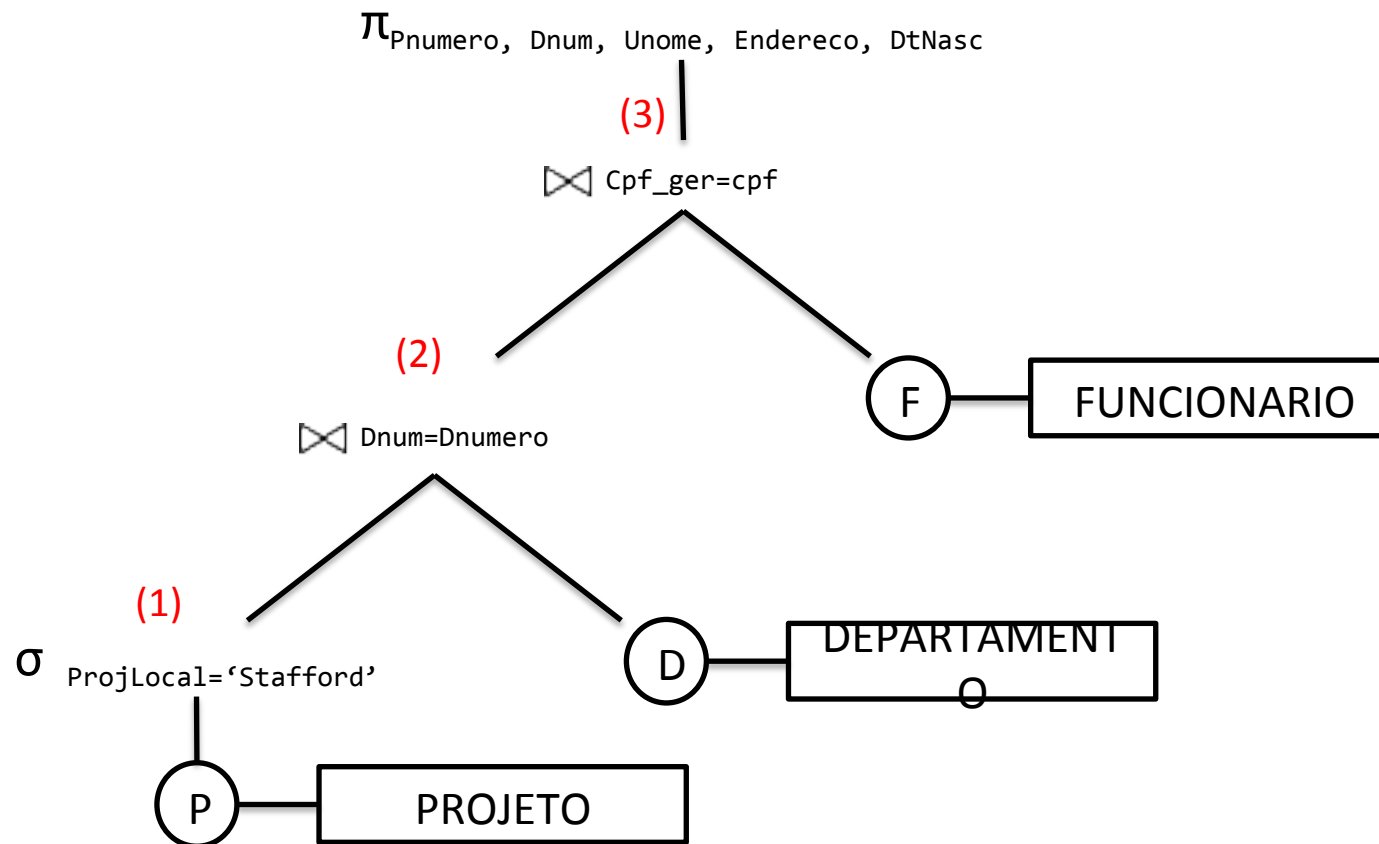
- Para cada projeto localizado em Stafford, recupere o número do projeto, o número do departamento, o sobrenome, endereço e data de nascimento do gerente do departamento

```
SELECT P.Pnumero, P.Dnum, F.Unome, F.Endereco, F.DtNasc
FROM PROJETO AS P, DEPARTAMENTO AS D, FUNCIONARIO AS F
WHERE P.Dnum=D.Dnumero
      AND D.cpf_ger=F.cpf
      AND P.ProjLocal= 'Stafford';
```

$\pi_{\text{Pnumero, Dnum, Unome, Endereco, DtNasc}} ($
 $((\sigma_{\text{ProjLocal='Stafford'}}(\text{PROJETO})) \bowtie_{\text{Dnum=Dnumero}} (\text{DEPARTAMENTO}))$
 $\bowtie_{\text{cpf_ger=cpf}} (\text{FUNCIONARIO})$
 $)$

Notação para árvores de consulta

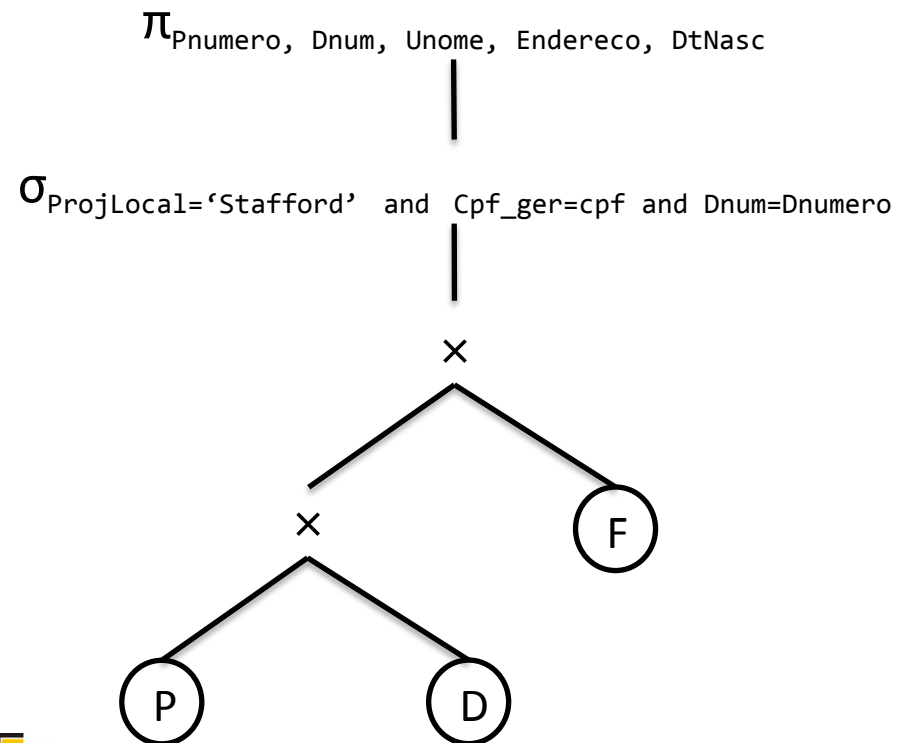
- A árvore de consulta representa uma ordem de operações específica para a execução da consulta



Otimização heurística para árvores de consultas

- Diferentes arvores que geram o mesmo resultado são ditas ser equivalentes.
- O analisador de consultas gera uma árvore de consulta inicial a partir do SQL, sem realizar qualquer otimização.
- Veja o exemplo anterior:

```
SELECT P.Pnumero, P.Dnum, F.Unome,  
       F.Endereco, F.DtNasc  
FROM PROJETO AS P,  
     DEPARTAMENTO AS D,  
     FUNCIONARIO AS F  
WHERE P.Dnum=D.Dnumero  
      AND D.cpf_ger=F.cpf  
      AND P.ProjLocal= 'Stafford';
```



Exemplo de transformação de uma consulta

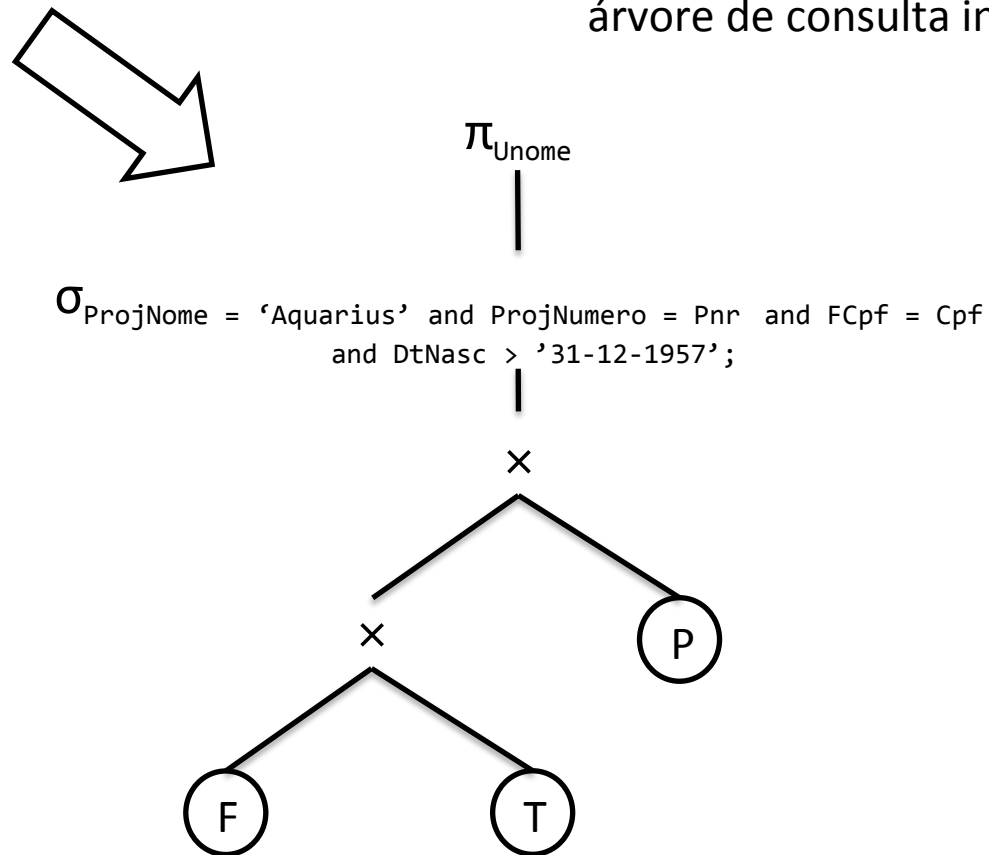
- Exemplo: Ache os sobrenomes dos funcionários nascidos após 1957 que trabalham em um projeto chamado *Aquarius*

```
SELECT Unome
FROM FUNCIONARIO,TRABALHA_EM, PROJETO
WHERE ProjNome = 'Aquarius'
      AND ProjNumero = Pnr
      AND FCpf = Cpf
      AND DtNasc > '31-12-1957';
```

Exemplo de transformação de uma consulta

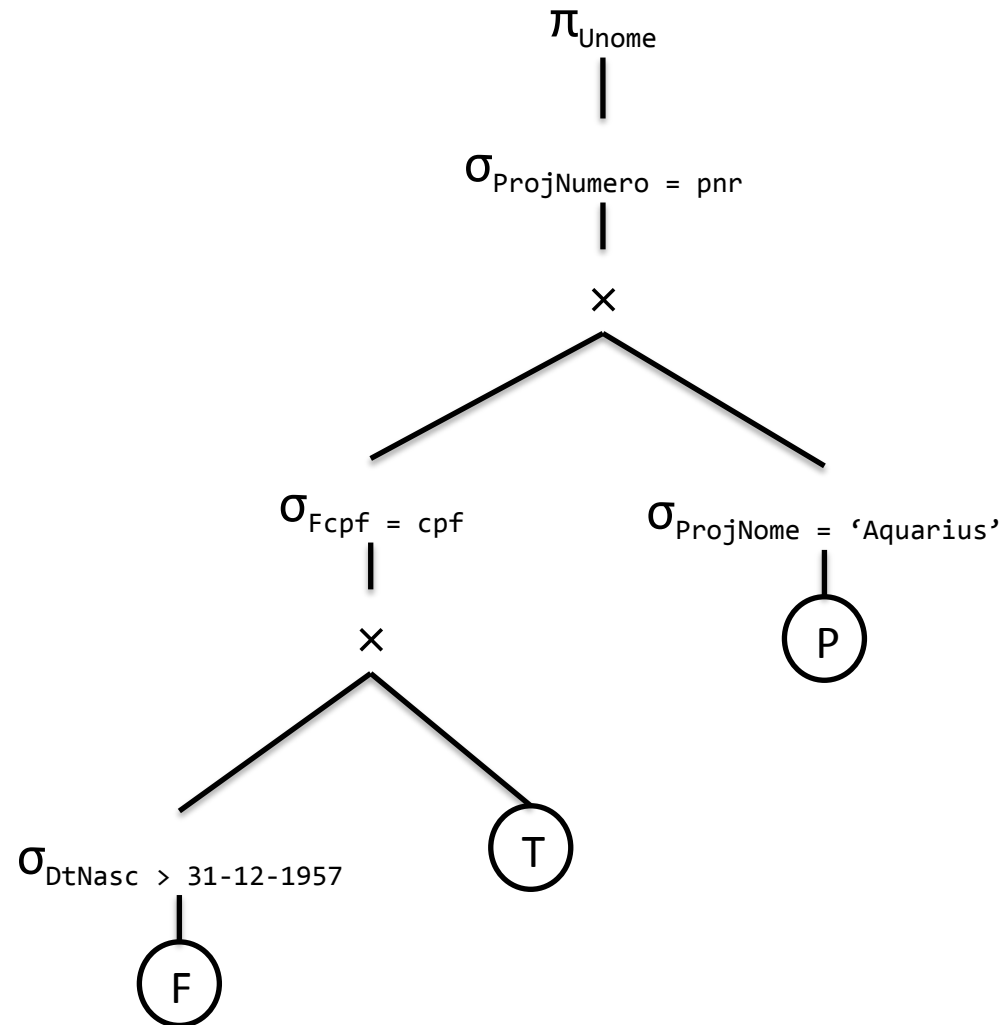
```
SELECT Unome
FROM FUNCIONARIO, TRABALHA_EM, PROJETO
WHERE ProjNome = 'Aquarius'
  AND ProjNumero = Pnr
  AND FCpf = Cpf
  AND DtNasc > '31-12-1957';
```

Conversão direta em uma
árvore de consulta inicial



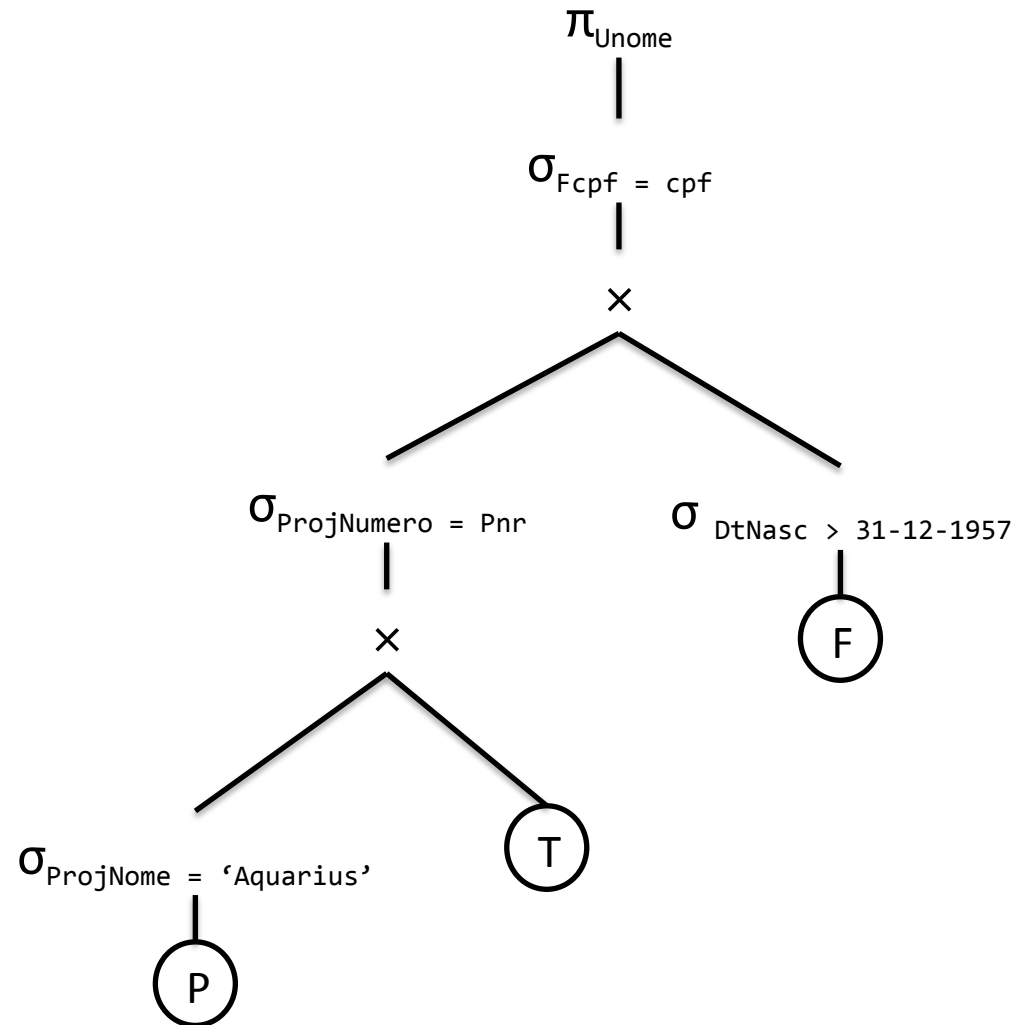
Exemplo de transformação de uma consulta

Otimização
1º. Passo



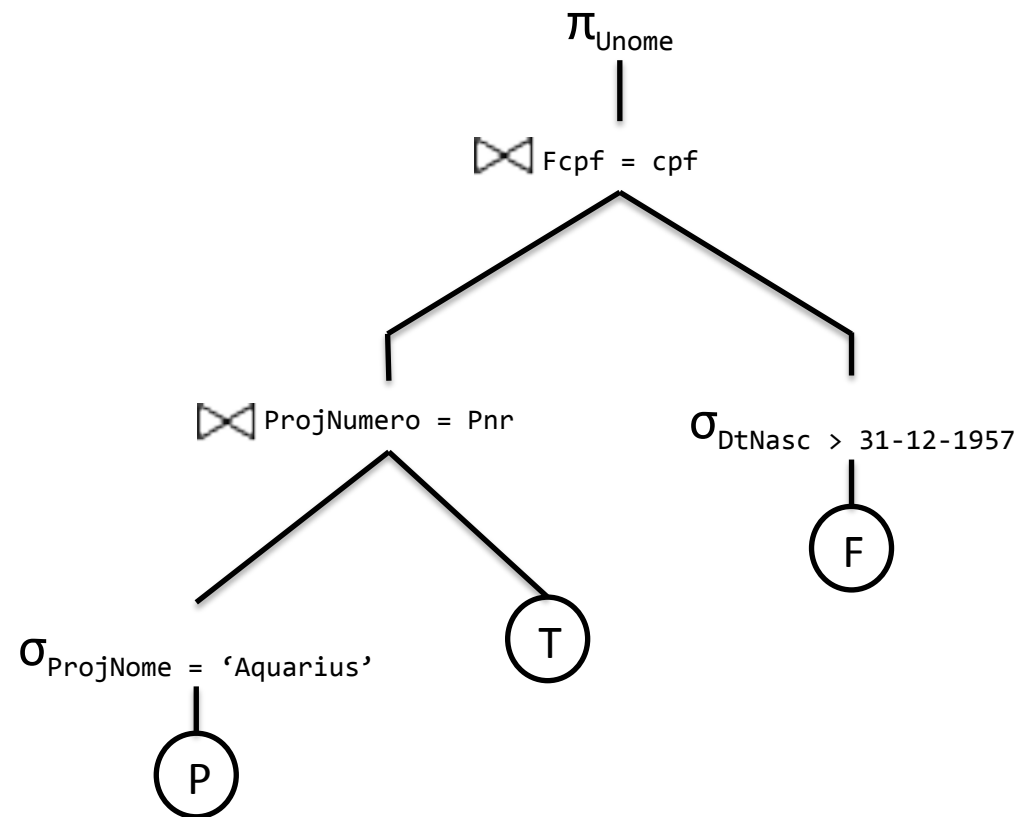
Exemplo de transformação de uma consulta

Otimização
2º. Passo



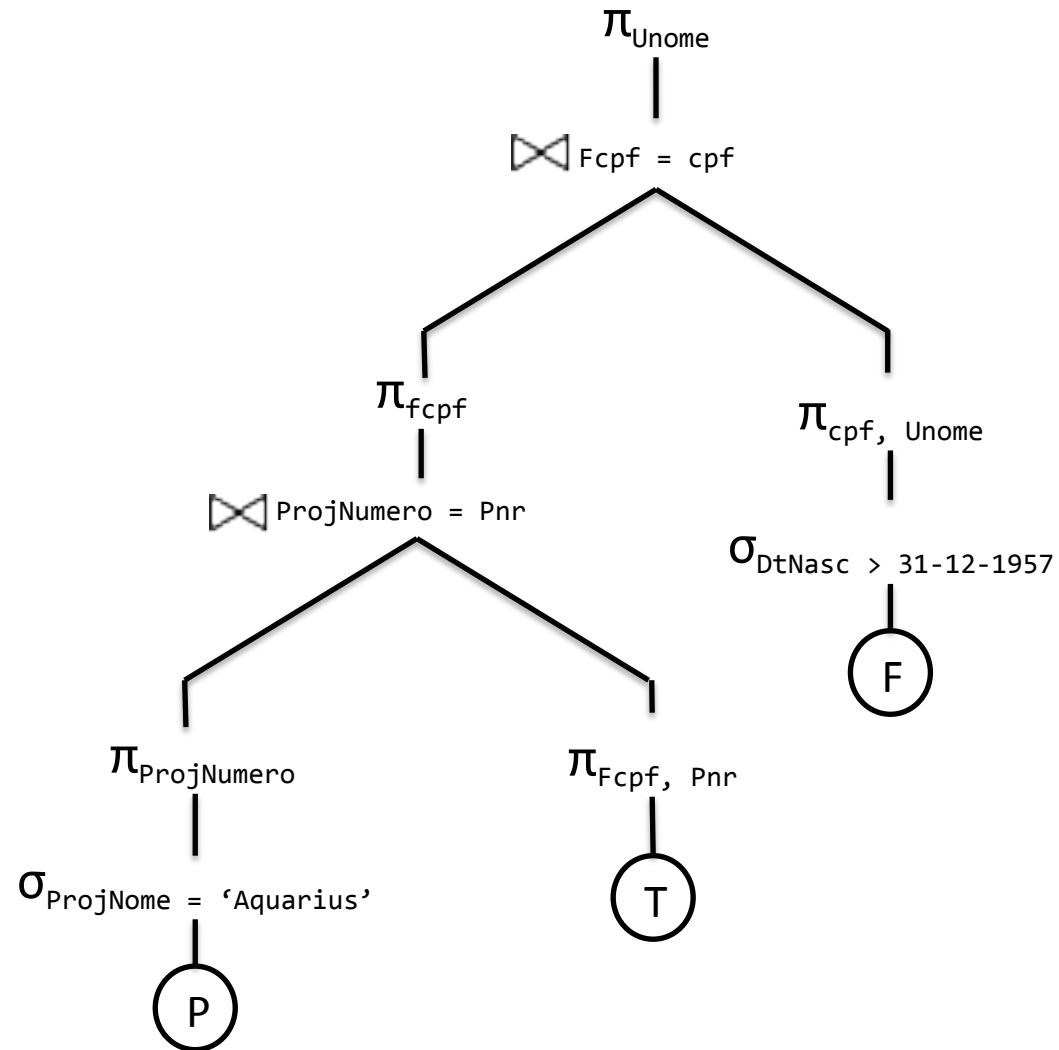
Exemplo de transformação de uma consulta

Otimização
3º. Passo



Exemplo de transformação de uma consulta

Otimização
4º. Passo



Regras de Transformação Geral

1. Cascata de σ

- Uma operação de seleção conjuntiva pode ser desmembrada em uma cascata (ou seja, uma sequência) de operações σ individuais

$$\sigma_{c1 \text{ and } c2 \text{ and } \dots \text{ and } cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\dots(\sigma_{cn}(R)\dots)))$$

2. Comutatividade de σ

- A operação σ é comutativa.

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

Regras de Transformação Geral

3. Cascata π

- Em uma cascata (sequência) de operações π , todas podem ser ignoradas menos a última.

$$\pi_{\text{lista1}}(\pi_{\text{lista2}}(\dots(\pi_{\text{listan}}(R))\dots)) \equiv \pi_{\text{lista1}}(R)$$

4. Comutação de σ com π

- Se a condição de seleção c envolve apenas os atributos A_1, \dots, A_n na lista de projeção, as duas operações podem ser comutadas.

$$\pi_{A_1, A_2, \dots, A_n}(\sigma_c(R)) \equiv \sigma_c(\pi_{A_1, A_2, \dots, A_n}(R))$$

Regras de Transformação Geral

5. Comutatividade de \bowtie (e \times)

- As operações de junção e de produto cartesiano são comutativas

$$\begin{aligned}(R \bowtie_c S) &\equiv (S \bowtie_c R) \\ (R \times S) &\equiv (S \times R)\end{aligned}$$

6. Comutação de σ com \bowtie (ou \times)

- Se todos os atributos na condição de seleção c envolvem apenas os atributos de uma das relações que estão sendo juntadas, as duas operações podem ser comutadas.

$$\sigma_c (R \bowtie S) \equiv (\sigma_c (R)) \bowtie S$$

- Se uma condição c puder ser escrita como $(C1 \text{ AND } C2)$, onde a condição $C1$ envolve os atributos de R e $C2$ envolve os atributos de S , então as operação são comutáveis.

$$\sigma_c (R \bowtie S) \equiv (\sigma_{c_1}(R)) \bowtie (\sigma_{c_2}(S))$$

Regras de Transformação Geral

7. Comutação de π com \bowtie (ou \times)

- Suponha a lista de projeção seja $L = \{ A_1, \dots, A_n, B_1, \dots, B_n \}$
 - $\{A_1, \dots, A_n\}$ são atributos de R
 - $\{B_1, \dots, B_n\}$ são atributos de S
- Se a condição de junção de c envolver apenas atributos de L , as duas operações podem ser comutadas.

$$\pi_L (R \bowtie_c S) \equiv (\pi_{A_1, \dots, A_n} (R)) \bowtie_c (\pi_{B_1, \dots, B_n} (S))$$

8. Comutatividade das operações de conjunto

- As operações de conjunto UNIÃO e INTERSECÇÃO são comutáveis mas, DIFERENÇA não é.

Regras de Transformação Geral

9. Associatividade de \bowtie , \times , \cup e \cap

- Essas quatro operações são associativas individualmente; ou seja, se Θ indicar qualquer uma dessas quatro operações (por toda a expressão), temos:

$$(R \Theta S) \Theta T \equiv R \Theta (S \Theta T)$$

10. Comutação de σ com operações de conjunto

- A operação σ comuta com \cup , \cap e -
- se Θ indicar qualquer uma dessas três operações (por toda a expressão), temos:

$$\sigma_c(R \Theta S) \equiv (\sigma_c(R)) \Theta (\sigma_c(S))$$

Regras de Transformação Geral

11. A operação π comuta com \cup

$$\pi_L (R \cup S) \equiv (\pi_L(R)) \cup (\pi_L(S))$$

12. Convertendo uma sequencia (σ, \times) em \bowtie

- Se a condição c de um σ que segue um \times corresponde a uma condição de junção, converta a sequencia por um (σ, \times) em um \bowtie

$$\sigma_c(R \times S) \equiv (R \bowtie_c S)$$

Regras de Transformação Geral

- Outras transformações possíveis:

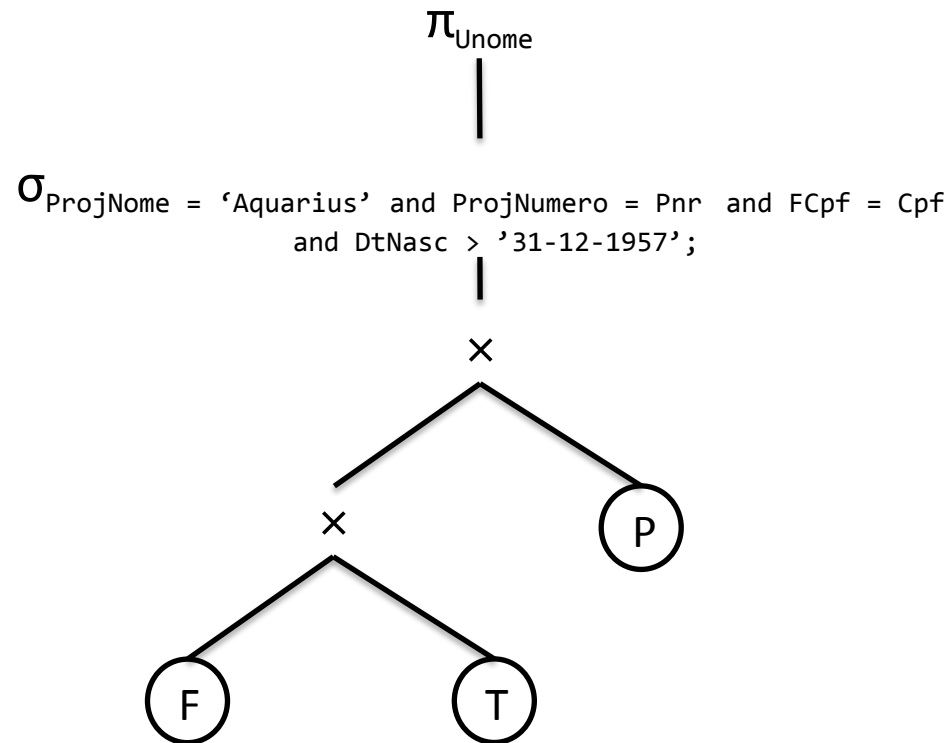
$$\text{NOT } (c_1 \text{ AND } c_2) \equiv (\text{NOT } c_1) \text{ OR } (\text{NOT } c_2)$$

$$\text{NOT } (c_1 \text{ OR } c_2) \equiv (\text{NOT } c_1) \text{ AND } (\text{NOT } c_2)$$

(DeMorgan's laws)

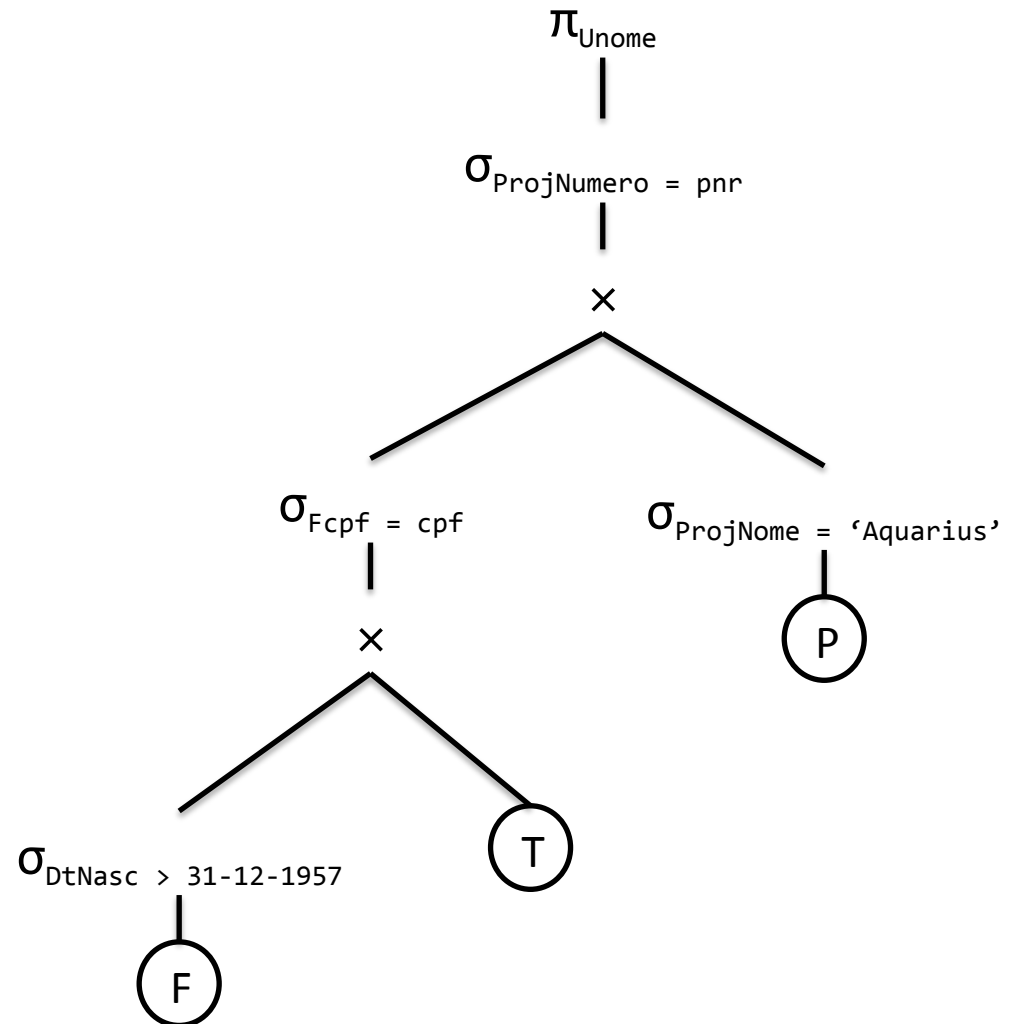
Algoritmo de Otimização Algébrica Heurística

- Usaremos o exemplo anterior para executar o algoritmo:



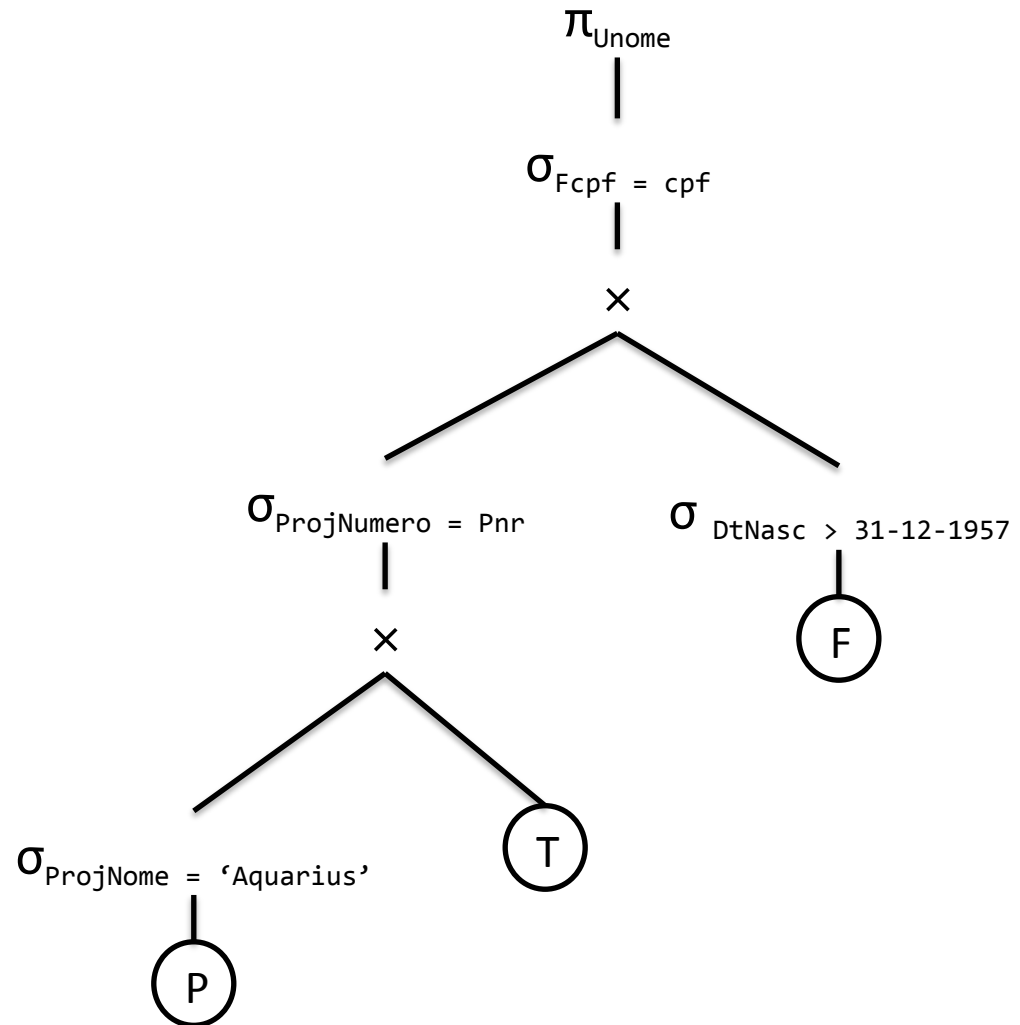
Algoritmo de Otimização Algébrica Heurística

1. Usando a regra 1, quebre quaisquer operações de SELEÇÃO com condições conjuntivas em uma cascata de operações SELEÇÃO
 - Isso permite uma maior liberdade na movimentação para baixo de operações de SELEÇÃO por diferentes ramos da árvore
2. Usando as regras 2, 4, 6 e 10 referentes à comutatividade de SELEÇÃO, mova cada operador de seleção o mais para baixo possível na árvore.



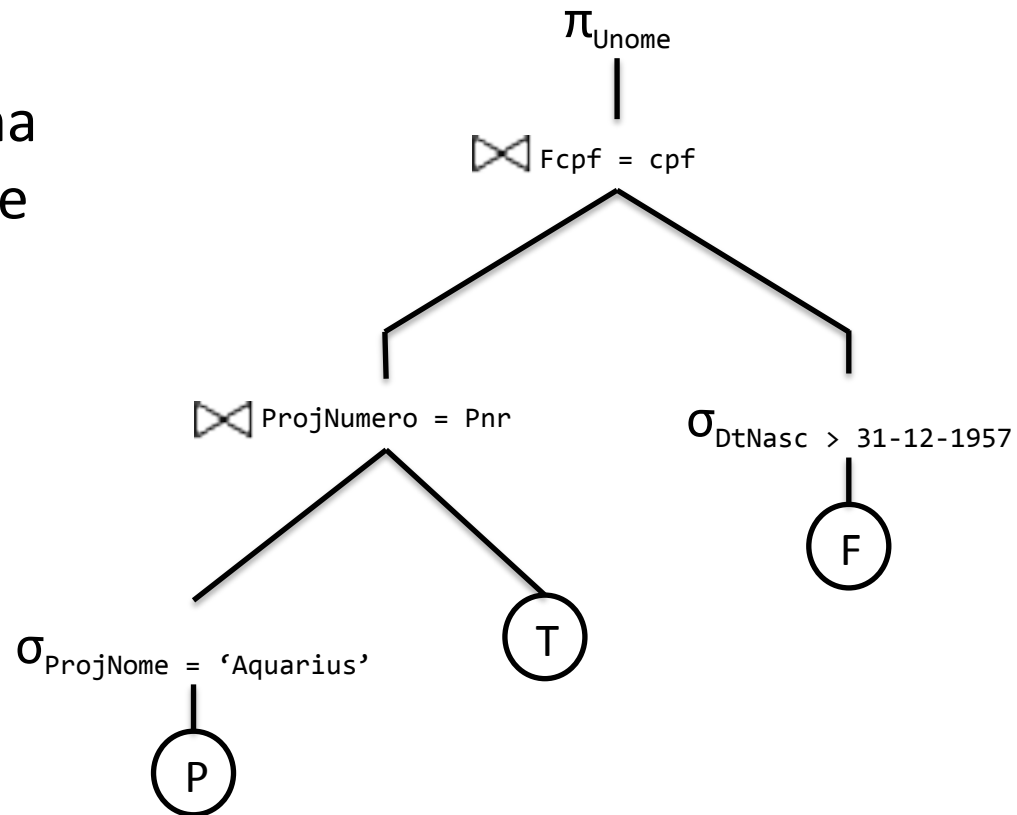
Algoritmo de Otimização Algébrica Heurística

3. Usando as regras 5 e 9 referentes à comutatividade e associatividade de operações binárias, reorganize os vértices folhas de tal forma:
1. Posicione as relações do vértice folha com as operações SELEÇÃO mais restritivas, de modo que sejam executadas primeiro.
 2. Garanta que a ordenação dos vértices folhas não cause uma operação de PRODUTO CARTESIANO



Algoritmo de Otimização Algébrica Heurística

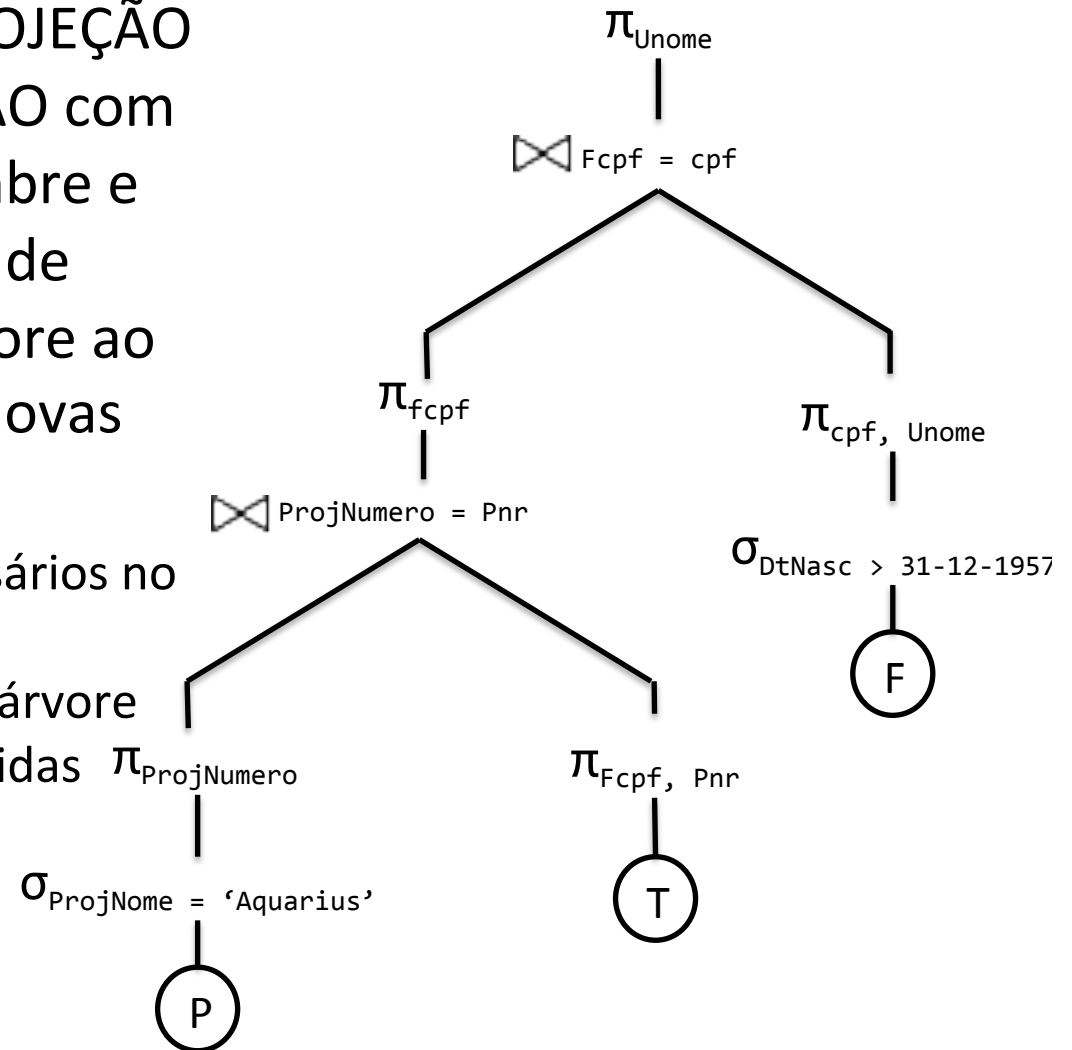
4. Usando a regra 12, combine uma operação PRODUTO CARTESIANO com uma SELEÇÃO subsequente na árvore para uma operação de JUNÇÃO



Algoritmo de Otimização Algébrica Heurística

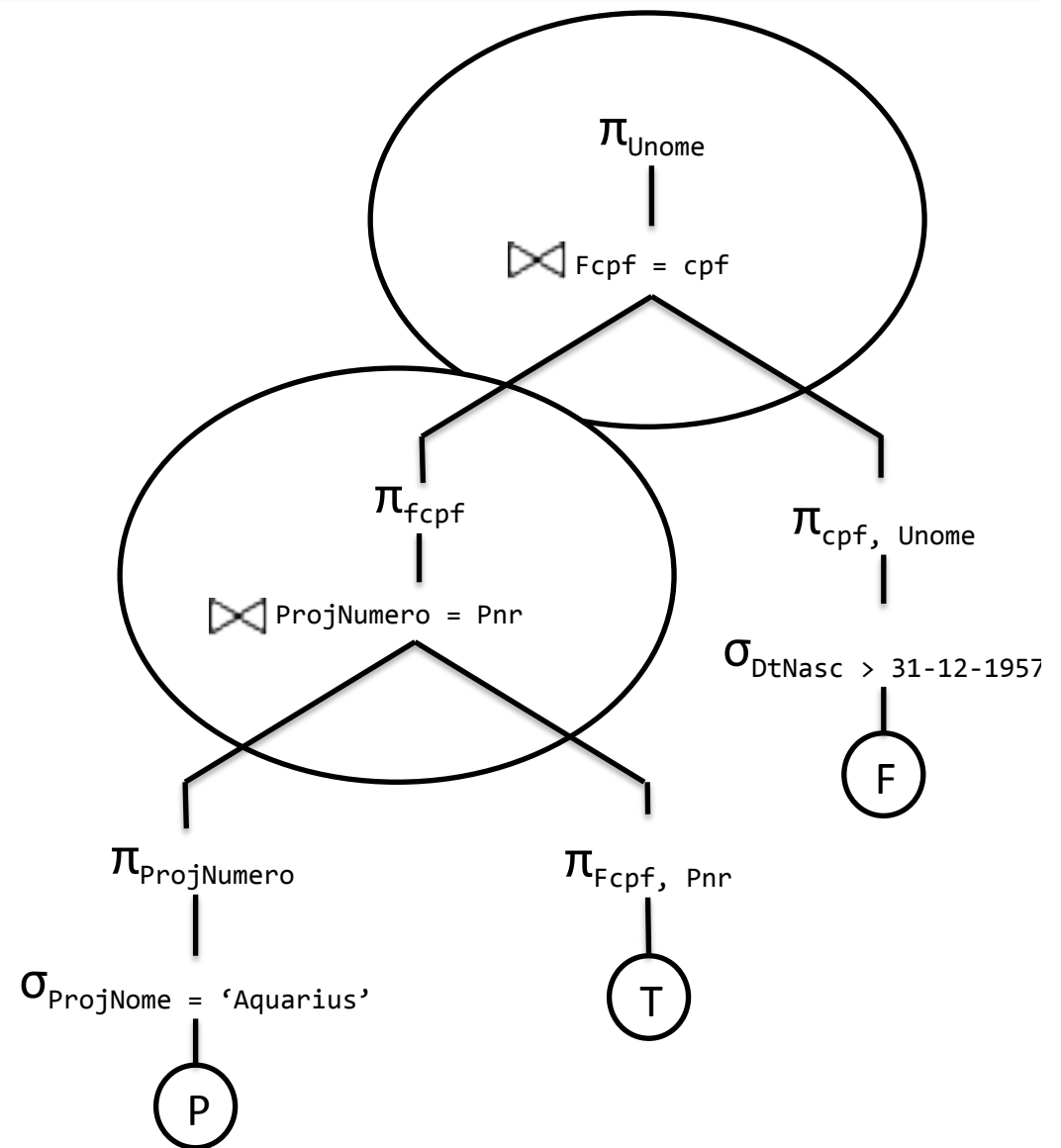
5. Usando as regras 3, 4, 7 e 11 referentes à cascata de PROJEÇÃO e a comutação de PROJEÇÃO com outras operações, desmembre e mova as listas de atributos de projeção para baixo na árvore ao máximo possível, criando novas operações PROJEÇÃO.

- Somente os atributos necessários no resultado da consulta e nas operações subsequentes na árvore de consulta devem ser mantidas após cada projeção.



Algoritmo de Otimização Algébrica Heurística

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



Dúvidas

