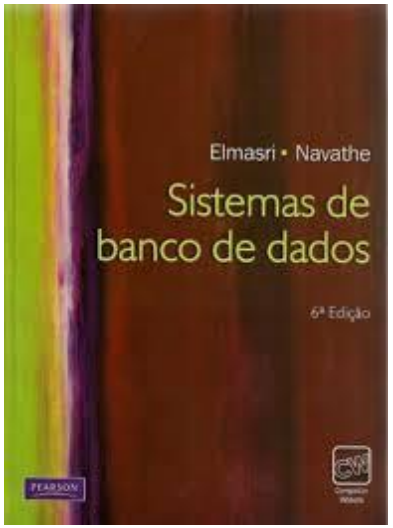
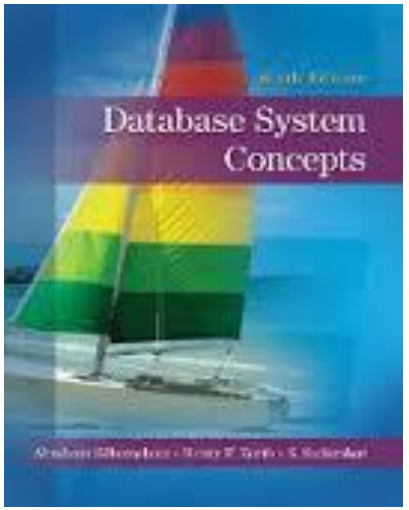
Unidade 14 – Algoritmos para Otimização de Consultas em Banco de Dados

Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

Bibliografia



Sistemas de Banco de Dados
Elmasri / Navathe 6ª edição

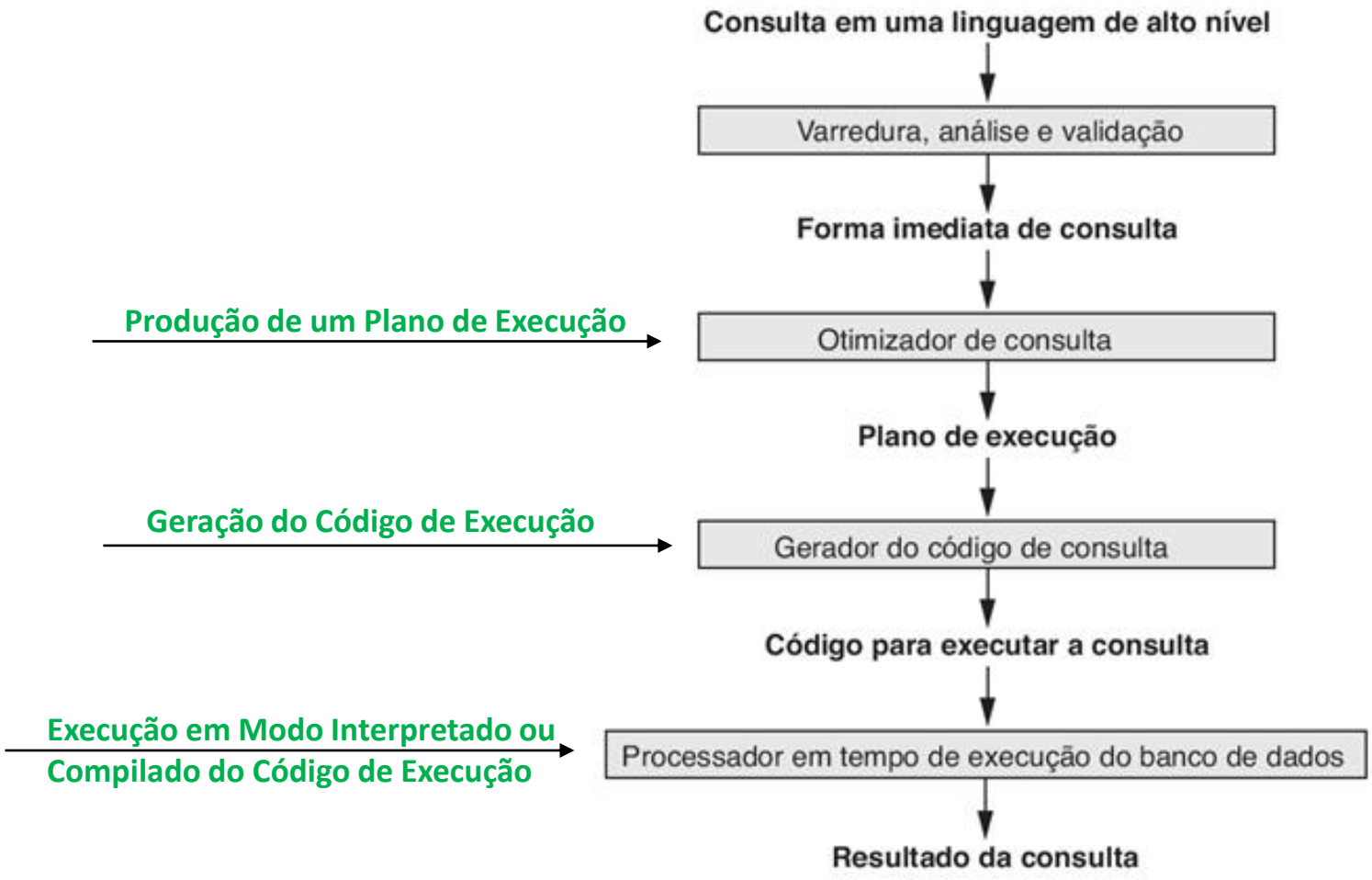


Sistema de Banco de Dados
Korth, Silberschatz – Sixth Editon

Processamento de Consultas

- Uma consulta no SGBD, expressa em **SQL**, precisa ser primeiramente lida, analisada e validada;
- A varredura consiste em identificar os tokens de consulta (**Análise Léxica**);
- A análise verifica a sintaxe da consulta para verificar se ela está formulada de acordo com as regras gramaticais da linguagem de consulta (**Análise Sintática**);
- A validação consiste em se verificar se os nomes de atributos e relações são válidos e semanticamente significativos no esquema do banco de dados.

Etapas do Processador de Consultas



Fonte: Elmasri,Navathe

Otimização de Consultas

- O SGBD precisa idealizar **uma estratégia de execução** (ou **plano de execução**) para recuperar os resultados da query com base nos arquivos de banco de dados;

- Uma query, em geral, costuma ter muitas estratégias de execução possíveis e o processo de escolha de uma dessas estratégias adequadas para a execução é conhecida por **Otimização de Consulta**.

- O termo **otimização**, na verdade, um nome errado, pois em alguns casos o plano de execução escolhido não é a estratégia **ótima**. As vezes, encontrar a estratégia ideal é muito demorado.

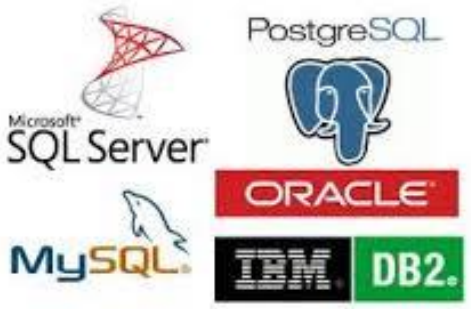
Por que encontrar o plano de execução ideal pode ser demorado ?

Plano de Execução

- A obtenção do plano ideal pode exigir informações detalhadas sobre como os arquivos do banco de dados são implementados e até mesmo sobre o seu conteúdo;
- Essas informações podem **não** estar disponíveis no catálogo do SGBD;
- Assim, o **planejamento de uma boa estratégia de execução** pode **não** ser, na verdade, o plano ótimo de execução.

Tradução de Consultas para a Álgebra Relacional

- Na prática, a linguagem **SQL** é usada na maioria dos SGBDR comerciais;
- Uma query **SQL** é inicialmente traduzida para uma expressão equivalente da Álgebra Relacional (representada por uma árvore de consulta) que é, então, **otimizada**.



Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

- Considere o seguinte esquema de banco de dados:

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

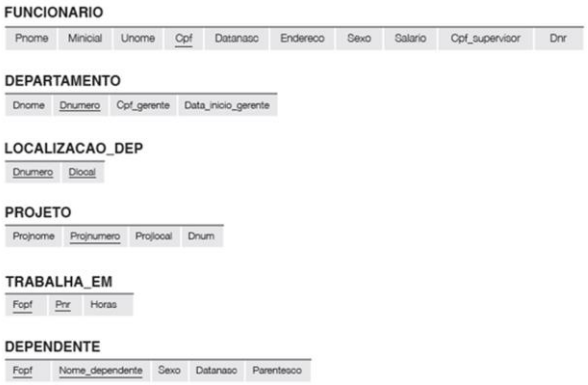
Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

- Considere a seguinte consulta SQL no banco de dados

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



Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

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

- Essa consulta recupera os nomes dos funcionários (**de qualquer departamento da empresa**) que ganham um salário maior que o **maior salário do Departamento 5**.

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

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

Bloco mais interno

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

- O bloco mais **interno** recupera o salário mais alto do departamento 5;

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

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

Bloco mais externo

```
SELECT   Unome, Pnome
FROM     FUNCIONARIO
WHERE    Salario > c
```

- No bloco mais **externo**, **c** representa o resultado retornado do bloco interno.

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

Bloco mais interno

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

- O bloco mais **interno** poderia ser traduzido para a seguinte expressão da Álgebra Relacional:

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

Notação de Função Agregada da Álgebra Relacional

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

Bloco mais externo

```
SELECT   Unome, Pnome
FROM     FUNCIONARIO
WHERE    Salario > c
```

- O bloco mais **externo** poderia ser traduzido para a seguinte expressão da Álgebra Relacional:

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

Fonte: Navathe/Elmasri

Exemplo – Consulta em Álgebra Relacional

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

- O **Otimizador de Consulta** irá escolher um Plano de Execução para cada bloco de consulta;

Bloco mais interno

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

Bloco mais externo

```
SELECT  Unome, Pnome
FROM    FUNCIONARIO
WHERE   Salario > c
```

Fonte: Navathe/Elmasri

Consultas Declarativas

- "**O quê**" ao invés de "**Como**";
- Paradigma **Declarativo**;
- Solução razoavelmente eficiente (Elmasri, 2011);
- Solução ótima pode ser custosa (Elmasri, 2011);
- Consulta **SQL** é decomposta em **blocos simples**;
  - ✓ Contém uma única expressão **SELECT-FROM-WHERE** (**GROUP BY** e **HAVING** se houver);
  - ✓ Consultas aninhadas são tratadas como consultas independentes;

Consultas SQL decomposta em BLOCOS

- Tabela

```
Pessoa(Codigo, Nome, Telefone, AnoFiliacao)
```

- Nome dos filiados mais antigos:

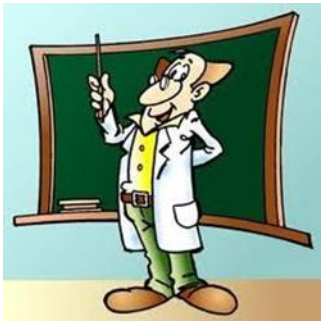
```
SELECT Codigo, Nome
FROM PESSOA
WHERE AnoFiliacao = (SELECT MIN(AnoFiliacao))
                     FROM PESSOA)
```

- Blocos

❶
```
SELECT Codigo, Nome
FROM PESSOA
WHERE AnoFiliacao = (referência ❷)
```

❷
```
SELECT MIN(AnoFiliacao))
FROM PESSOA
```

Algoritmos para as Operações Relacionais



Algoritmos para Seleção

- Existem vários algoritmos para executar uma operação de **SELEÇÃO**, que é basicamente uma operação de consulta para localizar registros;

- Utilizaremos o seguinte esquema de banco de dados para o estudo dos algoritmos de **SELEÇÃO**.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Algoritmos para Seleção

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

- Considere-se as seguintes operações de **SELEÇÃO:**

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

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

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

OP4: $\sigma_{\text{Dnr = 5 AND Salario > 30.000 AND Sexo = 'F'}}$ (FUNCIONARIO)

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

Métodos de Pesquisa para Seleção

- Diversos algoritmos de pesquisa são possíveis para se selecionar registros de um arquivo;
- Estes são conhecidos como **VARREDURAS de ARQUIVO** porque varrem os registros de um arquivo para procurar e recuperar registros que satisfazem a uma condição de seleção.
- Se o algoritmo de pesquisa envolve o uso de um **ÍNDICE**, a pesquisa do índice é denominada **VARREDURA do ÍNDICE**.

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

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

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

OP4: $\sigma_{\text{Dnr = 5 AND Salario > 30.000 AND Sexo = 'F'}}(\text{FUNCIONARIO})$

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

Métodos de Pesquisa para Seleção

- Pesquisa LINEAR;
- Pesquisa BINÁRIA;
- Uso de índice Primário;
- Uso de chave HASH;
- Uso de índice Secundário;
- Uso de Índice Primário para recuperar vários registros;
- Uso de índice B-tree;

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

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

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

OP4: $\sigma_{\text{Dnr = 5 AND Salario > 30.000 AND Sexo = 'F'}}$ (FUNCIONARIO)

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

Métodos de Pesquisa para Seleção

- Um Sistema Gerenciador de Banco de Dados terá à sua disposição muitos dos métodos vistos anteriormente;
- O otimizador de consultas precisa escolher o **método mais apropriado** para executar cada operação de SELEÇÃO em uma consulta;
- Essa otimização usa fórmulas que estimam os custos para cada método de acesso disponível;
- O **otimizador** escolhe o método de acesso com o **menor custo estimado**.

Implementação de operações de JUNÇÃO

- A operação de JUNÇÃO é uma das operações **mais demoradas** no processamento de consultas;
- Muitas das operações de junção encontradas nas consultas são das variedades **EQUIJUNÇÃO** e **JUNÇÃO NATURAL**.

Operação JUNÇÃO

- ✓ Na Álgebra Relacional é Indicada por ⋈ .
- ✓ Permite processar relacionamentos (tuplas relacionadas) entre duas relações.
- ✓ O resultado da **JUNÇÃO** é uma relação Q com n + m atributos Q (A_1, A_2, ..., A_n, B_1, B_2, ... B_m) nessa ordem, que tem uma tupla para cada combinação de tuplas - uma de R e outra de S – sempre que a combinação satisfaz a condição de junção.
- ✓ Essa é a principal diferença entre **PRODUTO CARTESIANO** e **JUNÇÃO**.
- ✓ Em JUNÇÃO apenas combinações de tuplas que satisfazem a condição de junção aparecem no resultado, enquanto que no PRODUTO CARTESIANO todas as combinações de tuplas são incluídas no resultado.

Exemplo JUNÇÃO ⋈

Join

- ✓ Necessita-se recuperar o nome do Gerente de cada Departamento;

DEPARTAMENTO

PK idDepto	NomeDepto	FK idGerente
10	COMPRAS	1000
20	ENGENHARIA	1200
40	VENDAS	2000
55	FINANCEIRO	5000

FUNCIONARIO

PK idFunc	NomeFunc
1000	Paulo de Souza Alves
1100	José da Silva
1200	Pedro Rangel de Souza
1450	Angela Silva Medeiros
2000	Alceu de Almeida
3000	Saulo de Araujo
3400	Pedro Silva
5000	Roberto Boschetti
5400	Ana de Souza Almeida

Exemplo JUNÇÃO ⋈

Join

- ✓ Aplicando-se a operação de JUNÇÃO com a condição idGerente = idFunc

DEPARTAMENTO

idDepto	NomeDepto	idGerente

FUNCIONARIO

idFunc	NomeFunc

A ← DEPARTAMENTO ⋈ idGerente = idFunc FUNCIONARIO

A

idDepto	NomeDepto	idGerente	idFunc	NomeFunc
10	COMPRAS	1000	1000	Paulo de Souza Alves
20	ENGENHARIA	1200	1200	Pedro Rangel de Souza
40	VENDAS	2000	2000	Alceu de Almeida
55	FINANCEIRO	5000	5000	Roberto Boschetti

Exemplo JUNÇÃO ⋈

- ✓ Aplicando-se a operação de **PROJEÇÃO** na relação intermediária

A

idDepto	NomeDepto	idGerente	idFunc	NomeFunc

RESULTADO ← π NomeDepto, NomeFunc (A)

RESULTADO

NomeDepto	NomeFunc
COMPRAS	Paulo de Souza Alves
ENGENHARIA	Pedro Rangel de Souza
VENDAS	Alceu de Almeida
FINANCEIRO	Roberto Boschetti

Exemplo JUNÇÃO ⋈

✓ Renomeando-se os atributos da relação RESULTADO

RESULTADO

NomeDepto	NomeFunc

ρ (NomeDepto, NomeGerente) (RESULTADO)

RESULTADO

NomeDepto	NomeGerente
COMPRAS	Paulo de Souza Alves
ENGENHARIA	Pedro Rangel de Souza
VENDAS	Alceu de Almeida
FINANCEIRO	Roberto Boschetti

EQUIJOIN

- ✓ O uso mais comum de JUNÇÃO envolve condições de junção em apenas comparações de IGUALDADE.
- ✓ Esse tipo de JUNÇÃO, em que se usa somente o operador de comparação =, é chamado EQUIJUNÇÃO ou EQUIJOIN.
- ✓ Na EQUIJUNÇÃO sempre se tem um ou mais pares de atributos com VALORES IDÊNTICOS.

EQUIJOIN - Exemplo

A ← DEPARTAMENTO ⋈ idGerente = idFunc FUNCIONARIO

A

idDepto	NomeDepto	idGerente	idFunc	NomeFunc
10	COMPRAS	1000	1000	Paulo de Souza Alves
20	ENGENHARIA	1200	1200	Pedro Rangel de Souza
40	VENDAS	2000	2000	Alceu de Almeida
55	FINANCEIRO	5000	5000	Roberto Boschetti

JUNÇÃO NATURAL

- ✓ Corresponde a uma EQUIJUNÇÃO (condição de junção de igualdade) no qual elimina-se o segundo atributo (desnecessário) uma vez que possuem valores idênticos.
- ✓ A definição padrão de JUNÇÃO NATURAL requer que os dois atributos de junção tenham o MESMO NOME nas duas relações. Se isto não ocorrer, deve-se aplicar uma operação de renomeação antes da operação de junção.
- ✓ A JUNÇÃO NATURAL é indicada por um *.

JUNÇÃO NATURAL - Exemplo

- ✓ Suponha que se queira combinar cada tupla de **PROJETO** com uma tupla de **DEPARTAMENTO** que controla um projeto.

DEPARTAMENTO

PK

idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO
90	RH

PROJETO

PK idProj	NomeProj	FK DeptoResponsavel
1000	Controle de Pedidos	10
1200	Projeto ABX	20
2000	Projeto Vendas Otimizadas	40
3000	Saulo de Araujo	55

- ✓ Nesse exemplo, as tuplas a serem combinadas devem ser relacionadas pelos atributos idDepto na relação de DEPARTAMENTO e DeptoResponsavel na relação PROJETO.

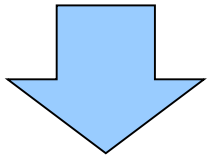
Join

JUNÇÃO NATURAL - Exemplo

✓Renomeia-se o nome de um dos atributos, para deixá-los com mesmo nome.

idProj	NomeProj	DeptoResponsavel

ρ (idProj, NomeProj, idDepto) (PROJETO)



PK **PROJETO** FK

idProj	NomeProj	idDepto
1000	Controle de Pedidos	10
1200	Projeto ABX	20
2000	Projeto Vendas Otimizadas	40
3000	Saulo de Araujo	55

Join

JUNÇÃO NATURAL - Exemplo

- ✓ As relações agora ficam com o atributo de join com mesmo nome (idDepto) nas relações de **DEPARTAMENTO** e **PROJETO**.

PK **DEPARTAMENTO**

idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO
90	RH

PK **PROJETO** **FK**

idProj	NomeProj	idDepto
1000	Controle de Pedidos	10
1200	Projeto ABX	20
2000	Projeto Vendas Otimizadas	40
3000	Saulo de Araujo	55

Join

JUNÇÃO NATURAL - Exemplo

- ✓ Aplica-se o JOIN NATURAL, com o atributo de junção idDepto.
- ✓ Somente um valor de atributo de junção será mantido na relação resultando.

PK DEPARTAMENTO

idDepto	NomeDepto
10	COMPRAS
20	ENGENHARIA
40	VENDAS
55	FINANCEIRO
90	RH

PK PROJETO FK

idProj	NomeProj	idDepto
1000	Controle de Pedidos	10
1200	Projeto ABX	20
2000	Projeto Vendas Otimizadas	40
3000	Saulo de Araujo	55

RESULTADO ← DEPARTAMENTO * PROJETO

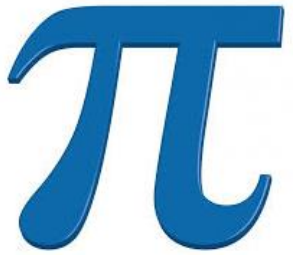
idDepto	NomeDepto	idProj	NomeProj
10	COMPRAS	1000	Controle de Pedidos
20	ENGENHARIA	1200	Projeto ABX
40	VENDAS	2000	Projeto Vendas Otimizadas
55	FINANCEIRO	3000	Saulo de Araujo

Algoritmos para operação de Junção

- ✓ Junção de Loop Aninhado – Sem caminhos de acesso (Força Bruta);
- ✓ Junção de Único Loop – Quando há um índice para um dos atributos de junção;
- ✓ Junção ordenação-intercalação – Quando os registros de R e S estiverem ordenados fisicamente (maneira mais eficiente).

Observação: Detalhamento dos Algoritmos em **Navathe,2011**

Algoritmos para operação de PROJEÇÃO



π <lista_de_atributos> (<R>)

- ✓ Uma operação de Projeção é simples de ser implementada se <lista_de_atributos> incluir a chave da relação R, pois nesse caso o resultado da operação terá o mesmo número de tuplas que R, mas com apenas os valores para os atributos em <lista_de_atributos> .
- ✓ Se <lista_de_atributos> não incluir a chave de R, as tuplas duplicadas devem ser eliminadas. Isso pode ser feito ordenando-se o resultado da operação e em seguida eliminam-se as tuplas duplicadas.

Observação: Detalhamento dos Algoritmos em Navathe,2011

Heurísticas para Otimização de Consultas

- ✓ Para se melhorar o desempenho de uma consulta, o SGBD utiliza **heurísticas** para modificar a representação interna da consulta – que normalmente está na forma de uma árvore de consulta;
- ✓ Assim, a representação inicial da consulta é então otimizada de acordo com regras heurísticas, levando a uma representação de consulta otimizada, que corresponde à estratégia de execução da consulta;

O que são heurísticas ?

Para entendermos o que é heurística, veremos antes o conceito de Corretude de Algoritmos

Algoritmos devem produzir saídas corretas.

Corretude

- ✓ Algoritmos corretos usualmente são acompanhados por uma **prova formal**, o qual explica o porquê o algoritmo gera saídas corretas para todas as instâncias do problema;
- ✓ Sem a prova matemática, as vezes, podemos nos enganar a respeito da **corretude** de um algoritmo;
- ✓ A **intuição**, muitas vezes, pode nos levar a resultados errôneos.
- ✓ Algoritmos corretos **não** são sempre óbvios para a maioria dos problemas de **otimização**.

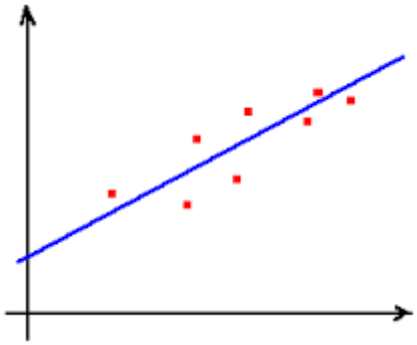
Corretude

- ✓ Um algoritmo é dito **correto** se, para cada instância de entrada ele pára com a saída correta.
- ✓ Dizemos que um **algoritmo correto** resolve o problema computacional dado.
- ✓ Um algoritmo **incorreto** pode não parar em algumas instâncias de entrada, ou então pode parar com outra resposta que não a desejada.

Algoritmos incorretos devem ser descartados ?

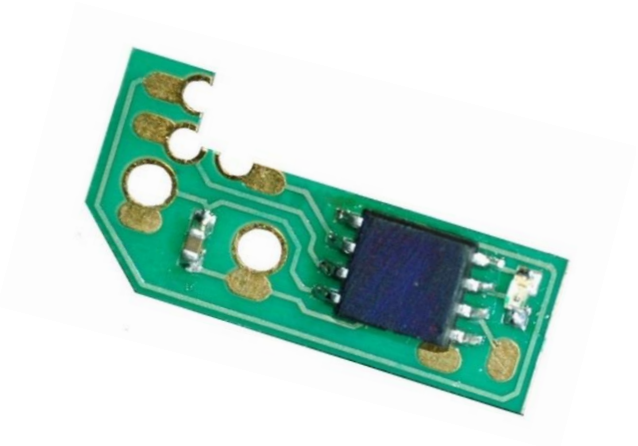
Algoritmos incorretos

- ✓ Ao contrário do que se poderia esperar, às vezes os algoritmos incorretos podem ser úteis, se sua taxa de erros puder ser controlada.
- ✓ **Exemplo**: algoritmos para se localizar grandes números **primos**.



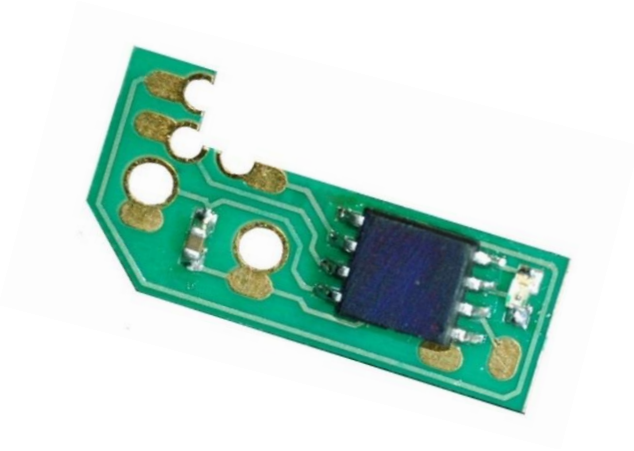
Corretude – Ilustração

- ✓ Suponha um robô equipado com uma ferramenta de solda para soldar pontos de um placa de circuito.
- ✓ O robô deve executar o trabalho de solda em determinados pontos de contato.
- ✓ O robô recebe uma quantidade de pontos de contato, devendo visitar o primeiro, o segundo, etc. .. até finalizar o trabalho...



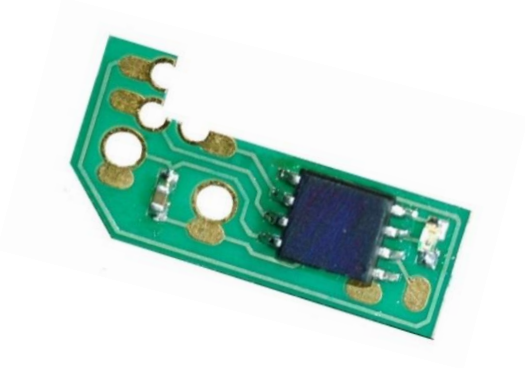
Corretude – Ilustração

- ✓ Robôs são **caros** !
- ✓ Assim, queremos **minimizar** o tempo que o braço do robô leva para soldar os pontos de contato...
- ✓ Assumimos que o braço do robô se move com velocidade constante.
- ✓ Assim, o tempo para processar a placa é proporcional à distância percorrida pelo braço do robô...

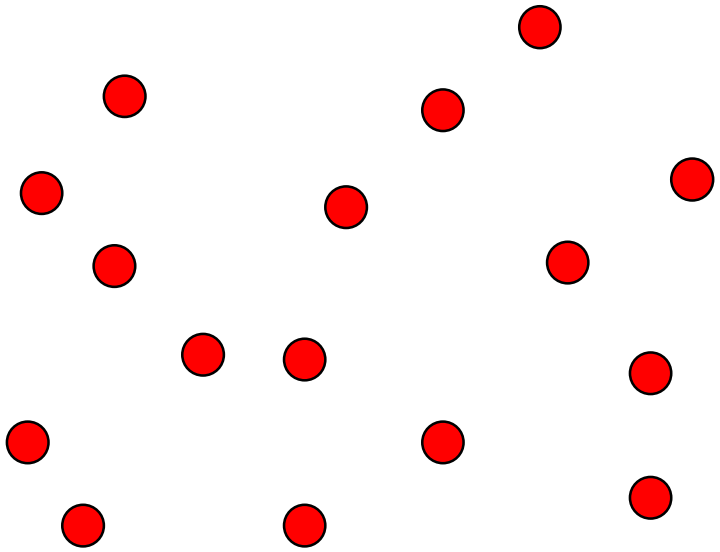


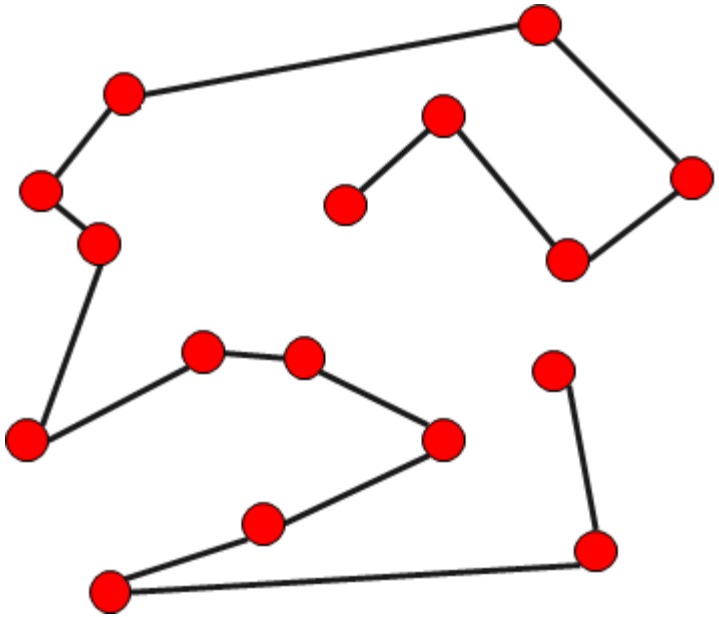
Qual é o problema ?

- **INPUT**: Um conjunto **S** de pontos num plano.
- **OUTPUT**: O caminho mais curto para visitar todos os pontos do ciclo.



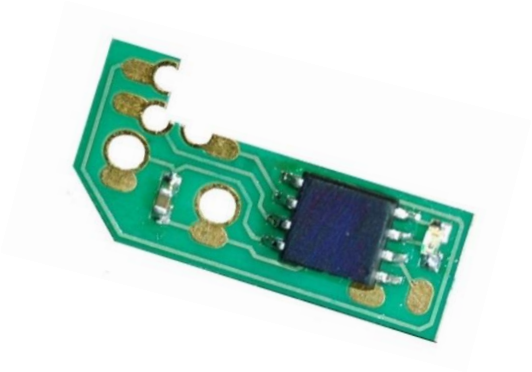
Encontrar o ciclo mais curto para visitar todos os pontos de solda





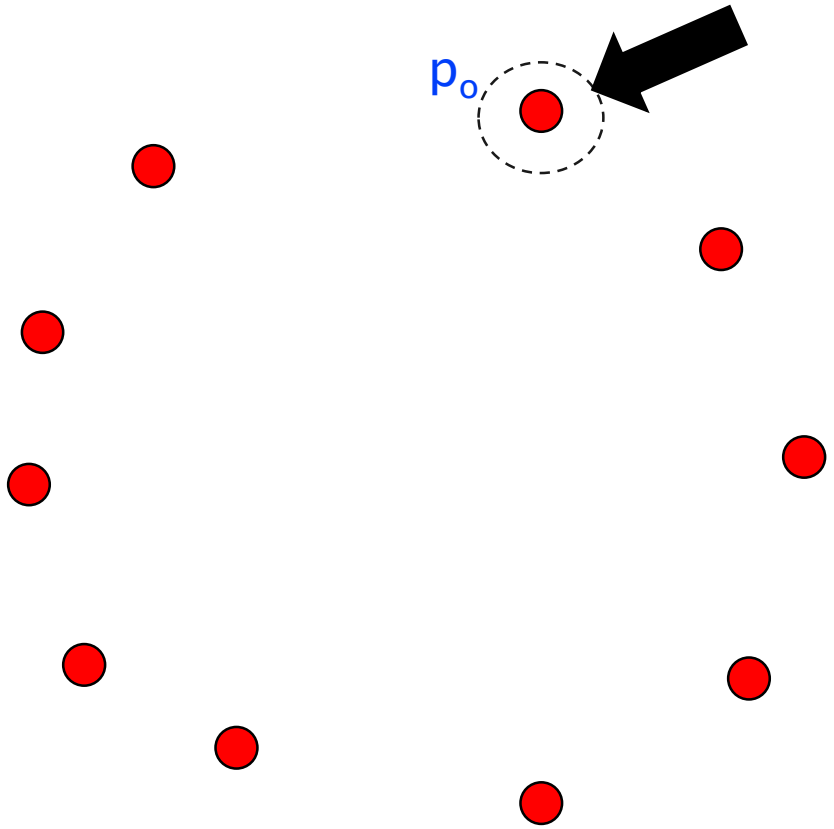
Você foi contratado para programar o braço do Robot ...

Pense um pouco em algum algoritmo para programar o braço do Robot ...



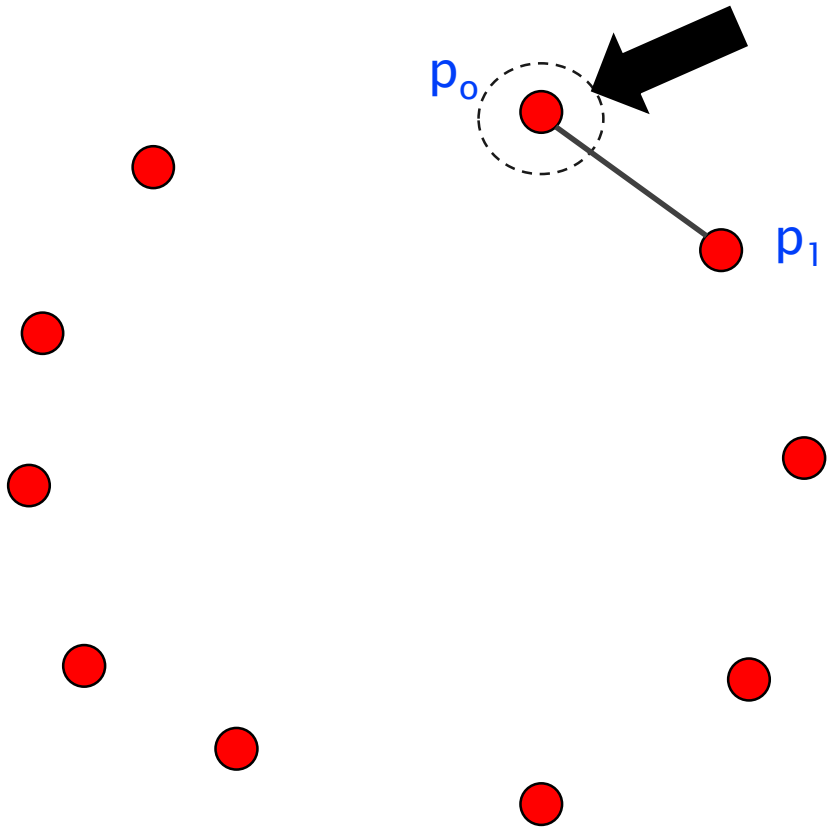
Heurística Nearest-Neighbor

- Inicie em algum ponto p_o e prossiga até o seu vizinho mais próximo p_1.



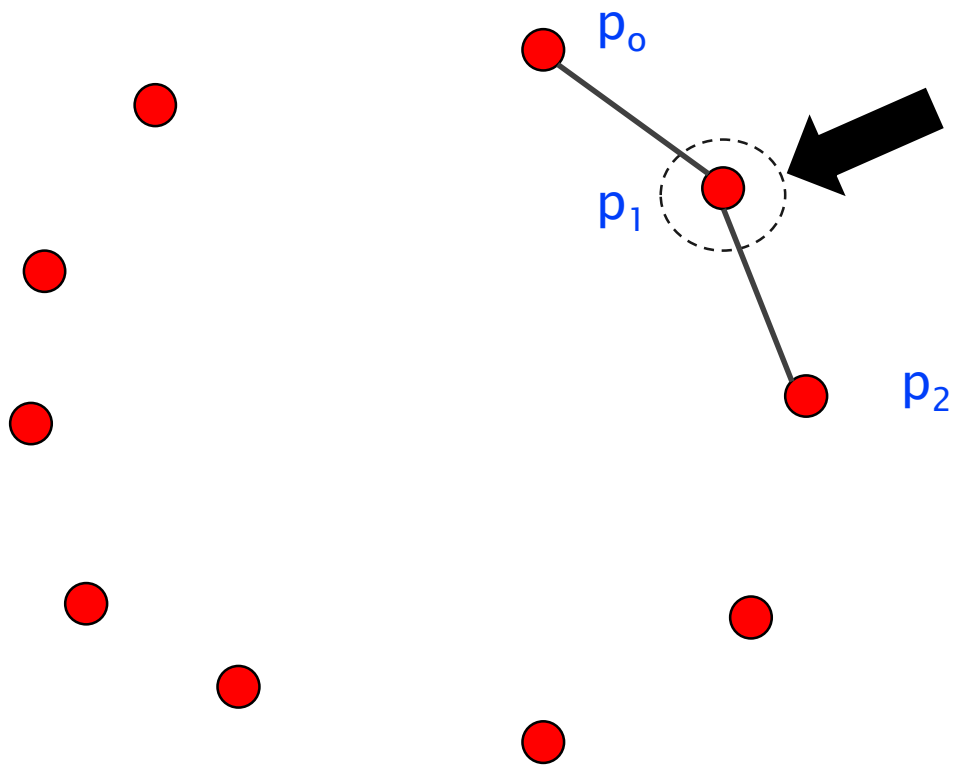
Heurística Nearest-Neighbor

- Inicie em algum ponto p_o e prossiga até o seu vizinho mais próximo p_1.



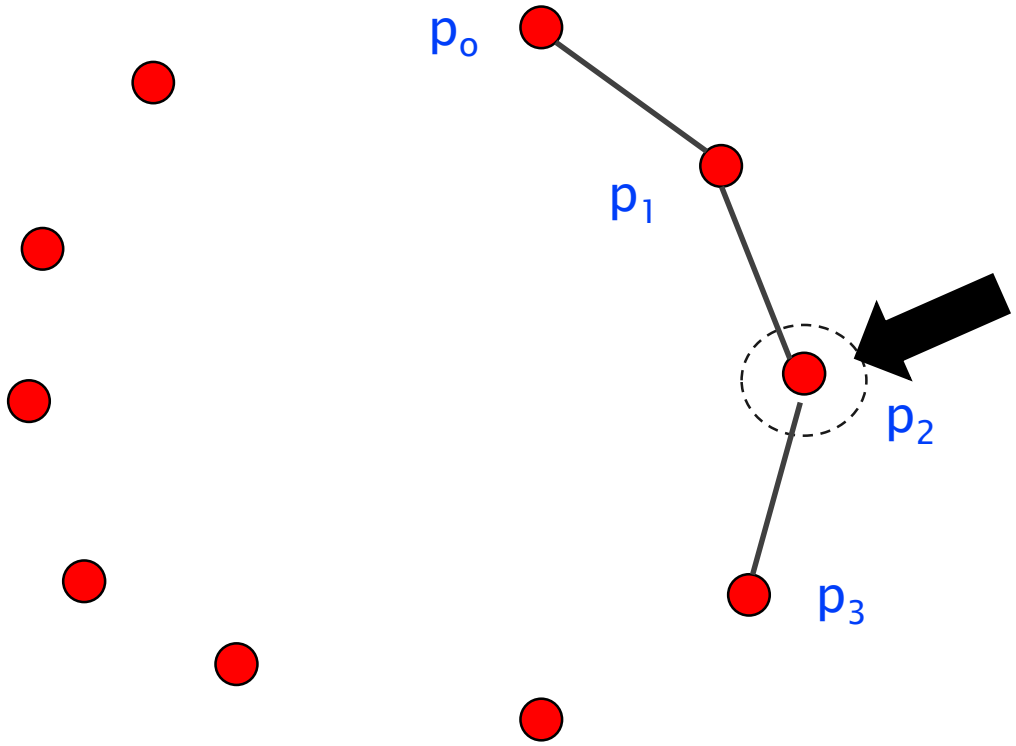
Heurística Nearest-Neighbor

- A partir de p_1, prossiga até seu vizinho mais próximo não-visitado, excluindo p_0 como um candidato.



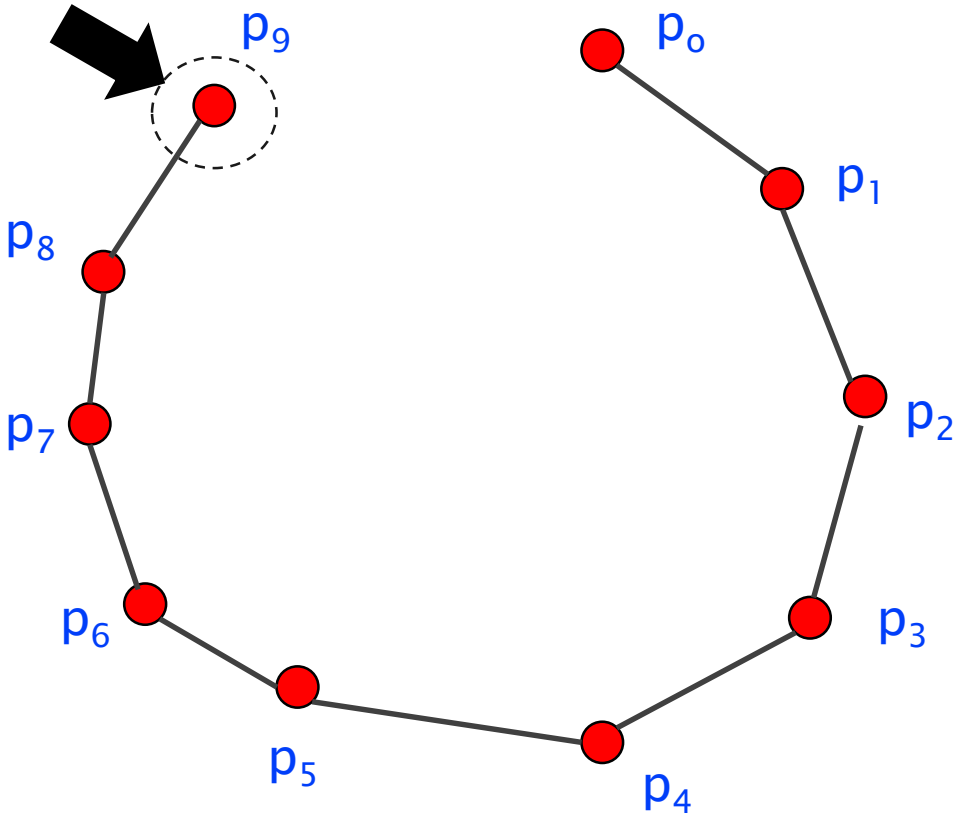
Heurística Nearest-Neighbor

- Repita este processo até passar por todos os pontos não-visitados.



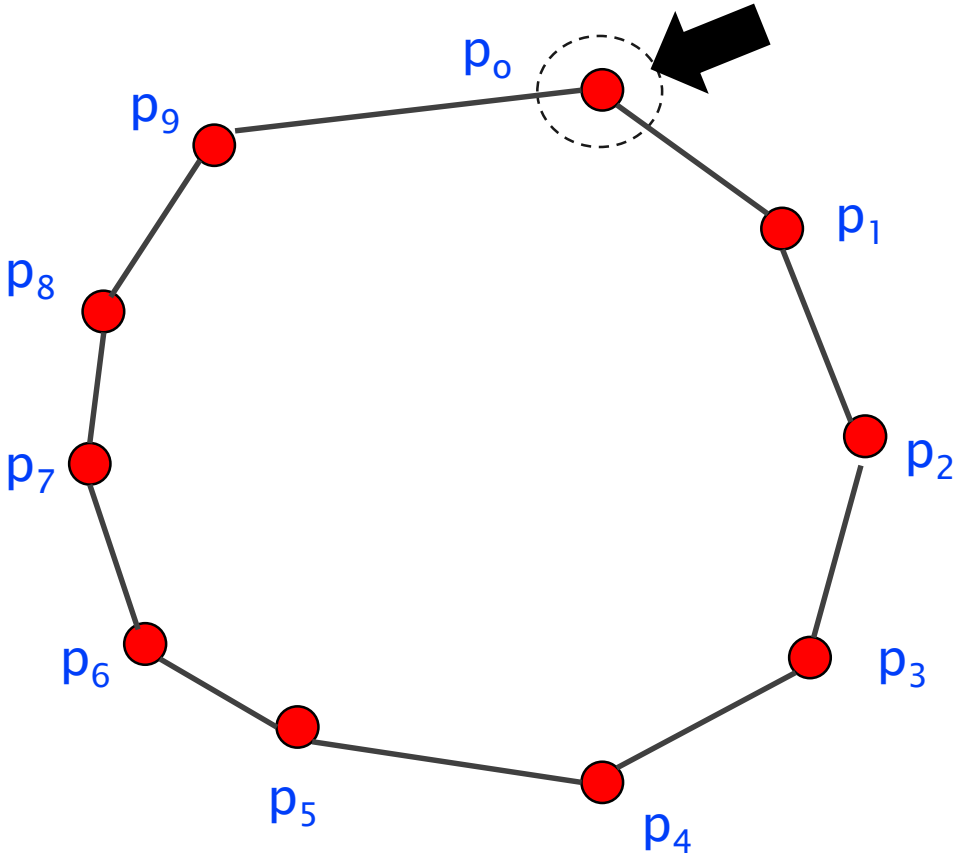
Heurística Nearest-Neighbor

- Repita este processo até passar por todos os pontos não-visitados.



Heurística Nearest-Neighbor

- Ao final retorne a p_o, fechando o ciclo.



Heurística Nearest-Neighbor

- Inicie em algum ponto po e prossiga até o seu vizinho mais próximo P_1.
- A partir de P_1, prossiga até seu vizinho mais próximo não-visitado, excluindo P_0 como um candidato.
- Repita este processo até passar por todos os pontos não-visitados.
- Ao final retorne a P_o, fechando o ciclo.

Heurística Nearest-Neighbor

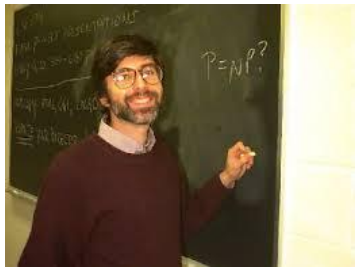
NearestNeighborTSP(P)

Pick and visit an initial point p_0

$p = p_0$
$i = 0$

While there are still unvisited points
    $i = i + 1$
    Let p_i be the closest unvisited point to p_{i-1}
    Visit p_i

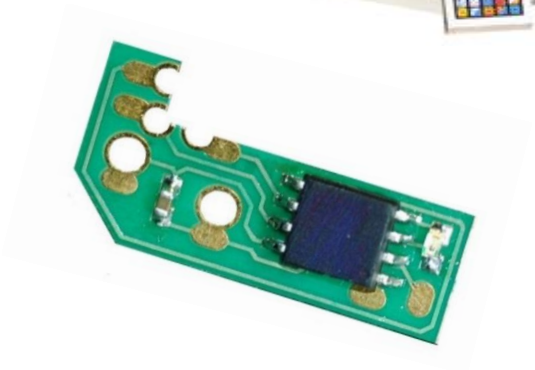
Return to p_0 from p_i



[Steven Skiena]

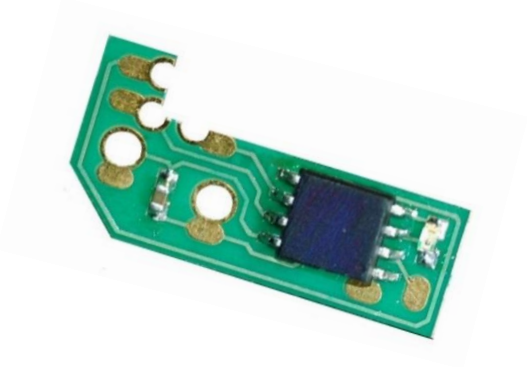
NearestNeighborTSP(P)

O que você achou do algoritmo ?



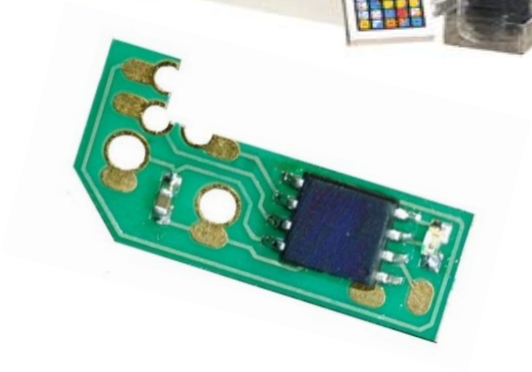
NearestNeighborTSP(P)

- ✓ Simples de se entender;
- ✓ Fácil de ser implementado;
- ✓ Faz sentido visitar pontos mais próximos se queremos minimizar o tempo total do ciclo;
- ✓ Trabalha perfeitamente na figura anterior.



NearestNeighborTSP(P)

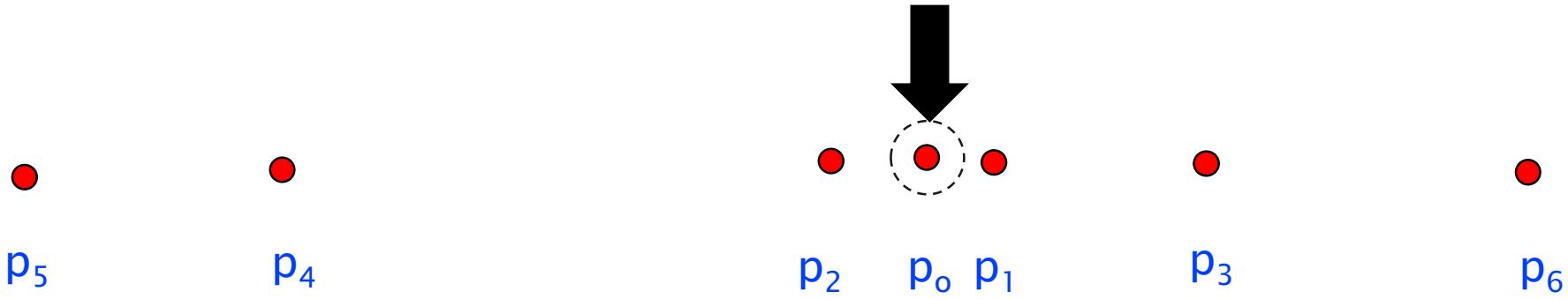
No entanto, o algoritmo nem sempre retorna a melhor rota!!!



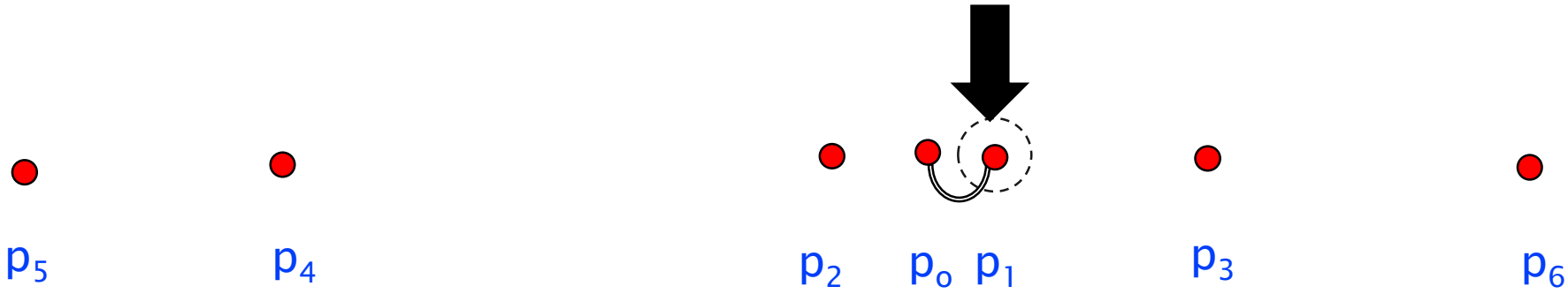
NearestNeighborTSP(P)

Vamos considerar outra instância do problema ...

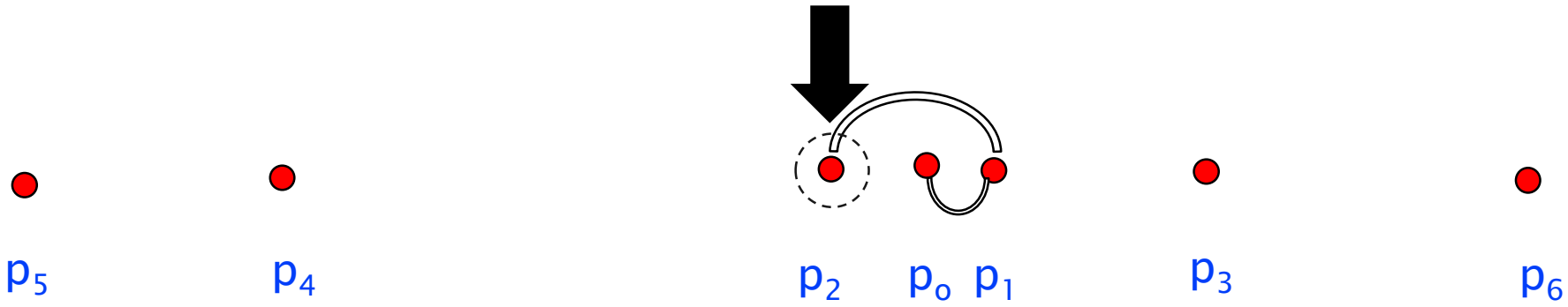
NearestNeighborTSP(P)



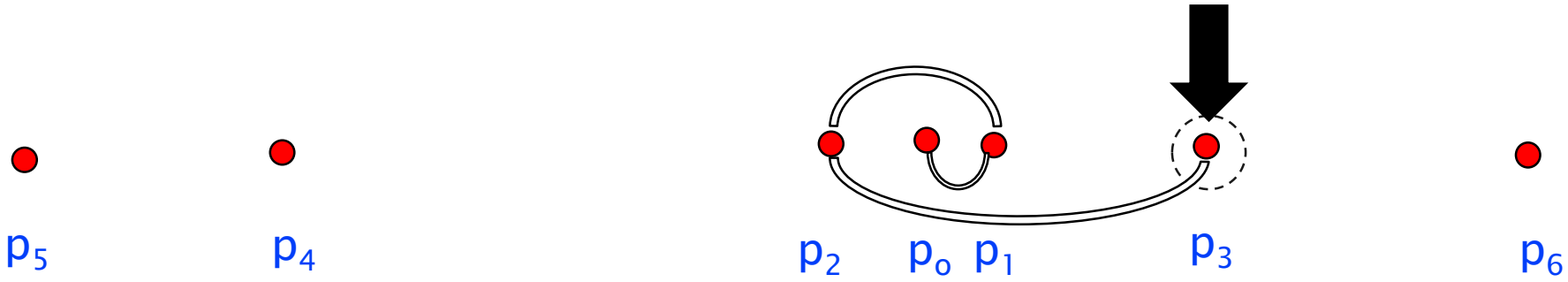
NearestNeighborTSP(P)



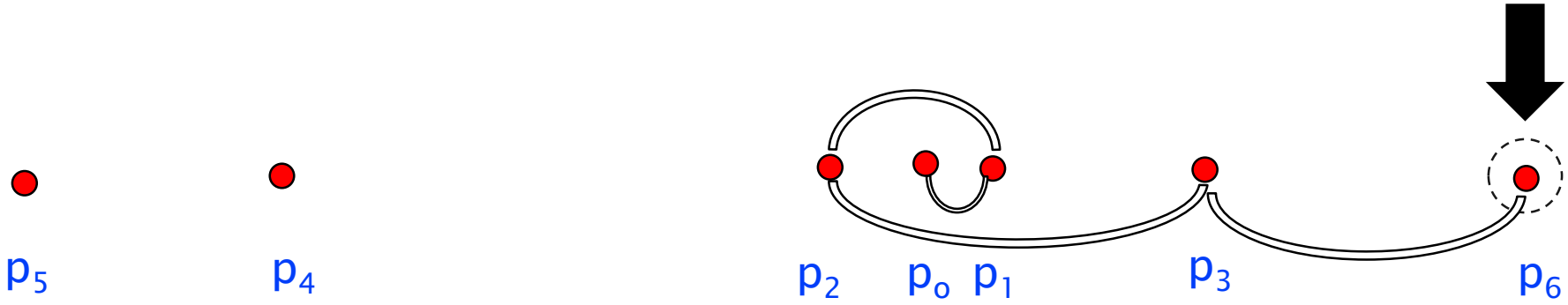
NearestNeighborTSP(P)



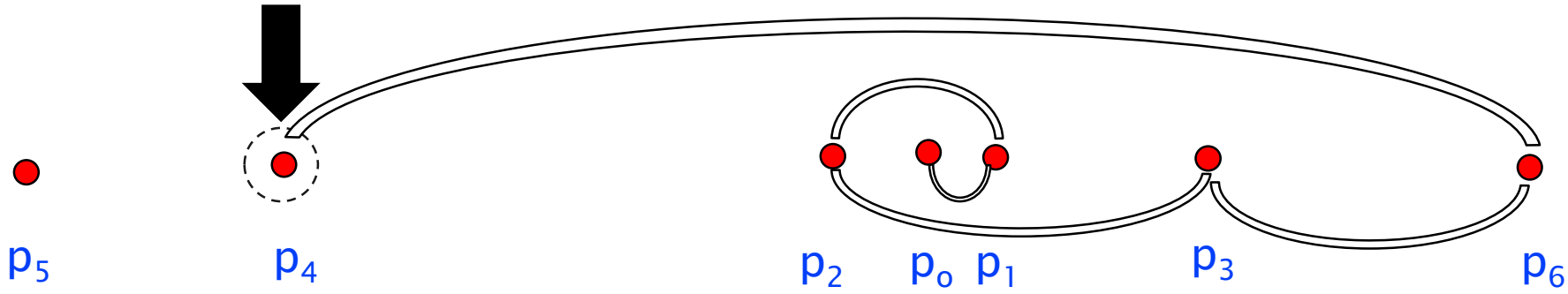
NearestNeighborTSP(P)



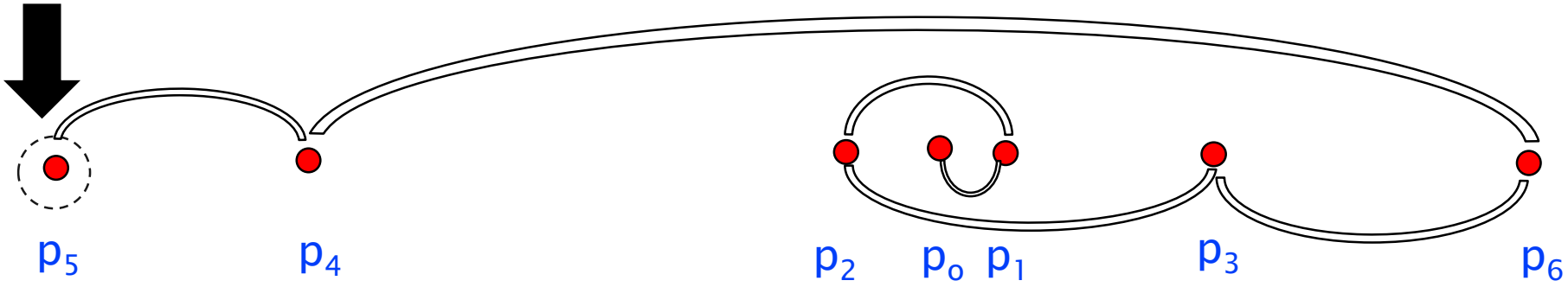
NearestNeighborTSP(P)



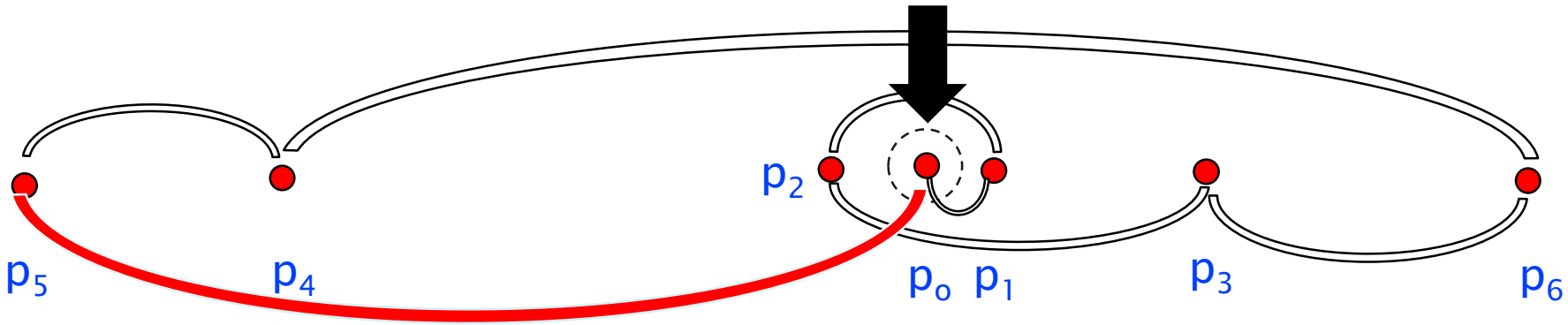
NearestNeighborTSP(P)



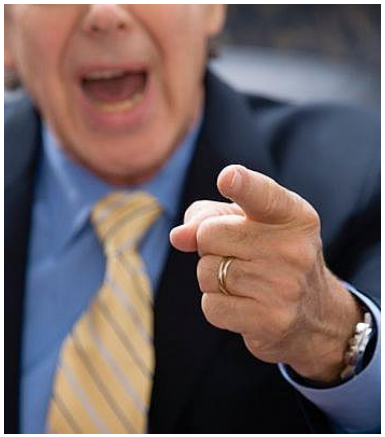
NearestNeighborTSP(P)

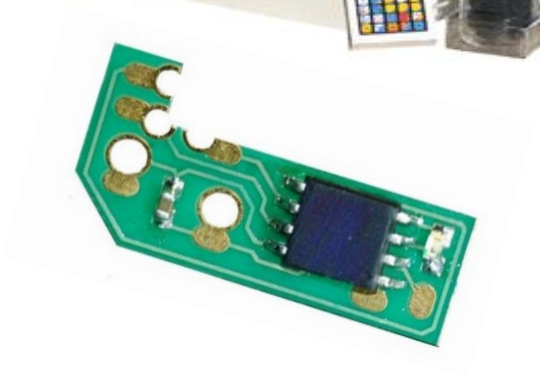


NearestNeighborTSP(P)



NearestNeighborTSP(P)



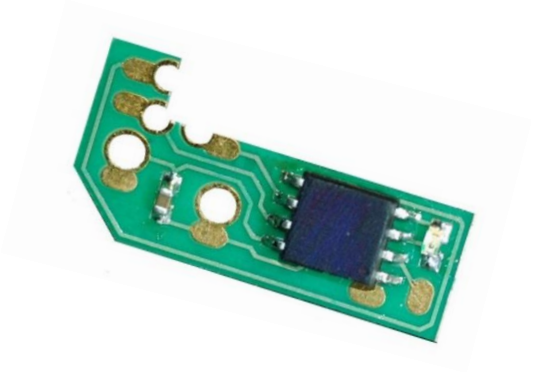


Imagine o seu chefe vendo o vai-e-vem do Robô...

Há algum algoritmo melhor ?

Busca Exaustiva

- ✓ Tente todas as possíveis combinações de pontos;
- ✓ Selecione dentre estas aquela que tem o menor custo;

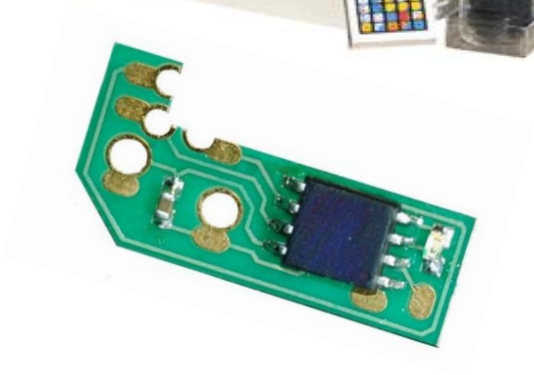


Busca Exaustiva

Desde que todas as combinações estão sendo consideradas, o algoritmo é correto.

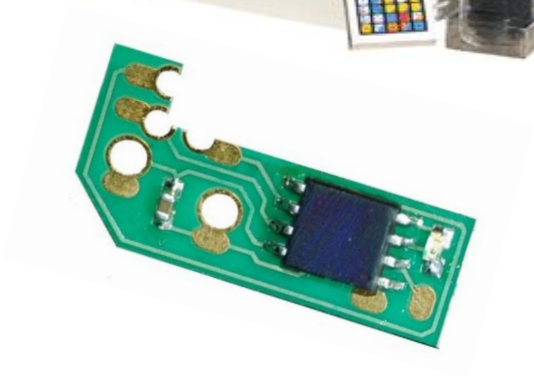
Busca Exaustiva

No entanto …



Busca Exaustiva

O algoritmo é extremamente lento . . .



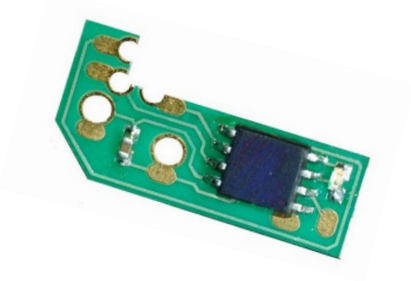
Busca Exaustiva

- ✓ Para 20 pontos:

20! = **2.432.902.008.176.640.000** combinações !

Busca Exaustiva

- ✓ Para outras situações, onde n≈ 1000 pontos;
- ✓ Esqueça este algoritmo…



Caixeiro Viajante

- ✓ O problema apresentado é clássico na Computação.
- ✓ Denomina-se **TSP** – **Travelling Salesman Problem**.

Caixeiro Viajante

- ✓ Não existe algoritmo correto e eficiente para este problema;
- ✓ O problema é atacado com **Heurísticas**.

Heurísticas

- ✓ Na Computação, busca-se criar algoritmos com tempo de execução **aceitável** e ser uma **solução ótima** para o problema em todas as suas instâncias;

- ✓ Um algoritmo **heurístico** pode encontrar boas soluções a maioria das vezes, mas **não há garantias** de que sempre as encontrará.

Usando heurísticas para Otimização de Consultas

- ✓ O analisador de consulta **SQL** gera primeiro uma estrutura de dados que corresponde a uma representação inicial da consulta, que é então otimizada com **regras heurísticas**;
- ✓ Isso leva a uma representação de consulta **otimizada**;
- ✓ Uma das principais regras heurísticas é aplicar operações de **SELEÇÃO** e **PROJEÇÃO** antes de aplicar a **JUNÇÃO**;

- ✓ As operações **SELEÇÃO** e **PROJEÇÃO**, em geral, **reduzem** o tamanho de um arquivo e, portanto, devem ser aplicadas **antes** de uma **JUNÇÃO**;

Árvore de Consulta

- ✓ Uma **árvore de consulta** é utilizada para representar uma expressão da **Álgebra Relacional**;
- ✓ Após a aplicação de **regras de otimização heurísticas**, essa árvore é convertida para uma árvore de consulta **equivalente** mais eficiente de ser executada;
- ✓ Ela representa as **relações** de entrada da consulta como **nós** folha da árvore;
- ✓ As **operações** da Álgebra Relacional são representadas pelos **nós internos**;
- ✓ Uma **execução** da árvore de consulta consiste na execução de uma operação de nó interno sempre que seus operandos estiverem disponíveis e depois na substituição desse nó interno pela relação que resulta da execução da operação;
- ✓ A ordem de execução das operações **começa** nos **nós folha**, que representa as relações de do banco de dados de entrada para a consulta, e **termina** no **nó raiz**, que representa a operação final da consulta.

Árvore de Consulta - Exemplo

- Considere o seguinte esquema de banco de dados:

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Árvore de Consulta - Exemplo

- Considere a seguinte consulta:

"Para cada projeto localizado em '**Mauá**', liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento."

```
SELECT   Projnumero, Dnum, Unome, Endereco, DataNasc

FROM PROJETO, DEPARTAMENTO, FUNCIONARIO

WHERE   Dnum = Dnumero   AND
                    CPF_gerente = CPF AND
                    Projlocal = 'Maua';
```

Árvore de Consulta - Exemplo

```
SELECT   Projnumero, Dnum, Unome, Endereco, DataNasc

FROM PROJETO, DEPARTAMENTO, FUNCIONARIO

WHERE   Dnum = Dnumero   AND
        CPF_gerente = CPF AND  Projlocal = 'Maua';
```

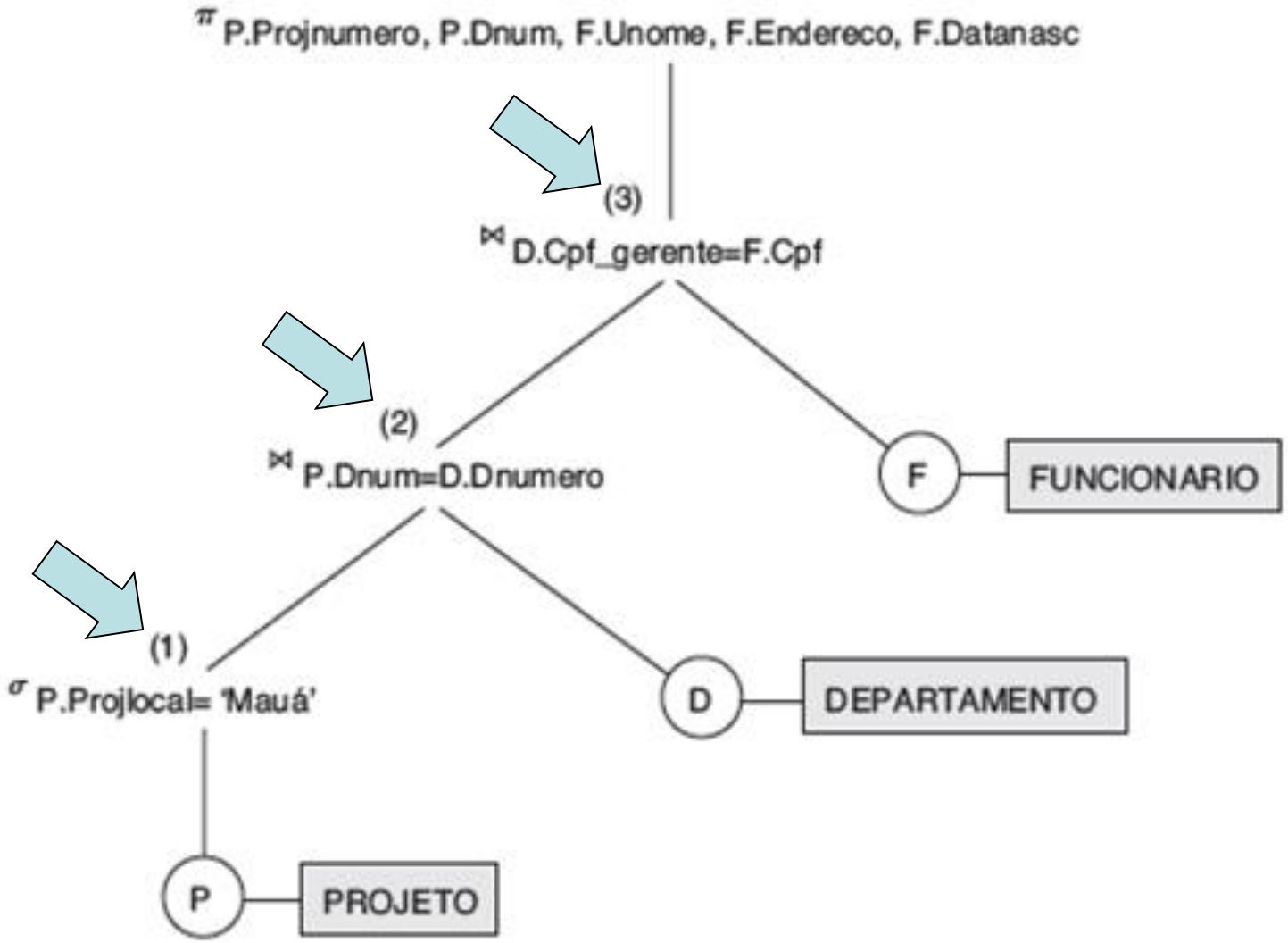
- A consulta corresponde a seguinte expressão da Álgebra Relacional:

π ProjNumero, Dnum, Unome, Endereco, Data_nasc

(((σ Projlocal = 'Maua' (**PROJETO**)) $\bowtie$ Dnum=Dnumero (**DEPARTAMENTO**))

$\bowtie$ CPF_gerente=CPF (**FUNCIONARIO**))

Árvore de Consulta - Exemplo

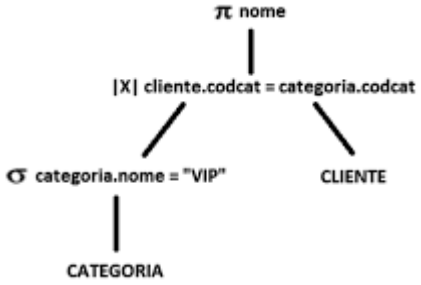


Otimização Heurística das Árvores de Consulta

- Em geral, muitas expressões diferentes da **Álgebra Relacional** e, portanto, muitas árvores de consulta diferentes, podem ser equivalentes: ou seja, podem representar a mesma consulta;

- O analisador de consulta normalmente gerará uma árvore de consulta inicial padrão correspondente a uma consulta **SQL**, **sem realizar qualquer otimização**;

- O otimizador de consulta heurística transformará a árvore de consulta inicial em uma árvore de consulta final equivalente, que é **eficiente** para ser executada.



Exemplo – Transformação de Consulta

- Considere a seguinte consulta C no banco de dados, cujo esquema é mostrado:

Encontre todos os sobrenomes dos funcionários nascidos após 1957, que trabalham em um projeto chamado '**Aquarius**'.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

PROJETO

Projnome	Projnumero	Projlocal	Dnum

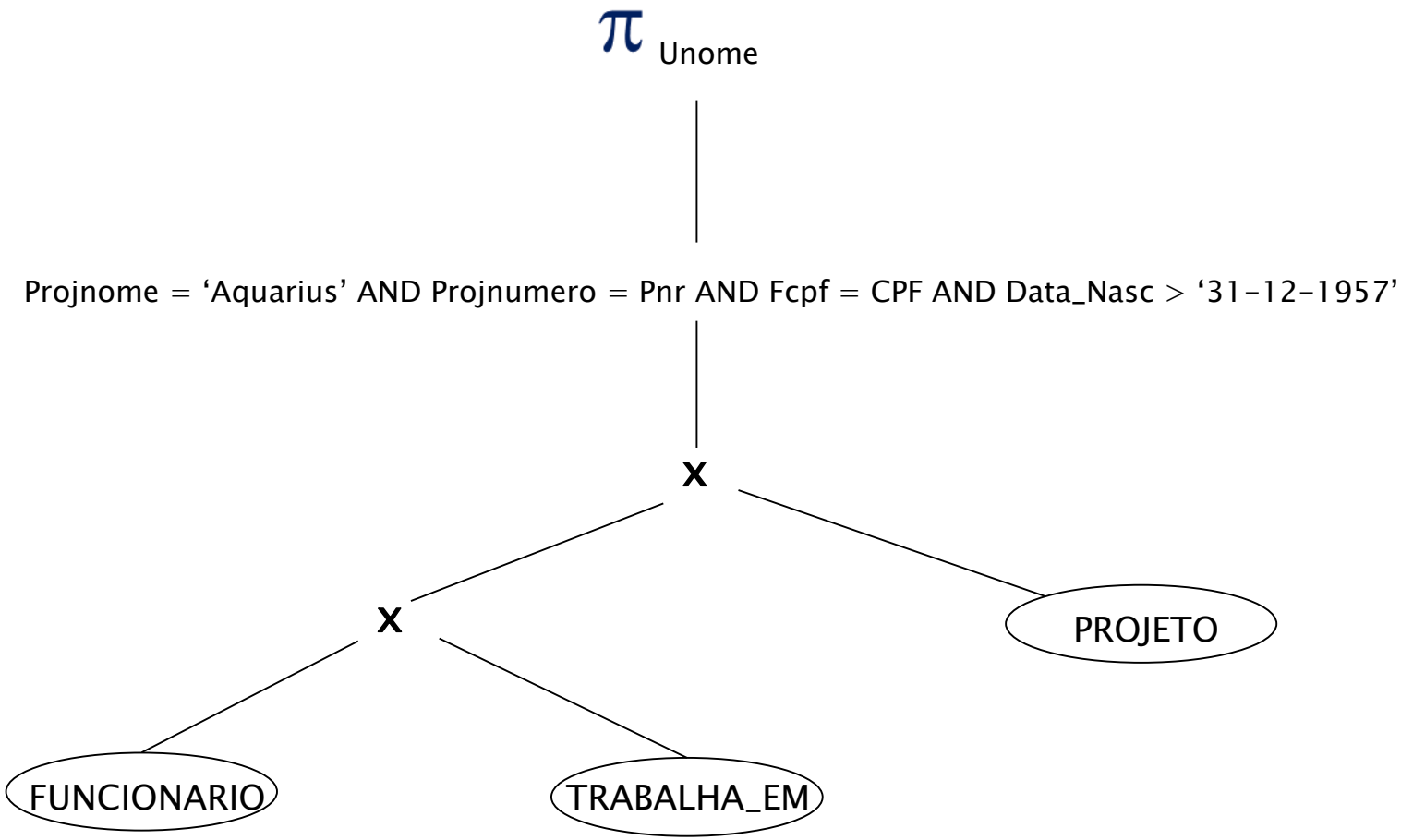
TRABALHA_EM

Fcpf	Pnr	Horas

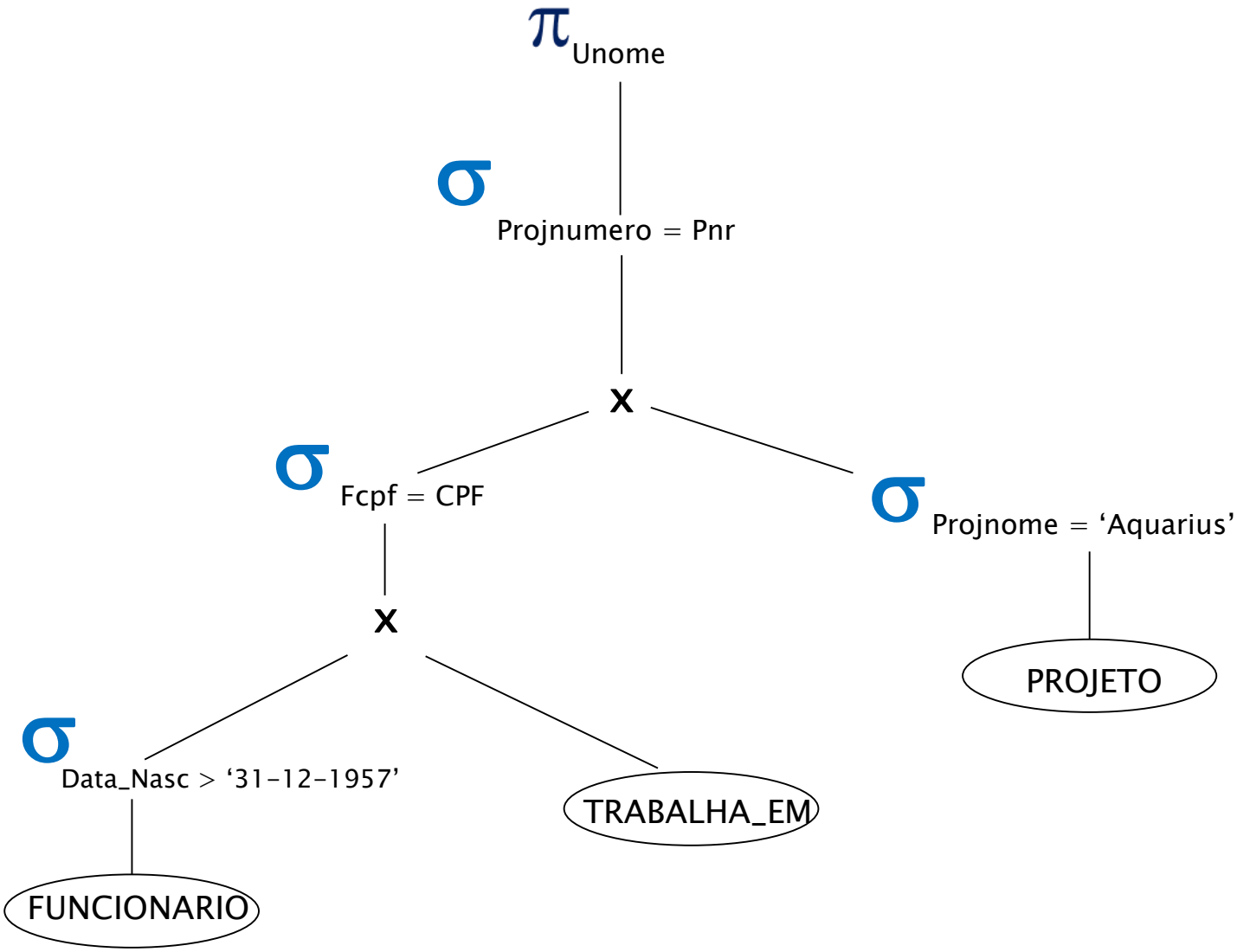
DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

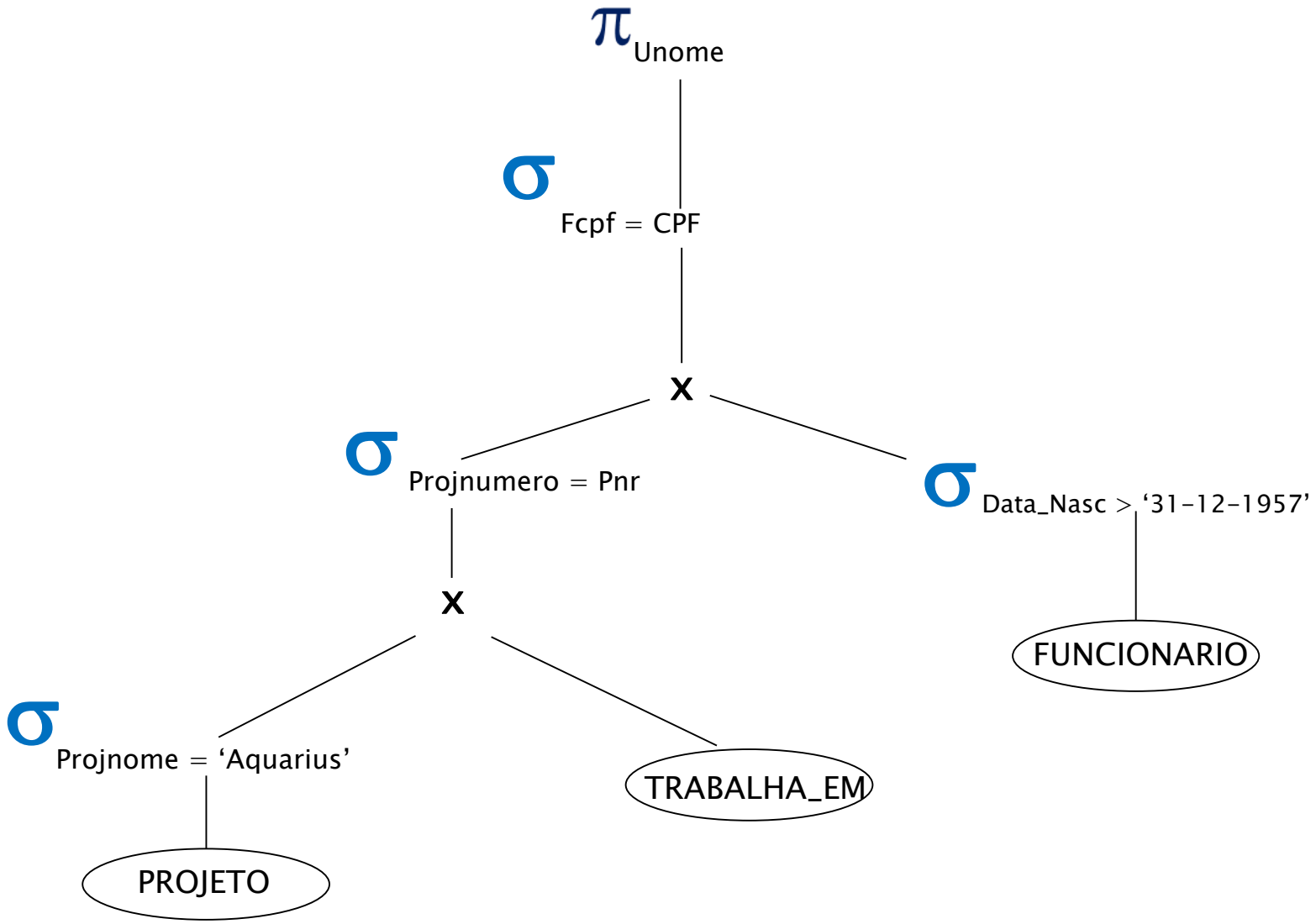
Exemplo – Árvore de Consulta Inicial



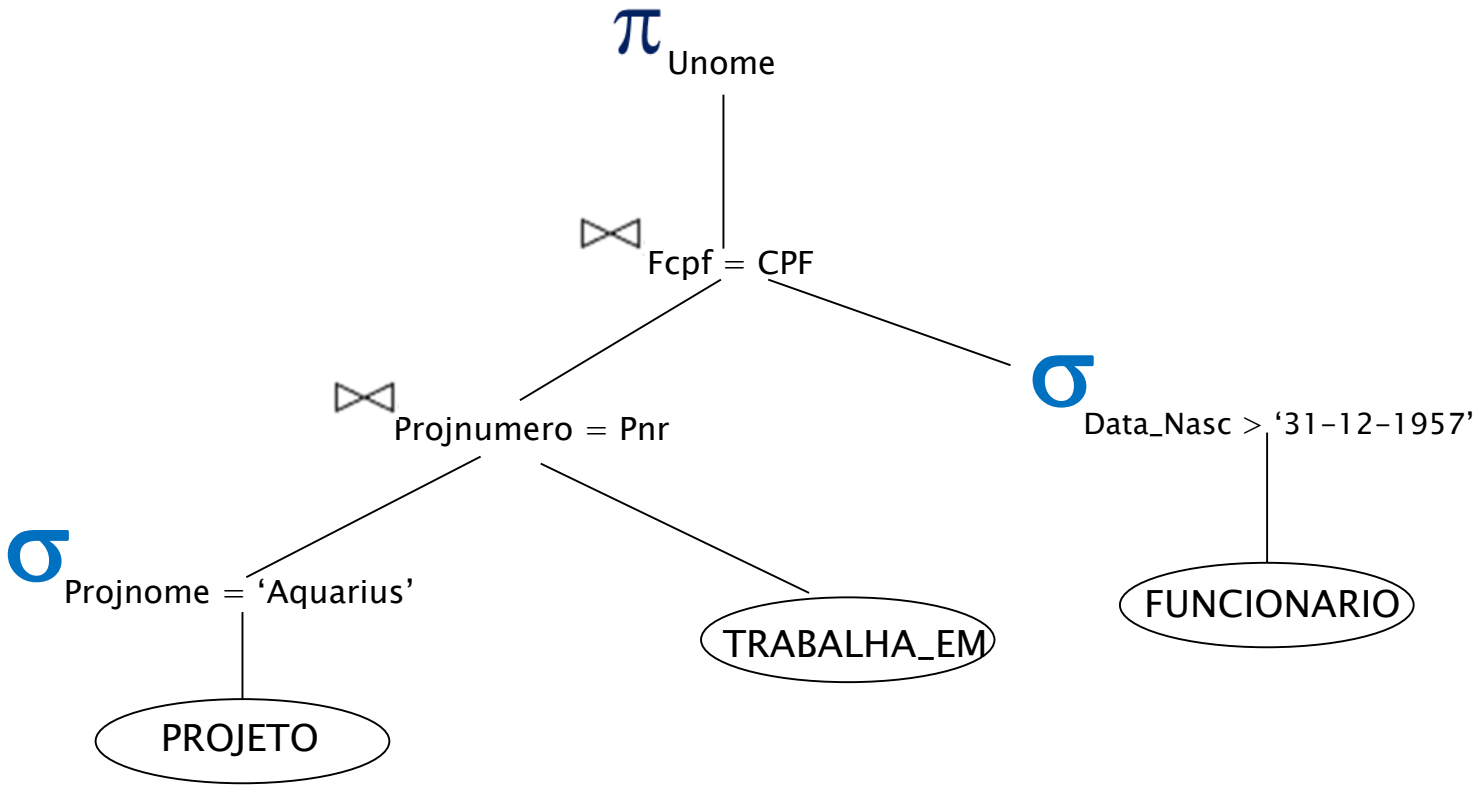
Exemplo – Árvore de Consulta – Passo 1



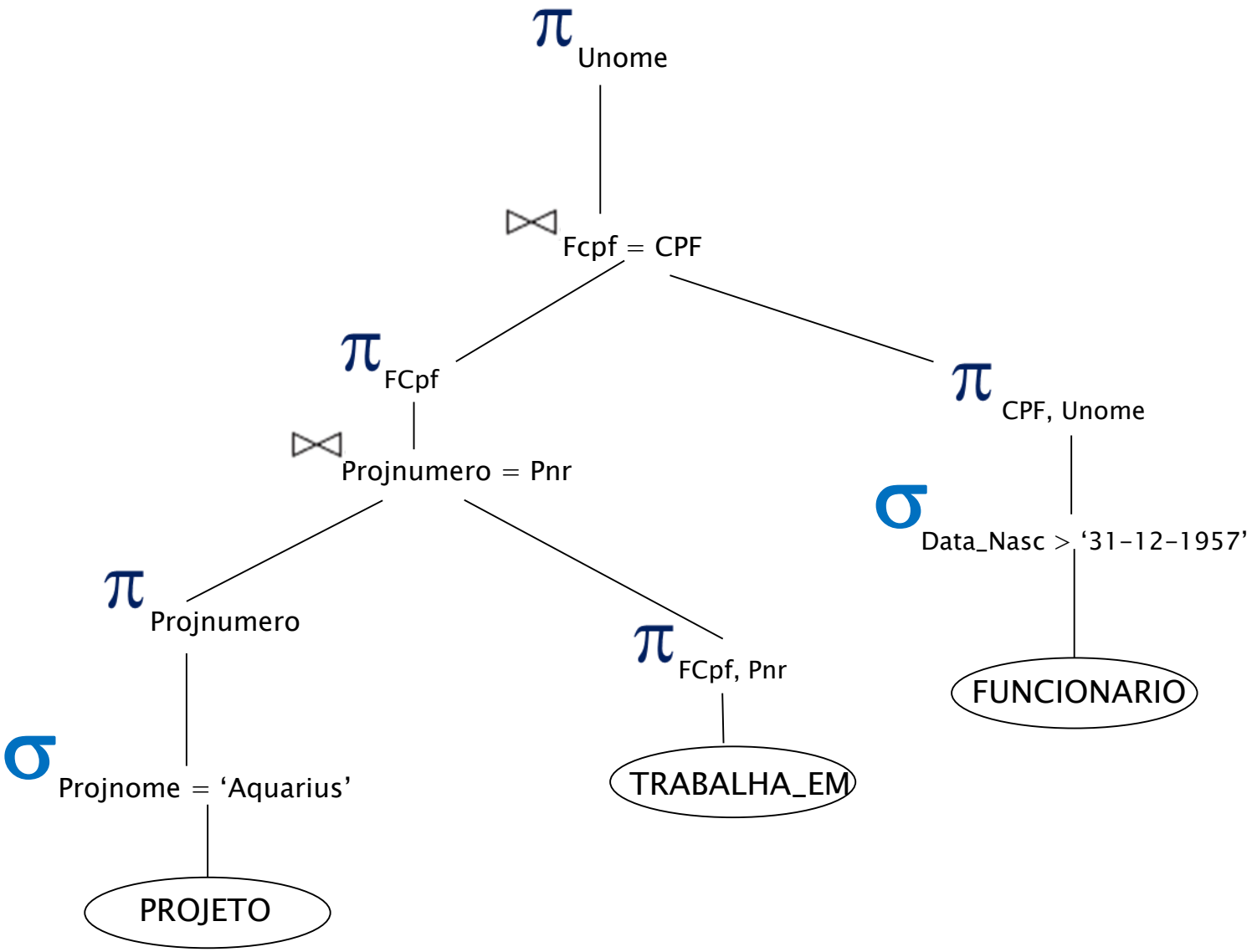
Exemplo – Árvore de Consulta – Passo 2



Exemplo – Árvore de Consulta – Passo 3

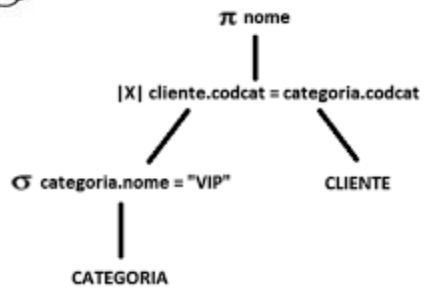


Exemplo – Árvore de Consulta – Passo 4



Otimização Heurística das Árvores de Consulta

- Conforme exemplo anterior, uma árvore de consulta pode ser transformada passo a passo em uma árvore de consulta equivalente que é mais eficiente de se executar;
- Porém, deve-se **garantir** que as etapas de transformações sempre levam a uma **árvore de consulta equivalente**;
- Para fazer isso, o **otimizador** de consulta precisa saber quais as regras de transformação preservam essa equivalência;
- Os detalhes das regras de transformação estão detalhadas em [**Elmasri,2011**].



Regras de Transformação – Árvore de Consulta

- Cascata de σ
- Comutatividade de σ
- Cascata de π
- Comutação de σ com π
- Comutatividade de ⋈ (e X)
- Comutação de σ com ⋈ ou (X)
- Comutação de π com ⋈ ou (X)
- Comutatividade das operações de conjunto
- Associatividade de ⋈ , X , U e ∩
- Comutação de σ com operações de conjunto
- A operação π comuta com U
- Conversão de um sequência de (σ,X) em ⋈