ACH2025

Laboratório de Bases de Dados

Aula 15

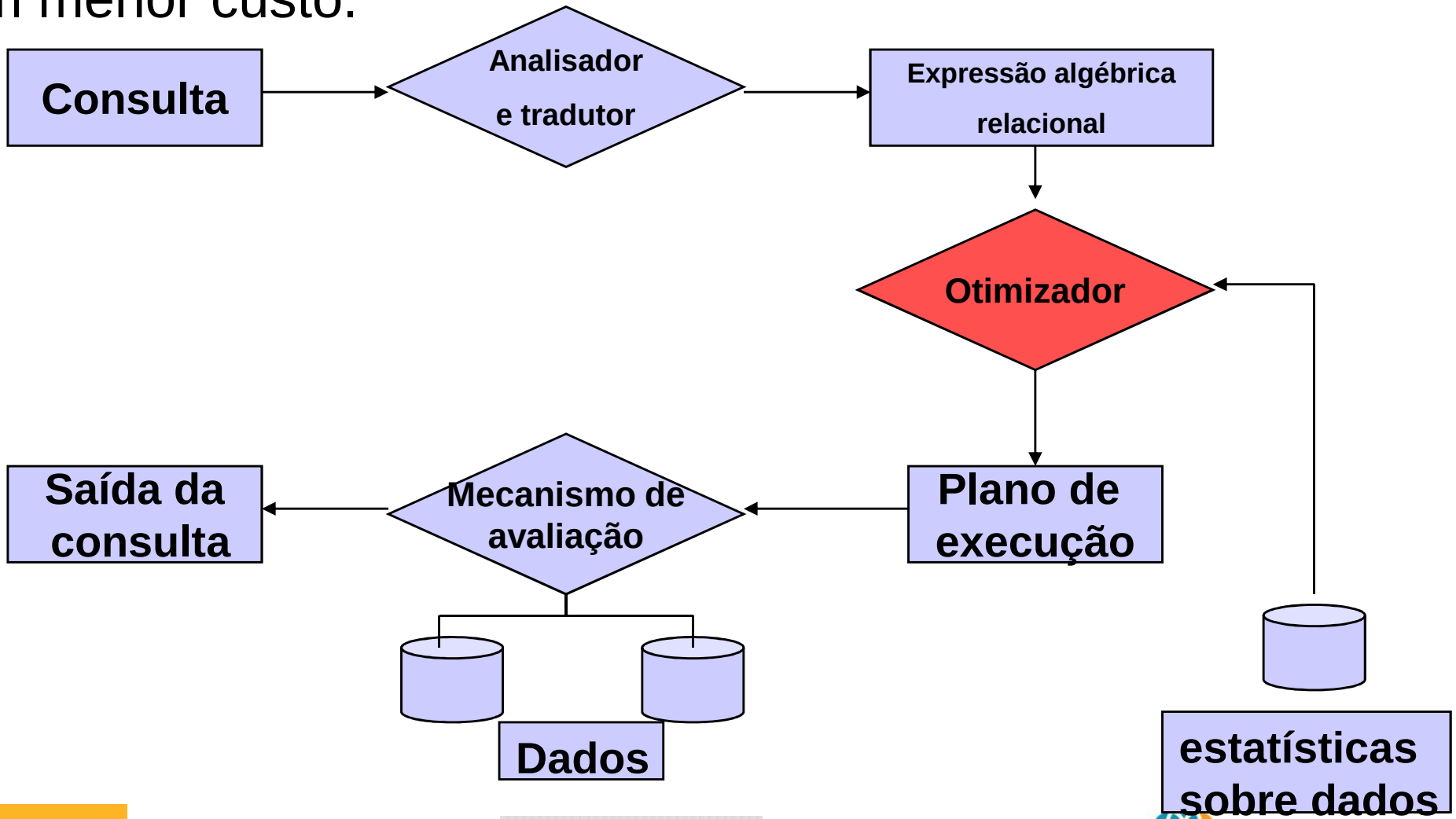
Processamento de Consultas – Parte 2
Otimização

Professora:

- **Fátima L. S. Nunes**

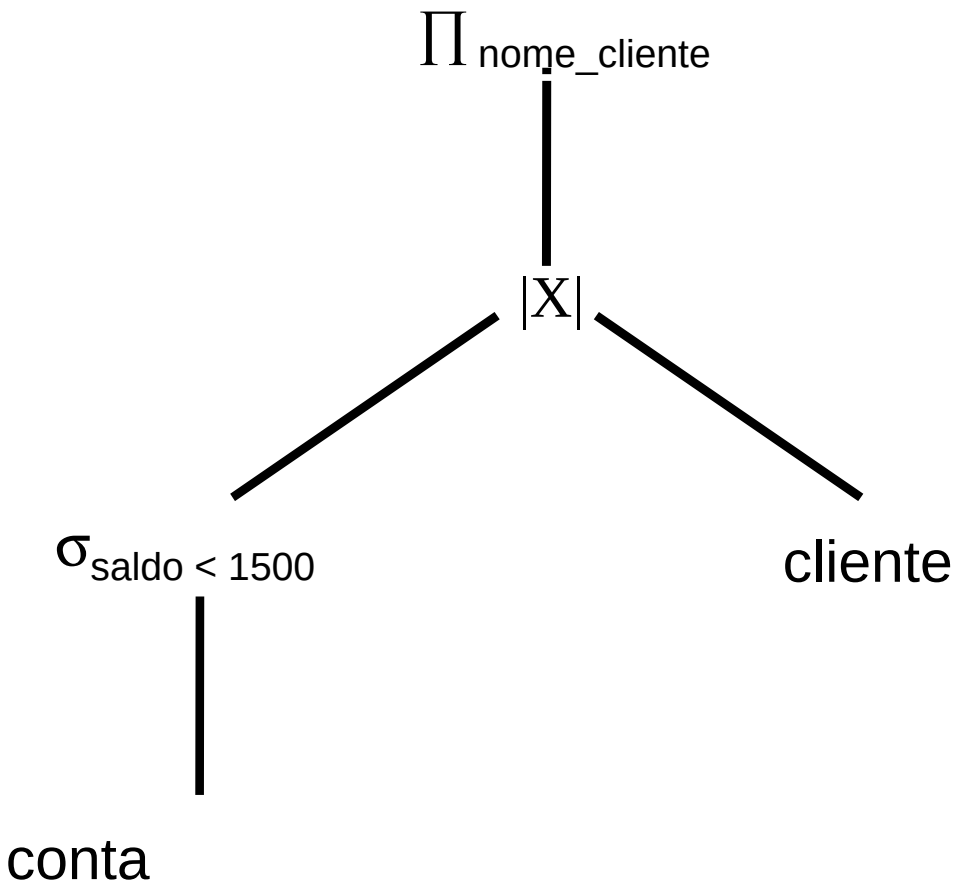
Contexto

- ✓ Dada uma expressão em álgebra relacional, o otimizador deve propor um plano de avaliação que gere o mesmo resultado com menor custo.



Avaliação materializada

$$\Pi_{nome\_cliente}\ (\sigma_{saldo < 1500}\ (conta)\ |X|\ cliente)$$

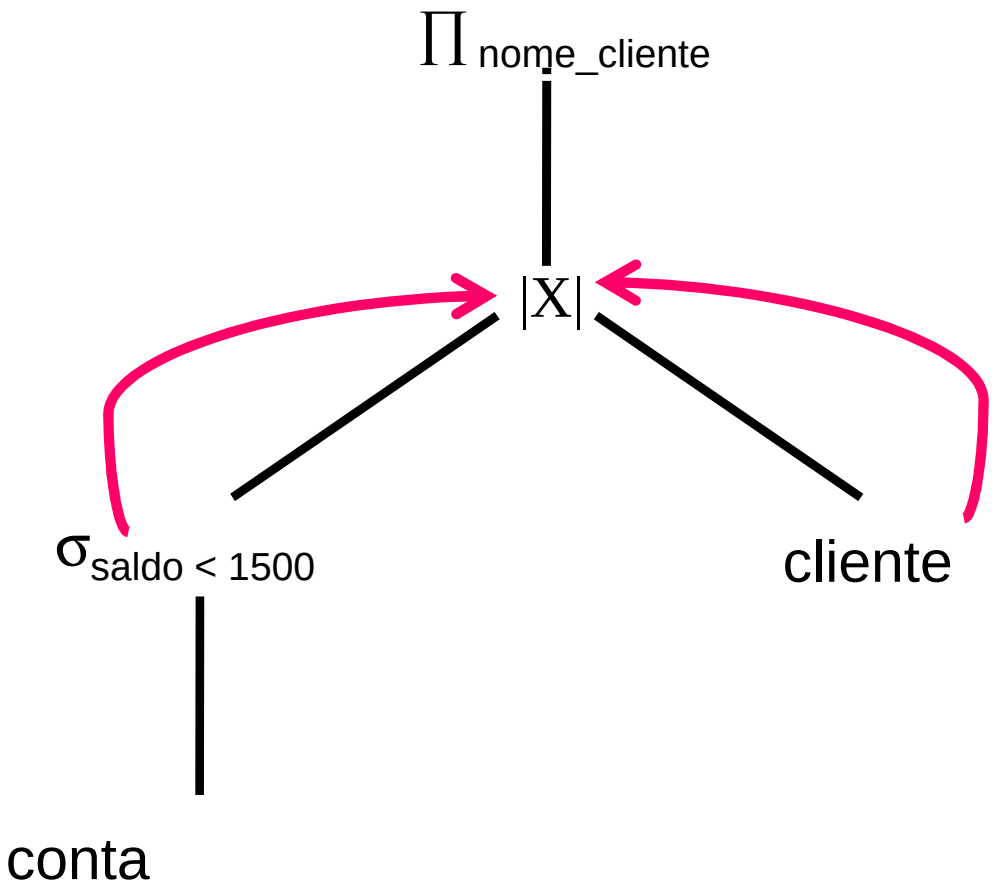


Começa com operações de nível mais baixo, que têm como entrada as relações do BD. Cada operação tem resultado armazenado em **relação temporária** no BD.

Deve-se considerar o custo das operações envolvidas e do armazenamento temporário.

Avaliação pipeline

$$\Pi_{nome\_cliente}\ (\sigma_{saldo < 1500}\ (conta)\ |X|\ cliente)$$



Começa com operações de nível mais baixo, que têm como entrada as relações do BD.

Resultado de uma operação passado para próxima, **sem armazenamento temporário**.

Contexto

- ✓ Dada uma expressão em álgebra relacional, o otimizador deve propor um plano de avaliação que gere o mesmo resultado com menor custo.
- ✓ Para isso, é necessário gerar planos alternativos.
- ✓ Como?
  1. gerar expressões logicamente equivalentes à expressão dada
  2. escrever expressões resultantes de maneiras alternativas para gerar planos de avaliação alternativos.
  3. escolher plano de avaliação de consulta com base nos custos estimados.

Regras de equivalência

- ✓ Expressões de duas formas diferentes são equivalentes
- ✓ Preservar a equivalência = relações geradas têm mesmos atributos e mesmas tuplas (ordem não é importante).
- ✓ Notação:
  - $\theta, \theta_1, ..., \theta_n$ são predicados
  - $L_1, L_2, , ..., L_n$ são listas de atributos
  - $E_1, E_2, , ..., E_n$ são expressões da álgebra relacional

Regras de equivalência

1. As operações de uma seleção conjuntiva podem ser divididas em uma sequência de seleções individuais (cascata de σ).

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. As operações de seleção são comutativas.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

Regras de equivalência

3. Somente operações finais são necessárias em uma sequência de operações de projeção.

$$\Pi_{t_1}(\Pi_{t_2}(\ldots(\Pi_{tn}(E))\ldots)) = \Pi_{t_1}(E)$$

4. Seleções podem ser combinadas com os produtos cartesianos e junções theta.

$$\sigma_{\theta}(E_1 \text{ X } E_2) = E_1 \,|X|_{\theta}\, E_2$$

$$\sigma_{\theta 1}(E_1 \,|X|_{\theta 2}\, E_2) = E_1 \,|X|_{\theta 1 \wedge \theta 2}\, E_2$$

Regras de equivalência

5. Operações de junção theta (e junções naturais) são comutativas

$$E_1 \,|X|_{\theta}\, E_2 = E_2 \,|X|_{\theta}\, E_1$$

6. (a) Operações de junção natural são associativas:

$$(E_1 \,|X|\, E_2) \,|X|\, E_3 = E_1 \,|X|\, (E_2 \,|X|\, E_3)$$

(b) Junções theta são associativas da seguinte maneira:

$$(E_1 \,|X|_{\theta 1}\, E_2) \,|X|_{\theta 2 \wedge \theta 3}\, E_3 = E_1 \,|X|_{\theta 1 \wedge \theta 3}\, (E_2 \,|X|_{\theta 2}\, E_3)$$

onde θ_2 contém somente atributos de E_2 e E_3.

Regras de equivalência

7. A operação de seleção pode ser distribuída por meio da operação de junção theta, observando as seguintes condições:

(a) Quando todos os atributos de θ_0 envolvem somente os atributos de uma das expressões (E_1) que estão participando da junção.

$$\sigma_{\theta 0}(E_1 \,|X|_{\theta}\, E_2) = (\sigma_{\theta 0}(E_1))\, |X|_{\theta}\, E_2$$

(b) Quando θ_1 envolve somente atributos de E_1 e θ_2 envolve somente atributos de E_2.

$$\sigma_{\theta 1 \wedge \theta 2}\,(E_1 \,|X|_{\theta}\, E_2) = \; (\sigma_{\theta 1}(E_1))\, |X|_{\theta}\, (\sigma_{\theta 2}\,(E_2))$$

Regras de equivalência

8. A operação de projeção é distribuída pela operação de junção theta como segue:
   - Sejam L_1 e L_2 conjuntos de atributos de E_1 e E_2, respectivamente.

   (a) se θ envolve unicamente atributos de $L_1 \cup L_2$:

$$\Pi_{L_1 \cup L_2}(E_1 \mid X \mid_\theta E_2) = (\Pi_{L_1}(E_1)) \mid X \mid_\theta (\Pi_{L_2}(E_2))$$

   (b) Considere uma junção $E_1 |X|_\theta E_2$.
   - Seja L_3 os atributos de E_1 que são envolvidos na condição de junção θ, mas não estão em $L_1 \cup L_2$, e
   - Seja L_4 os atributos de E_2 que são envolvidos na condição de junção θ, mas não estão em $L_1 \cup L_2$.

$$\Pi_{L_1 \cup L_2}(E_1 \mid X \mid_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \mid X \mid_\theta (\Pi_{L_2 \cup L_4}(E_2)))$$

Regras de equivalência

9. O conjunto de operações união e intersecção são comutativas

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

- (diferença de conjuntos não é comutativa)

10. União e intersecção são associativas

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

Regras de equivalência

11. A operação seleção é distribuída sobre $\cup$, $\cap$ e –.

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2)$$

e similarmente para $\cup$ e $\cap$ no lugar de –

Também:

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

e similarmente para $\cap$ no lugar de –,

mas não para $\cup$

12. A operação de projeção é distribuída por meio da operação de união

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Exemplo de transformação

- ✓ Esquema_agencia = (nome_agência, cidade_agência, fundos)
- ✓ Esquema_conta = (nome_agência, número_conta, saldo)
- ✓ Esquema_dono_conta= (nome_cliente, número_conta)

Exemplo de transformação

- ✓ Mostrar os nomes de todos os clientes que têm uma conta em alguma agência localizada em Brooklyn.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência} \mid X \mid (\textit{conta} \mid X \mid \textit{dono-conta})))$$

- ✓ Como transformar usando regra 7 ?

7. A operação de seleção pode ser distribuída por meio da operação de junção theta, observando as duas condições:

(a) Quando todos os atributos de θ_0 envolvem somente os atributos de uma das expressões (E_1) que estão participando da junção.

$$\sigma_{\theta 0}(E_1 \mid X \mid_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \mid X \mid_{\theta} E_2$$

(b) Quando θ_1 envolve somente atributos de E_1 e θ_2 envolve somente atributos de E_2.

$$\sigma_{\theta 1 \wedge \theta 2} (E_1 \mid X \mid_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \mid X \mid_{\theta} (\sigma_{\theta 2} (E_2))$$

Exemplo de transformação

- ✓ Mostrar os nomes de todos os clientes que têm uma conta em alguma agência localizada em Brooklyn.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência} \mid\!X\!\mid (\textit{conta} \mid\!X\!\mid \textit{dono-conta})))$$

- ✓ Como transformar usando regra 7 ?

$$\Pi_{\textit{nome-cliente}} ((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência})) \mid\!X\!\mid (\textit{conta} \mid\!X\!\mid \textit{dono-conta}))$$

7. A operação de seleção pode ser distribuída por meio da operação de junção theta, observando as duas condições:

(a) Quando todos os atributos de θ_0 envolvem somente os atributos de uma das expressões (E_1) que estão participando da junção.

$$\sigma_{\theta 0}(E_1 \mid\!X\!\mid_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \mid\!X\!\mid_{\theta} E_2$$

(b) Quando θ_1 envolve somente atributos de E_1 e θ_2 envolve somente atributos de E_2.

$$\sigma_{\theta 1 \wedge \theta 2} (E_1 \mid\!X\!\mid_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \mid\!X\!\mid_{\theta} (\sigma_{\theta 2} (E_2))$$

Exemplo de transformação

- ✓ Mostrar os nomes de todos os clientes que têm uma conta em alguma agência localizada em Brooklyn.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência} \mid\!X\!\mid (\textit{conta} \mid\!X\!\mid \textit{dono-conta})))$$

- ✓ Como transformar usando regra 7 ?

$$\Pi_{\textit{nome-cliente}} ((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência})) \mid\!X\!\mid (\textit{conta} \mid\!X\!\mid \textit{dono-conta}))$$

- ✓ Há vantagens ?

Exemplo de transformação

- ✓ Mostrar os nomes de todos os clientes que têm uma conta em alguma agência localizada em Brooklyn.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência} (\textit{conta} \mid\!X\!\mid \textit{dono-conta})))$$

- ✓ Como transformar usando regra 7 ?

$$\Pi_{\textit{nome-cliente}} ((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência})) \mid\!X\!\mid (\textit{conta} \mid\!X\!\mid \textit{dono-conta}))$$

- ✓ Há vantagens ?
  - ✓ relações intermediárias menores

Exemplo de várias transformações

- ✓ Mostrar os nomes de todos os clientes com uma conta numa agência de Brooklyn cujo saldo é maior de $1000.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”} \wedge \textit{saldo} > 1000} (\textit{agência} \mid X \mid (\textit{conta} \mid X \mid \textit{dono-conta})))$$

- ✓ Transformação usando junção associativa (Regra 6):

6. (a) Operações de junção natural são associativas:

$$(E_1 \mid X \mid E_2) \mid X \mid E_3 = E_1 \mid X \mid (E_2 \mid X \mid E_3)$$

(b) Junções theta são associativas da seguinte maneira:

$$(E_1 \mid X \mid_{\theta 1} E_2) \mid X \mid_{\theta 2 \wedge \theta 3} E_3 = E_1 \mid X \mid_{\theta 1 \wedge \theta 3} (E_2 \mid X \mid_{\theta 2} E_3)$$

onde θ_2 contém somente atributos de E_2 e E_3.

Exemplo de várias transformações

- ✓ Mostrar os nomes de todos os clientes com uma conta numa agência de Brooklyn cujo saldo é maior de $1000.

$$\Pi_{\textit{nome-cliente}} (\sigma_{\textit{cidade-agência} = \text{“Brooklyn”} \wedge \textit{saldo} > 1000} (\textit{agência} \mid X \mid (\textit{conta} \mid X \mid \textit{dono-conta})))$$

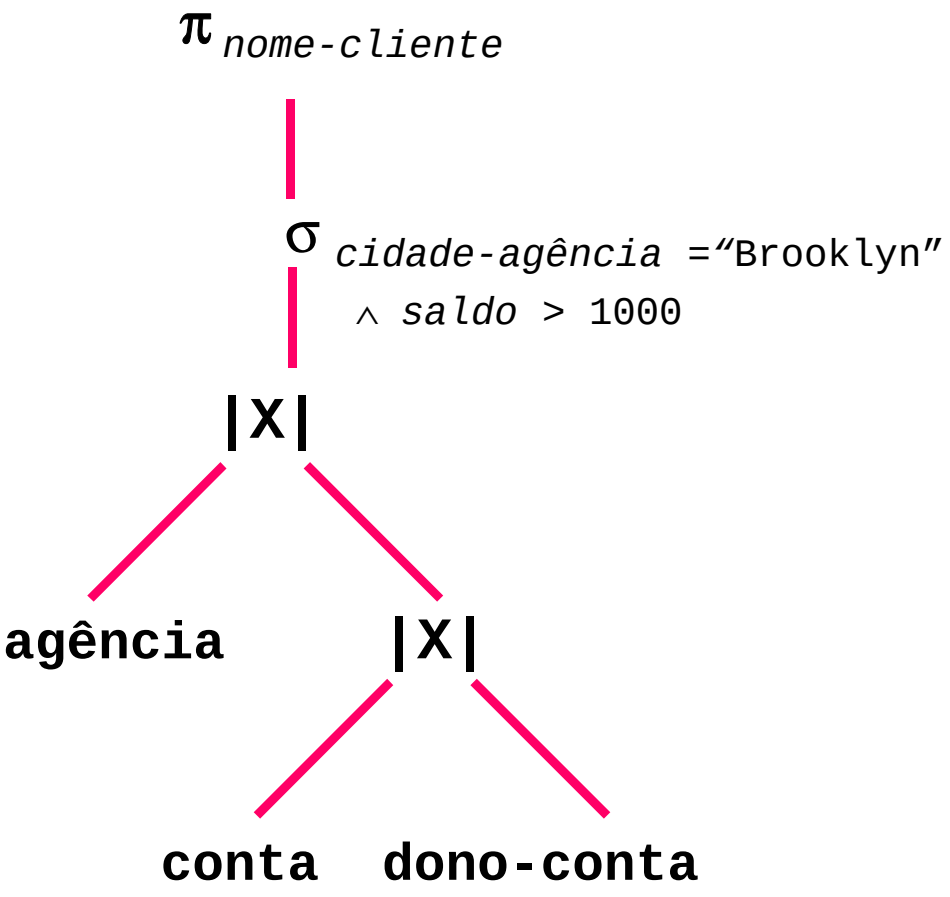
- ✓ Transformação usando junção associativa (Regra 6):

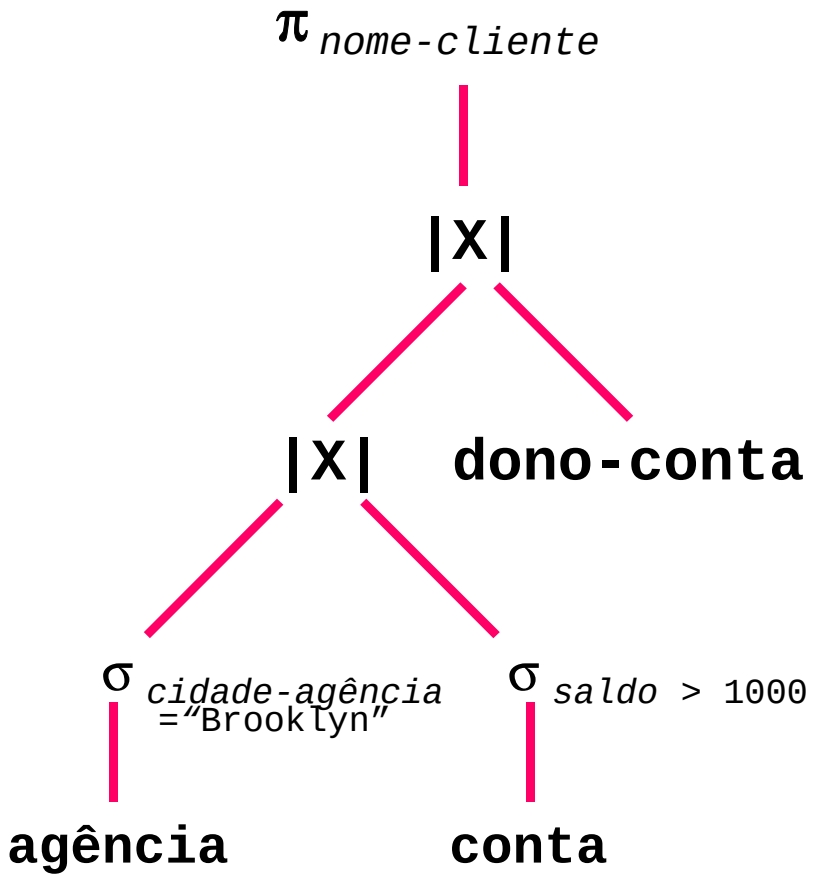
$$\Pi_{\textit{nome-cliente}} ((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”} \wedge \textit{saldo} > 1000} (\textit{agência} \mid X \mid (\textit{conta}))) \mid X \mid \textit{dono-conta})$$

Segunda forma fornece uma oportunidade para aplicar a regra 7:

$$\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência}) \mid X \mid \sigma_{\textit{saldo} > 1000} (\textit{conta})$$

Múltiplas Transformações





$\Pi_{\text{nome-cliente}}(\sigma_{\text{cidade-agência} = \text{“Brooklyn”} \wedge \text{saldo} > 1000}(\text{agência} \mid X \mid (\text{conta} \mid X \mid \text{dono-conta})))$

$\Pi_{\text{nome-cliente}}(\sigma_{\text{cidade-agência} = \text{“Brooklyn”}}(\text{agência}) \mid X \mid \sigma_{\text{saldo} > 1000}(\text{conta}) \mid X \mid \text{dono-conta})$

Exemplo da Operação de Projeção

$\Pi_{\textit{nome-cliente}}((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência}) \,|X|\, \textit{conta}) \,|X|\, \textit{dono-conta})$

- ✓ Quando calculamos

  $(\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência}) \,|X|\, \textit{conta})$

  obtemos uma relação cujo esquema é:
  (*nome-agência,cidade-agência, ativo, número-conta, saldo)*

- ✓ Usando a regra de equivalência 8b é possível eliminar atributos não necessários de resultados intermediários:

$\Pi_{\textit{nome-cliente}}((\Pi_{\textit{número-conta}}((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}} (\textit{agência}) \,|X|\, \textit{conta})) \,|X|\, \textit{dono-conta}))$

- ✓ Esquema_agencia = (nome_agência, cidade_agência, fundos)
- ✓ Esquema_conta = (nome_agência, número_conta, saldo)
- ✓ Esquema_dono_conta= (nome_cliente, número_conta)

- ✓ *Fazer a projeção o mais cedo possível reduz o tamanho da relação intermediária.*

Exemplo de Ordenação de Junções

- ✓ Para todas as relações r_1, r_2 e r_3,

  $(r_1 \text{ |X| } r_2) \text{ |X| } r_3 = r_1 \text{ |X| } (r_2 \text{ |X| } r_3)$

- ✓ Se r_2 |X| r_3 é muito grande e r_1 |X| r_2 é pequena, nós selecionamos

  $(r_1 \text{ |X| } r_2) \text{ |X| } r_3$

- ✓ Assim é possível calcular e armazenar a menor relação temporária.

Exemplo de Ordenação de Junções

$$\Pi_{\textit{nome-cliente}}((\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}}(\textit{agência})) \mid\!X\!\mid (\text{conta} \mid\!X\!\mid \textit{dono-conta}))$$

- ✓ Poderíamos calcular conta|*X*| *dono-conta* primeiro, e juntar o resultado com
  $\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}}(\textit{agência})$

  mas conta|*X*|*dono-conta* é provavelmente uma relação grande.

- ✓ É mais provável que somente uma pequena fração dos clientes do banco têm contas em agências localizadas em Brooklyn.

- ✓ Então, o melhor é calcular primeiro:
  $\sigma_{\textit{cidade-agência} = \text{“Brooklyn”}}(\textit{agência}) \mid\!X\!\mid \text{conta}$

Enumeração de Expressões Equivalentes

- ✓ Otimizadores de consulta usam regras de equivalência para gerar sistematicamente expressões equivalentes para uma expressão de consulta.
- ✓ Passos para gerar todas as expressões equivalentes:

```
Repita
    Aplicar todas as regras de equivalência possíveis sobre toda
      expressão achada até o momento
    Adicionar expressões geradas ao conjunto de expressões
      equivalentes
Até que nenhuma nova expressão  equivalente possa ser gerada
```

- ✓ Qual é o problema?

Enumeração de Expressões Equivalentes

- ✓ Otimizadores de consulta usam regras de equivalência para gerar sistematicamente expressões equivalentes para uma expressão de consulta.
- ✓ Passos para gerar todas as expressões equivalentes:

```
Repita
    Aplicar todas as regras de equivalência possíveis sobre toda
      expressão achada até o momento
    Adicionar expressões geradas ao conjunto de expressões
      equivalentes
Até que nenhuma nova expressão  equivalente possa ser gerada
```

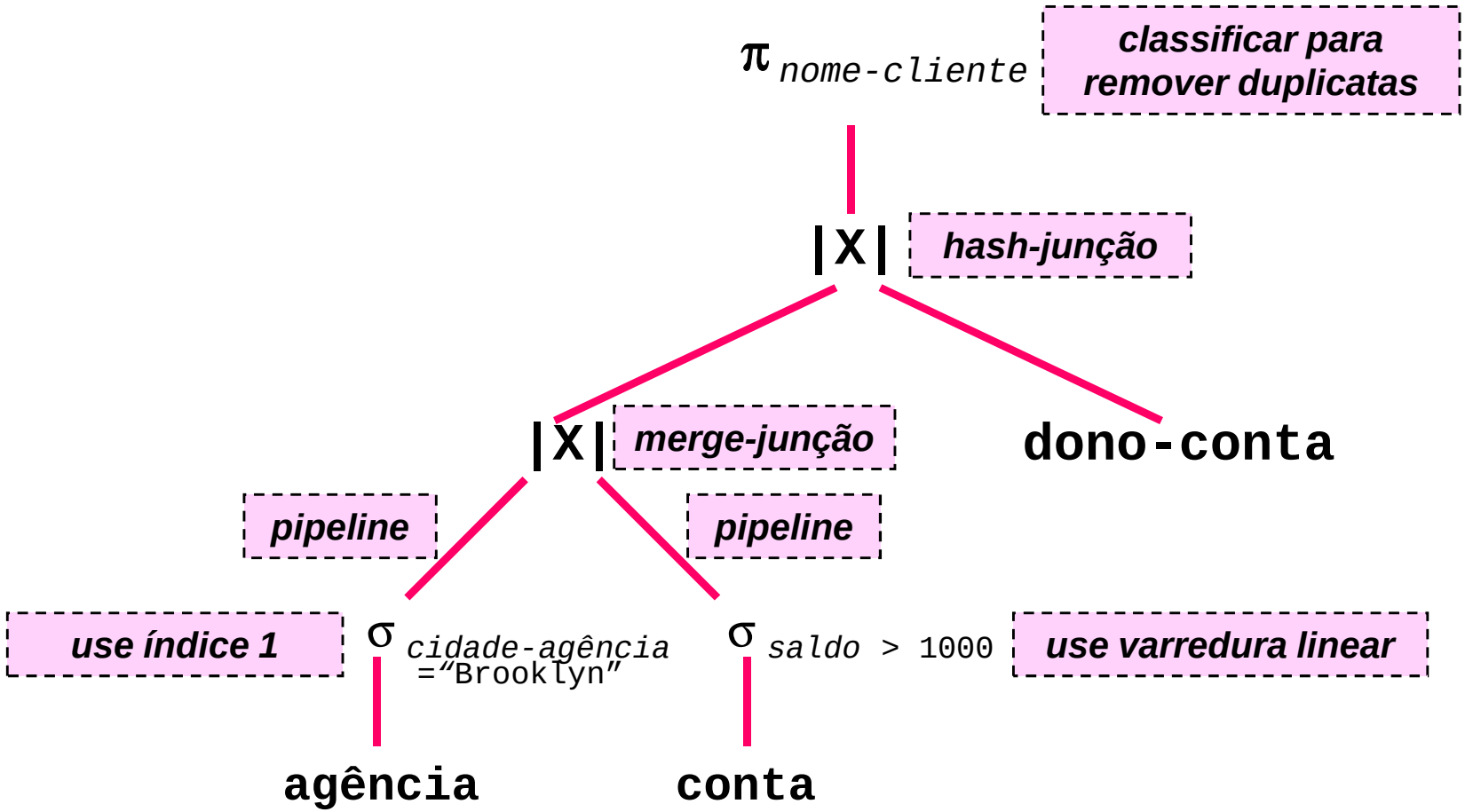
- ✓ **Abordagem muito custosa em espaço e tempo**
- ✓ Duas outras abordagens:
  - Geração do plano otimizado baseado nas regras de transformação
  - Abordagem especial para consultas com somente seleções, projeções e junções

Escolha de planos de avaliação

- ✓ Como um plano de avaliação é escolhido?

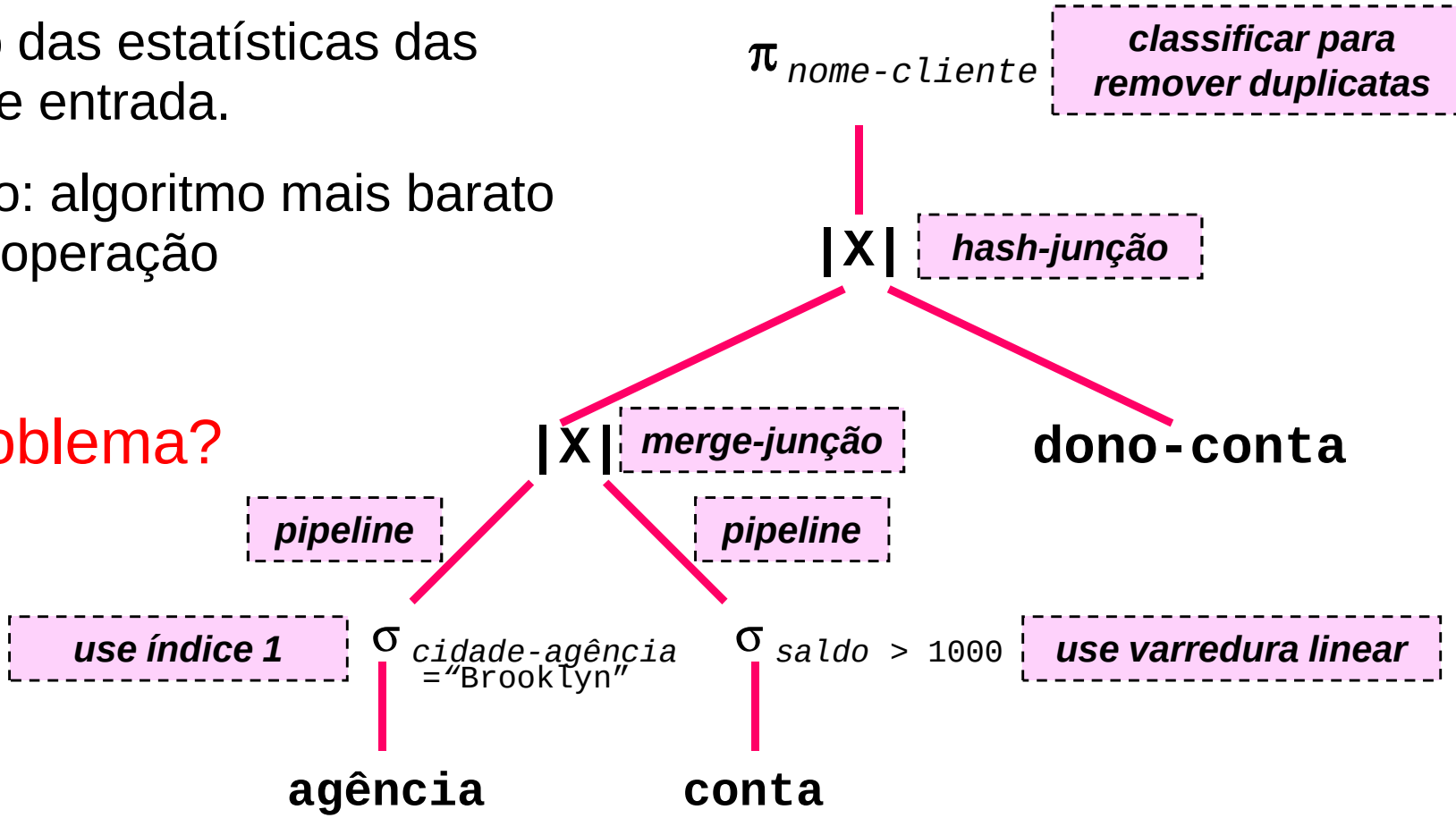
Escolha de planos de avaliação

✓ Como um plano de avaliação é escolhido?



Escolha de planos de avaliação

- ✓ Como um plano de avaliação é escolhido?
  - Em função das estatísticas das relações de entrada.
  - Uma opção: algoritmo mais barato para cada operação
- ✓ Qual é o problema?



Escolha de planos de avaliação

- ✓ Como um plano de avaliação é escolhido?
  - Em função das estatísticas das relações de entrada.
  - Uma opção: algoritmo mais barato para cada operação

- ✓ Qual é o problema?
  - Saída de uma operação pode não ser a melhor opção para a continuidade do plano
  - Ex: merge-junção (mais cara) e hash-junção (mais barata)

Escolha dos Planos de Avaliação

- ✓ Escolher o algoritmo mais barato para cada uma das operações pode não conduzir ao melhor plano total.
  - A junção merge pode ser mais custosa que a junção hash, mas pode fornecer uma saída ordenada que reduz o custo das operações seguintes.
  - A junção de laços aninhados pode fornecer facilidades para o *pipeline*

- ✓ Otimizadores de consultas práticos incorporam elementos das seguintes abordagens:
  1. Busca todos os planos e escolhe o melhor usando o custo.
  2. Usa **heurísticas** para escolher um plano.

Otimização Baseada no Custo

- ✓ Gera faixa de planos de avaliação a partir de uma determinada consulta usando regras de equivalênica
- ✓ Escolhe o de menor custo
- ✓ Problema?

Otimização Baseada no Custo

- ✓ Gera faixa de planos de avaliação a partir de uma determinada consulta usando regras de equivalênica
- ✓ Escolhe o de menor custo
- ✓ Problema?
  - Quantidade de planos pode ser imensa

Otimização Baseada no Custo

- ✓ Exemplo: achar a melhor ordenação de junções para

$$r_1 \,|X|\, r_2 \,|X|\, \ldots \,|X|\, r_n$$

- ✓ Quantidade de ordens de junção diferentes: $\frac{(2(n-1))!}{(n-1)!}$
  - Se n = 5 ➔ 1.680 ordens de junção de diferentes
  - Se n = 7 ➔ 665.280 ordens de junção de diferentes
- ✓ n = 10 ➔ número é maior que 17,6 bilhões!

Programação Dinâmica

- ✓ Método para a construção de algoritmos para a resolução de problemas computacionais, em especial os de otimização combinatória.
- ✓ Aplicável a problemas no qual a solução ótima pode ser computada a partir da solução ótima previamente calculada e memorizada
- ✓ Evita recálculo de outros subproblemas que, sobrepostos, compõem o problema original.

Programação Dinâmica na Otimização

- ✓ Ideia: armazenar resultados de cálculos para reutilizá-los

- ✓ Exemplo:
  - Calcular $(r_1 \,|X|\, r_2 \,|X|\, r_3) \;\; |X|\, r_4 \,|X|\, r_5$
  - 12 ordens de junção para $(r_1 \,|X|\, r_2 \,|X|\, r_3)$
  - Uma vez conhecida a melhor combinação, basta usá-la para o resultado com $|X|\, r_4 \,|X|\, r_5$
  - Em vez de 144 cálculos, executa-se 12 + 12 cálculos

Programação Dinâmica na Otimização

- ✓ Para achar a melhor árvore de junções para um conjunto de *n* relações:
  - Achar o melhor plano para um conjunto *S* de *n* relações, considerando todos os planos possíveis da forma:

    S_1 |X| (S – S_1) onde S_1 é qualquer subconjunto não vazio de *S*.
  - Recursivamente calcular os custos para juntar subconjuntos de S para achar o custo de cada plano. Escolher o mais barato das 2^n – 1 alternativas.
  - Quando o plano para qualquer subconjunto é calculado, armazenar este e reusá-lo quando for requerido de novo, em vez de calcular novamente.

Algoritmo de Otimização da Ordenação de Junções

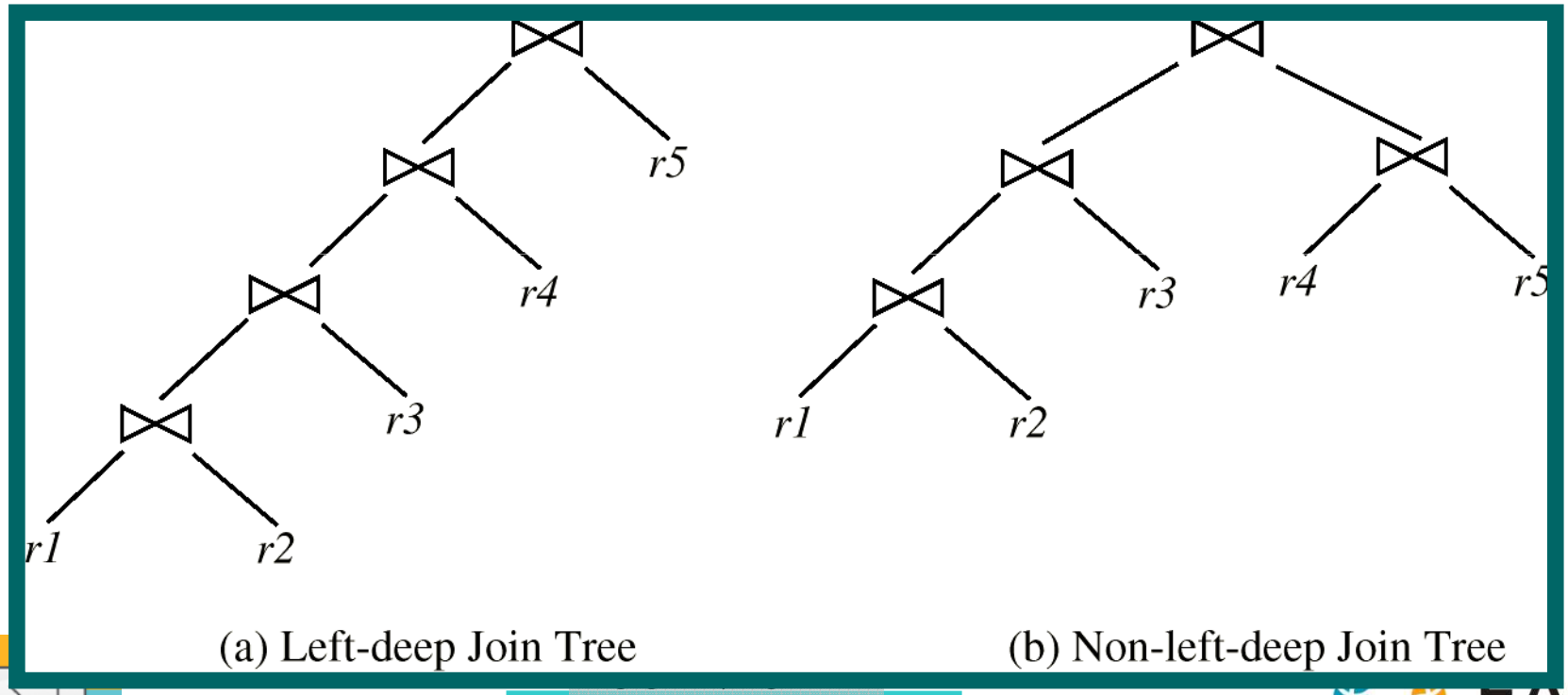
procedure achamelhorplano(*S*)
    if (*melhorplano*[*S*].*custo* ≠ ∞)
        return *melhorplano*[*S*]
    // else *melhorplano*[*S*] não foi calculado antes, calcule-o agora
    if (*S* contém somente 1 relação)
        faça *melhorplano*[*S*].*plano* e *melhorplano*[*S*].*custo* baseado na melhor forma de acessar *S* /* *Usando seleções sobre S e indices sobre S* */
    else for each sub-conjunto não-vazio *S*1 de *S* tal que *S*1 ≠ *S*
        P1= achamelhorplano(*S*1)
        P2= achamelhorplano(*S* - *S*1)
        A = melhor algoritmo para juntar resultados de *P*1 e *P*2
        custo = *P*1.*custo* + *P*2.*custo* + custo de *A*
        if *custo* < *melhorplano*[*S*]. custo
            melhorplano[*S*]. custo = custo
            melhorplano[*S*]. *plano n* = "execute *P*1. *plano*; execute *P*2. *plano*;
                e faça join dos resultados de *P*1 e *P*2 usando *A*"
    return *melhorplano*[*S*]

Custo de Otimização

- ✓ Com programação dinâmica a complexidade em tempo da otimização com árvores é $O(3^n)$.
  - – Com n = 10, este número ≈ 59000 no lugar de 17,6 bilhões
- ✓ Complexidade em espaço: 2^n

Árvores de junção em profundidade pela esquerda

- ✓ Nas árvores de junção em profundidade pela esquerda, a entrada do lado direito para cada junção é uma relação, não o resultado de uma junção intermediária.



Custo de Otimização

- ✓ Para achar a melhor árvore de junções em profundidade pela esquerda para um conjunto de *n* relações:
  - Considere *n* alternativas com uma relação como entrada do lado direito e as outras relações como entradas do lado esquerdo
  - Modificar o algoritmo de otimização:
    - Substitua "**para cada** sub-conjunto não-vazio *S*1 de *S* tal que *S*1 ≠ *S*"
    - Por: **para cada** relação r em S

      seja S1 = S – r .

Custo de Otimização

- ✓ Se somente árvores em profundidade pela esquerda são consideradas, a complexidade em tempo para achar a melhor junção é da ordem $O(n\ 2^n)$
  - complexidade em espaço permanece em $O(2^n)$
- ✓ Otimização baseada no custo é cara, mas útil para consultas sobre grandes conjuntos de dados (consultas típicas têm n pequeno, geralmente < 10)

Otimização Heurística

- ✓ Muitas vezes calcular o custo é muito caro
- ✓ Otimização heurística: usa regras heurísticas para transformar consultas, independentemente do custo
- ✓ Exemplos:
  - Execute operações de seleção assim que possível
  - Execute projeções antes de outras operações

Passos do algoritmo típico de Otimização Heurística

1. Decompor as seleções conjuntivas em uma sequência de operações de seleção simples (regra de equivalência 1).
2. Mover as operações de seleção para a parte inferior da árvore de consultas para poder executá-las o mais cedo possível (regras de equiv. 2, 7a, 7b, 10).
3. Executar primeiro as operações de seleção e junção que produzirão as relações de menor tamanho. Usar associatividade da operação |X| para reorganizar a árvore de tal forma que as relações dos nós folha com seleções mais restritivas sejam executadas antes (regra de equiv. 6 e 10).
4. Trocar as operações de produto cartesiano seguidas por uma condição de seleção por operações de junção (regra de equiv. 4a).
5. Decompor e mover para baixo da árvore o quanto possível as listas de atributos de projeção, criando novas projeções onde seja necessário (regras de equiv. 3, 8a, 8b, 11).
6. Identificar as sub-árvores cujas operações podem ser colocadas em *pipeline*, e executá-las usando *pipelining*.

Estrutura dos Otimizadores de Consultas

- ✓ Maioria dos otimizadores dos SGBDs combina elementos de custo com elementos de heurísticas.
- ✓ Mesmo com o uso de heurísticas, a otimização baseada no custo da consulta impõe um *overhead* considerável.
- ✓ Este custo é mais do que compensado pela economia no tempo de execução da consulta, especificamente pela redução do número de acessos de discos lentos.
- ✓ Pesquisa: qual abordagem otimizador do Oracle usa? E o PostGreSQL?

Estrutura dos Otimizadores de Consultas

- ✓ Maioria dos otimizadores dos SGBDs combina elementos de custo com elementos de heurísticas.
- ✓ Mesmo com o uso de heurísticas, a otimização baseada no custo da consulta impõe um *overhead* considerável.
- ✓ Este custo é mais do que compensado pela economia no tempo de execução da consulta, especificamente pela redução do número de acessos de discos lentos.
- ✓ Pesquisa: qual abordagem otimizador do Oracle usa? E o PostGreSQL?

  Material adicional:
  http://www.sqlmagazine.com.br/artigos/oracle/07_OtimizadorOracle-I.asp
  http://dspace.c3sl.ufpr.br/dspace/handle/1884/7868
  http://pgdocptbr.sourceforge.net/pg80/planner-optimizer.html

Informação estatística para estimação do custo

- ✓ n_r: número de tuplas na relação *r.*
- ✓ b_r: número de blocos contendo as tuplas de *r.*
- ✓ s_r: tamanho de uma tupla de *r.*
- ✓ f_r: fator de bloqueio de *r* → o número de tuplas de *r* que cabem num bloco.
- ✓ *V(A, r):* número de valores diferentes que aparecem em *r* para o atributo *A; o mesmo que o tamanho de* $\Pi_A(r)$.
- ✓ Se as tuplas de *r* são armazenadas fisicamente juntas num arquivo, então:

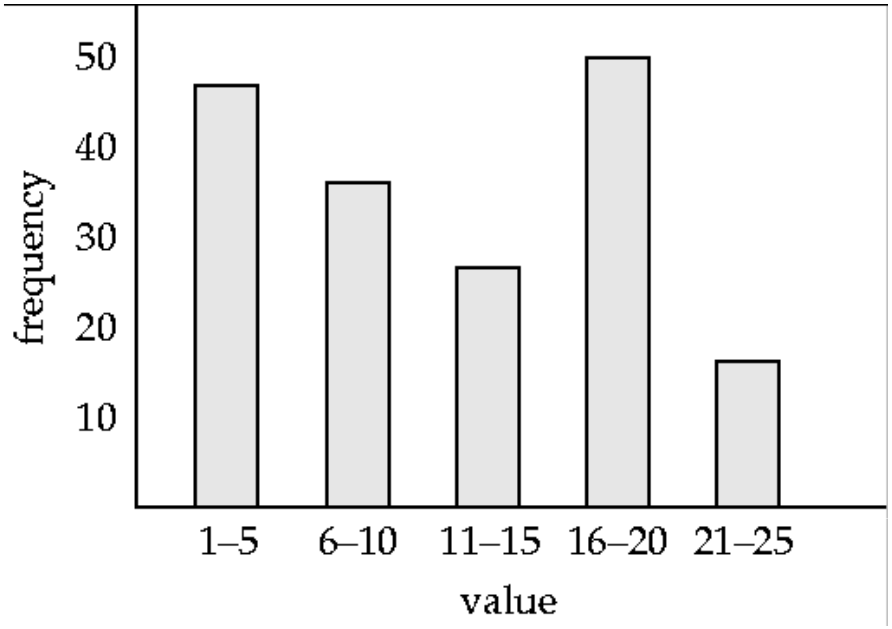
$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

Informação estatística para estimação do custo

- ✓ f_i: número médio de nós internos do índice *i*, para índices estruturados em árvores tais como árvores B+.
- ✓ HT_i: número de níveis no índice *i* — a altura de *i.*
  - Para um índice em árvore balanceado (por exemplo árvore B+) sobre o atributo *A* da relação *r*, $HT_i = \log_{fi}(V(A,r))$.
  - Para um índice de hash, HT_i *é* 1.
  - LB_i: número de blocos do mais baixo nível do índice *i (*o número de blocos no nível das folhas do índice).

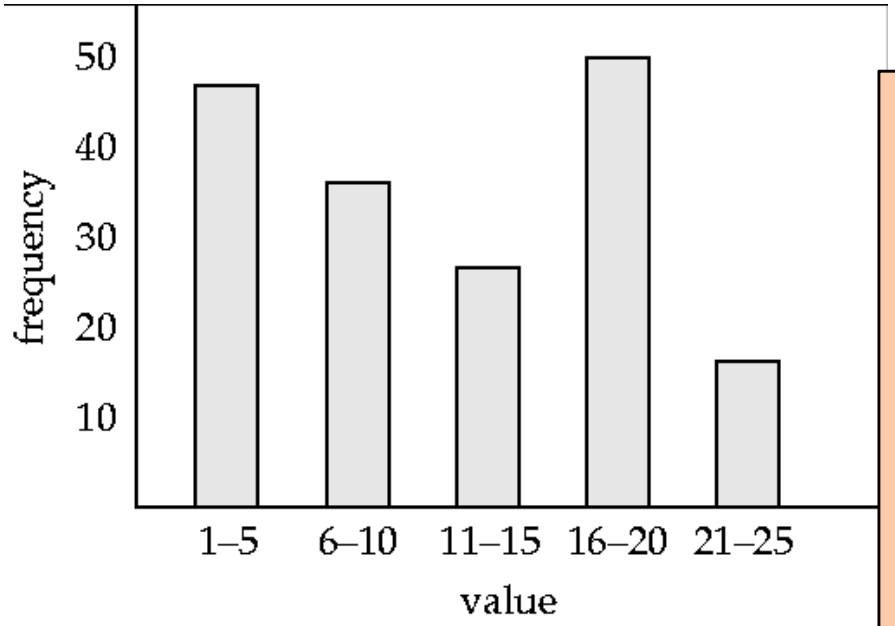
Informação estatística para estimação do custo – histograma dos atributos

- ✓ Distribuição de valores para cada atributo como um histograma. Caso contrário~, se assume distribuição uniforme de valores.
- ✓ Exemplo: histograma sobre o atributo *idade* da relação *pessoa*



Informação estatística para estimação do custo – histograma dos atributos

- ✓ Distribuição de valores para cada atributo como um histograma. Caso contrário~, se assume distribuição uniforme de valores.
- ✓ Exemplo: histograma sobre o atributo *idade* da relação *pessoa*



- Histogramas de largura igual (intervalo de valores com mesmo tamanho)
- Histogramas de profundidade igual (ajusta intervalos para ter frequência igual em todos)

Otimização das subconsultas aninhadas

- ✓ SQL conceitualmente trata subconsultas aninhadas na cláusula *where* como funções que recebem parâmetros e retornam ou um único valor ou um conjunto de valores
- ✓ Parâmetros são variáveis de consulta de nível externo que são usadas na subconsulta aninhada ➔ variáveis são chamadas de **variáveis de correlação**
- ✓ Exemplo:

```
select nome-cliente
from credor
where exists (select *
              from dono-conta
              where dono-conta.nome-cliente =
                                    credor. nome-cliente)
```

Otimização das subconsultas aninhadas

- ✓ Exemplo:

```
select nome-cliente
from credor
where exists (select *
                from dono-conta
                where dono-conta.nome-cliente =
                                        credor. nome-cliente)
```

- ✓ Conceitualmente, subconsulta aninhada é executada **uma vez para cada tupla no produto cartesiano** gerado pela cláusula **from** externa
  - avaliação é chamada **avaliação correlacionada**

Otimização das subconsultas aninhadas

- ✓ Avaliação correlacionada pode ser muito ineficiente já que
  - um grande número de chamadas podem ser feitas à consulta aninhada
  - Pode resultar em E/S de disco aleatórias desnecessárias
- ✓ Otimizadores SQL tentam transformar subconsultas aninhadas em junções onde for possível, facilitando o uso eficiente das técnicas de junção
- ✓ Exemplo:
  - consulta aninhada anterior pode ser reescrita como:
    select *nome-cliente*
    from *credor, dono-conta*
    where *dono-conta.nome-cliente = credor. nome-cliente*
  - Nota: esta consulta não trabalha corretamente com duplicatas, mas pode ser modificada para fazer isso

Otimização das subconsultas aninhadas

- ✓ Antes:

  select *nome-cliente*
  from *credor*
  where exists (**select** *
  from *dono-conta*
  where *dono-conta.nome-cliente =*
  credor. nome-cliente)

- ✓ Depois:
  - consulta aninhada anterior pode ser reescrita como:

    select *nome-cliente*
    from *credor, dono-conta*
    where *dono-conta.nome-cliente = credor. nome-cliente*

- ✓ Se não for possível passar diretamente as relações da consulta aninhada para a cláusula from da consulta externa:
  - mantém as subconsultas
  - uma relação temporária é criada no lugar e usada no corpo da consulta externa

Otimização das subconsultas aninhadas

- ✓ O processo de trocar uma consulta aninhada por uma consulta com uma junção (possivelmente com uma relação temporária) é chamado **descorrelação**.
- ✓ Descorrelação é mais complexa quando:
  - subconsulta aninhada usa agregação
  - resultado da subconsulta aninhada é usado para teste de igualdade
  - condição ligando a subconsulta aninhada à outra consulta é **not exists**

Bibliografia

- ✓ Silberschatz, A.; Korth, H.F.; Sudarshan, S. “Sistema de Banco de Dados”, 5a. edição, Makron Books, 2006 (capítulo 14)
- ✓ Elmasri, R.; Navathe, S.B. Fundamentals of Database Systems, Benjamin Cummings, 3a edição, 2000 (capítulo 15).
- ✓ Date, C. J. Introdução a Sistemas de Banco de Dados - Tradução da 7ª Edição, 2000 - Editora Campus (capítulo 17).

Exercícios

- ✓ Silberschatz, A.; Korth, H.F.; Sudarshan, S. “Sistema de Banco de Dados”, 5a. edição, Makron Books, 2006. Capítulo 14 – Exercícios de 1 a 19
- ✓ Elmasri, R.; Navathe, S.B. Fundamentals of Database Systems, Benjamin Cummings, 3a edição, 2000. Capítulo 15 – Exercícios de 1 a 11, 13 a 16, 22
- ✓ Date, C. J. Introdução a Sistemas de Banco de Dados - Tradução da 7ª Edição, 2000 - Editora Campus. Capítulo 17 – Exercícios de 1 a 11, 13 a 15

ACH2025

Laboratório de Bases de Dados

Aula 15

Processamento de Consultas – Parte 2
Otimização

Professora:

- **Fátima L. S. Nunes**