



CEFET/RJ



OTIMIZAÇÃO DE CONSULTAS

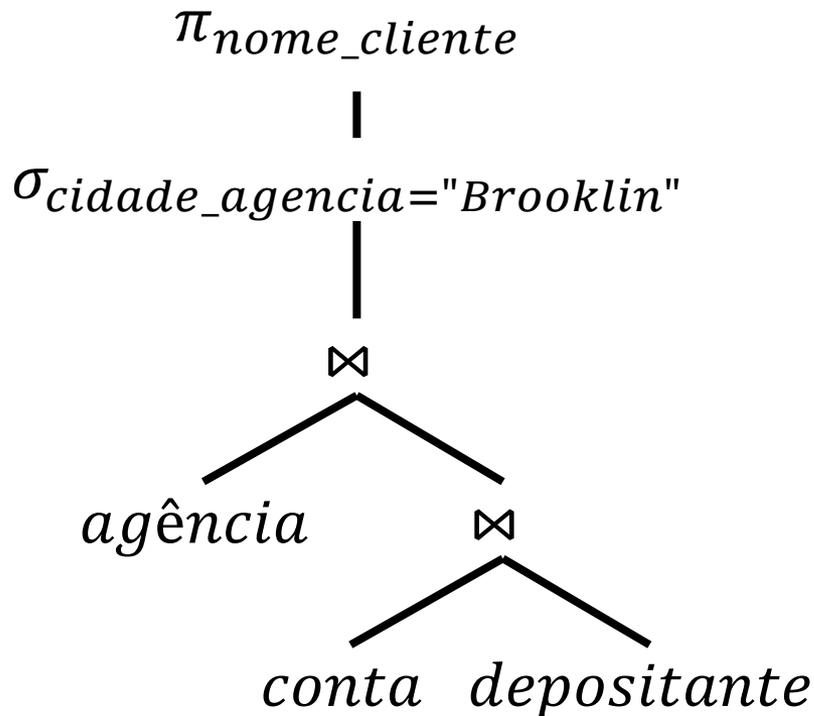
Eduardo Ogasawara
eogasawara@ieee.org
<https://eic.cefet-rj.br/~eogasawara>

Introdução

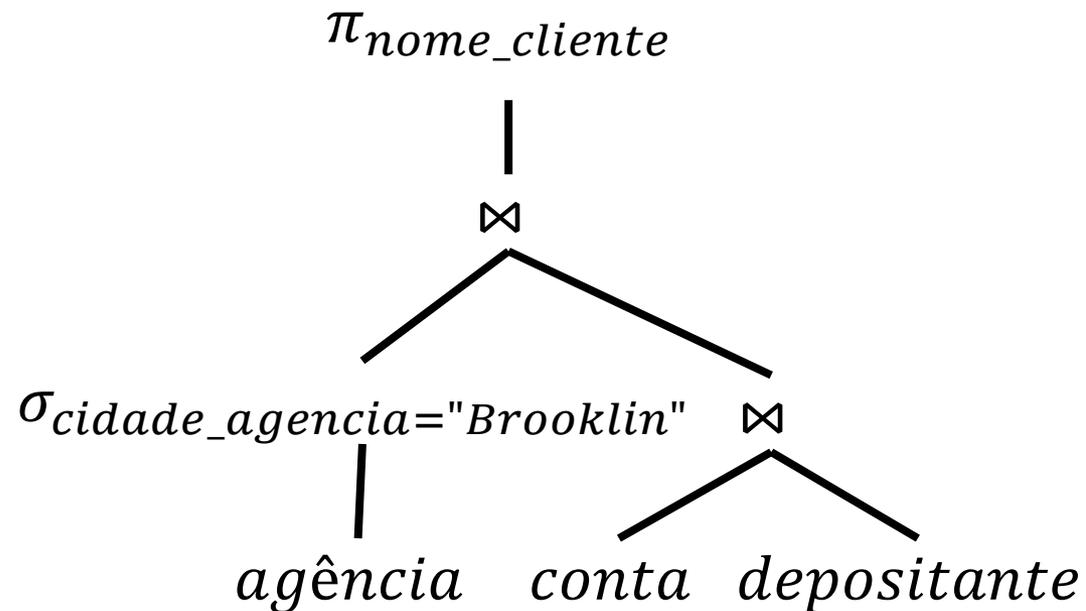
- Modos alternativos de avaliar determinada consulta
 - Expressões equivalentes
 - Diferentes algoritmos para cada operação
- A diferença de custo entre uma maneira boa e ruim para avaliar uma consulta pode ser enorme
 - Exemplo: realizar um $R \times S$ seguido por uma seleção $\sigma_{R.A = S.B}$ é muito mais lento do que realizar uma junção na mesma condição
- Necessidade de estimar o custo das operações
 - Depende criticamente de informações estatísticas sobre relações que o banco de dados precisa manter
 - Exemplo: número de tuplas, número de valores distintos para atributos de junção
 - Precisa estimar estatísticas para resultados intermediários, para calcular o custo de expressões complexas

Comparações de árvores de processamento de consultas

- Relações geradas por duas expressões equivalentes têm o mesmo conjunto de atributos e contêm o mesmo conjunto de tuplas, embora seus atributos possam ser ordenados de modo diferente



(a) árvore da expressão inicial



(b) árvore transformada

Otimização baseada em custo

- A geração de planos de avaliação de consulta para uma expressão envolve várias etapas:
 - Gerar expressões logicamente equivalentes usando regras de equivalência para transformar uma expressão em uma equivalente
 - Anotar expressões resultantes para obter planos de consulta alternativos
 - Escolher o plano mais barato com base no custo estimado
- O processo geral é denominado otimização baseada em custo

Transformação de expressões relacionais

- Duas expressões da álgebra relacional são consideradas equivalentes se em cada instância de banco de dados válida as duas expressões gerarem o mesmo conjunto de tuplas
 - Nota: a ordem das tuplas é irrelevante
- Em SQL, entradas e saídas são multiconjuntos de tuplas
 - Duas expressões na versão multiconjunto da álgebra relacional são consideradas equivalentes se em cada instância de banco de dados válida as duas expressões gerarem o mesmo multiconjunto de tuplas
- Uma regra de equivalência diz que expressões de duas formas são equivalentes
 - Pode substituir a expressão da primeira forma pela segunda, ou vice-versa

Regras de equivalência

- Operações de seleção conjuntiva podem ser desmembradas em uma sequência de seleções individuais

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

- Operações de seleção são comutativas

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

- Somente a última em uma sequência de operações de projeção é necessária, e as outras podem ser omitidas (em cascata)

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

- As seleções em produtos cartesianos podem ser combinados em junções teta

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

$$\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$$

Regras de equivalência

- Operações de junção teta (e junções naturais) são comutativas

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

- Operações de junção natural são associativas

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

- Operações de junção teta são associativas

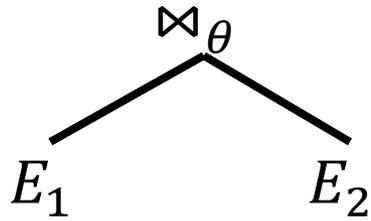
$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

onde θ_2 envolve atributos apenas de E_2 e E_3

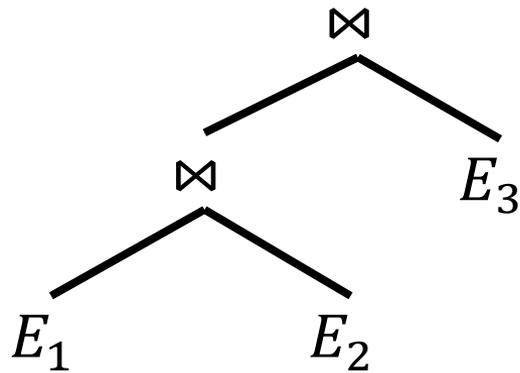
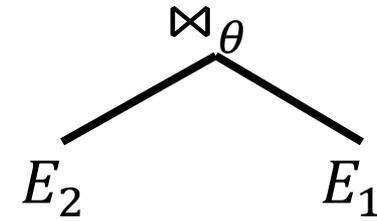
Regras de equivalência

- A operação de seleção tem distribuição com a junção teta sob as duas condições a seguir:
 - Quando todos os atributos em θ_0 envolvem apenas os atributos de uma das expressões (E_1) sendo juntadas
$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$
 - Quando θ_1 envolve apenas os atributos de E_1 e θ_2 envolve apenas os atributos de E_2
$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = \sigma_{\theta_1}(E_1) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$
- Essas transformações são conhecidas como push-down selection

Representação das regras de equivalência



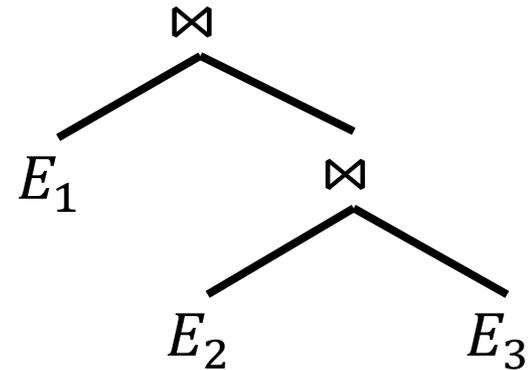
comutatividade



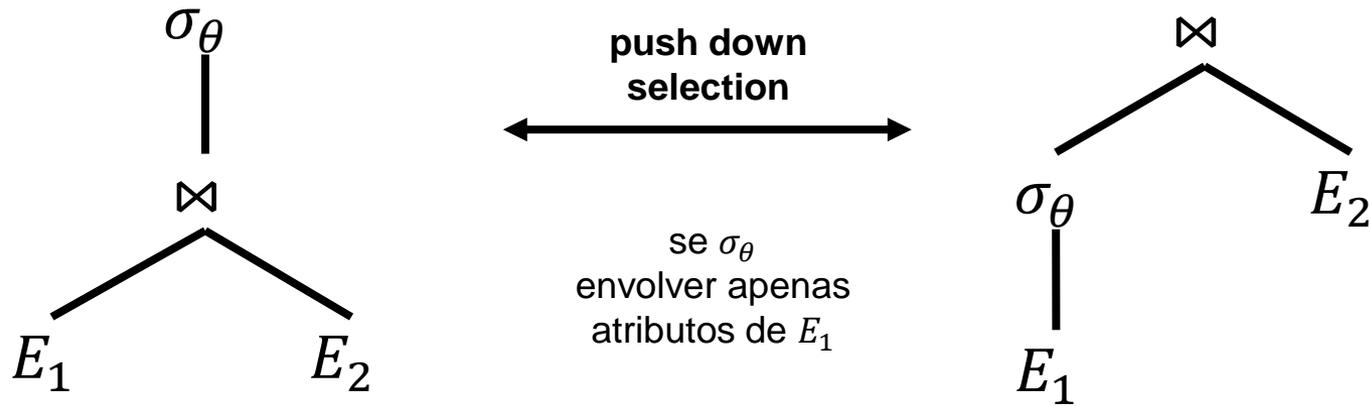
associatividade



se E_2 e E_3
tiverem atributos
comuns para
junção



Representação das regras de equivalência



Regras de equivalência

- A operação de projeção distribui pela operação de junção teta da seguinte forma:
- se π envolver apenas atributos de $L_1 \cup L_2$:

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \pi_{L_1}(E_1) \bowtie_{\theta} \pi_{L_2}(E_2)$$

Regras de equivalência

- A operação de projeção tem cascata na junção teta
- Considere uma junção $E_1 \bowtie_{\theta} E_2$
 - Considere L_1 e L_2 conjuntos de atributos de E_1 e E_2 , respectivamente
 - Considere L_3 um conjunto de atributos de E_1 que estão envolvidos na condição de junção θ , mas não estão em $L_1 \cup L_2$
 - Considere L_4 um conjunto de atributos de E_2 que estão envolvidos na condição de junção θ , mas não estão em $L_1 \cup L_2$

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \pi_{L_1 \cup L_2}(\pi_{L_1 \cup L_3}(E_1) \bowtie_{\theta} \pi_{L_2 \cup L_4}(E_2))$$

Regras de equivalência

- A união e interseção de conjuntos são comutativas

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

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

(a diferente de conjunto não é comutativa)

- A união e a interseção de conjuntos 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

- A operação de seleção tem distribuição por \cup , \cap e $-$

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

e semelhantemente para \cup e \cap no lugar de $-$

- Também:

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

e semelhantemente para \cap no lugar de $-$, mas não para \cup

- A operação de projeção tem distribuição pela união

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

Exemplo de transformação

- Encontre os nomes de todos os clientes que têm uma conta em alguma agência localizada em Brooklyn

$\pi_{\text{nomeCliente}}(\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"}}(\text{agencia} \bowtie \text{conta} \bowtie \text{depositante}))$

\equiv

$\pi_{\text{nomeCliente}}(\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"}}(\text{agencia}) \bowtie \text{conta} \bowtie \text{depositante})$

- Reduz a seleção o mais cedo possível reduz o tamanho da relação a ser juntada

Exemplo com múltiplas transformações

- Encontrar os nomes de todos os clientes com uma conta em uma agência do Brooklyn cujo saldo de conta é maior que \$1000

$$\pi_{\text{nomeCliente}} \left(\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"} \wedge \text{saldo} > 1000} (\text{agencia} \bowtie \text{conta} \bowtie \text{depositante}) \right)$$
$$\equiv$$

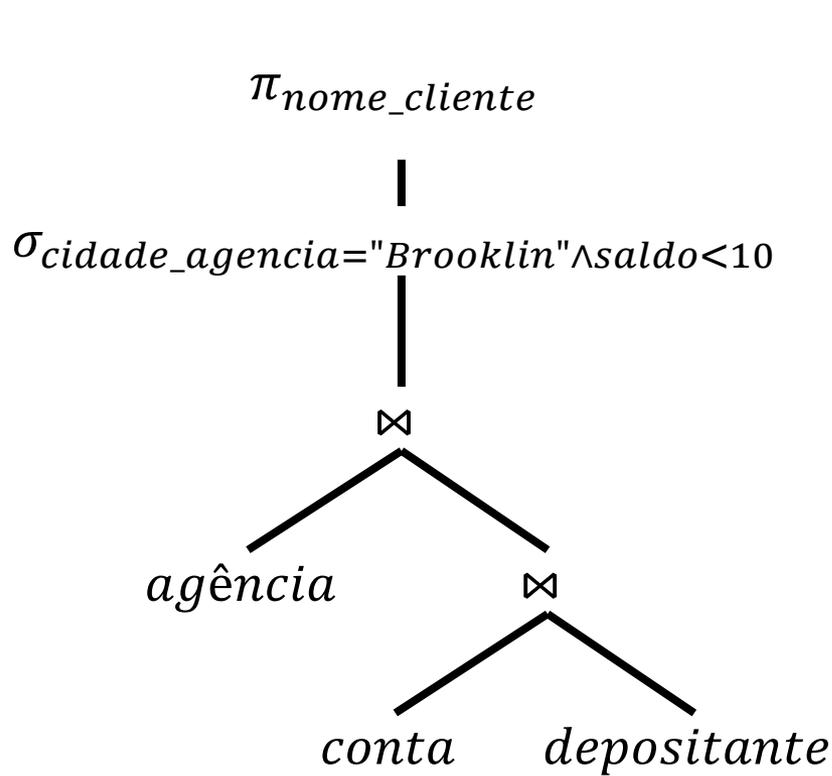
$$\pi_{\text{nomeCliente}} \left(\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"} \wedge \text{saldo} > 1000} (\text{agencia} \bowtie \text{conta}) \right) \bowtie \text{depositante}$$

- A segunda forma oferece uma oportunidade para aplicar a regra “realizar seleções cedo”, resultando na subexpressão

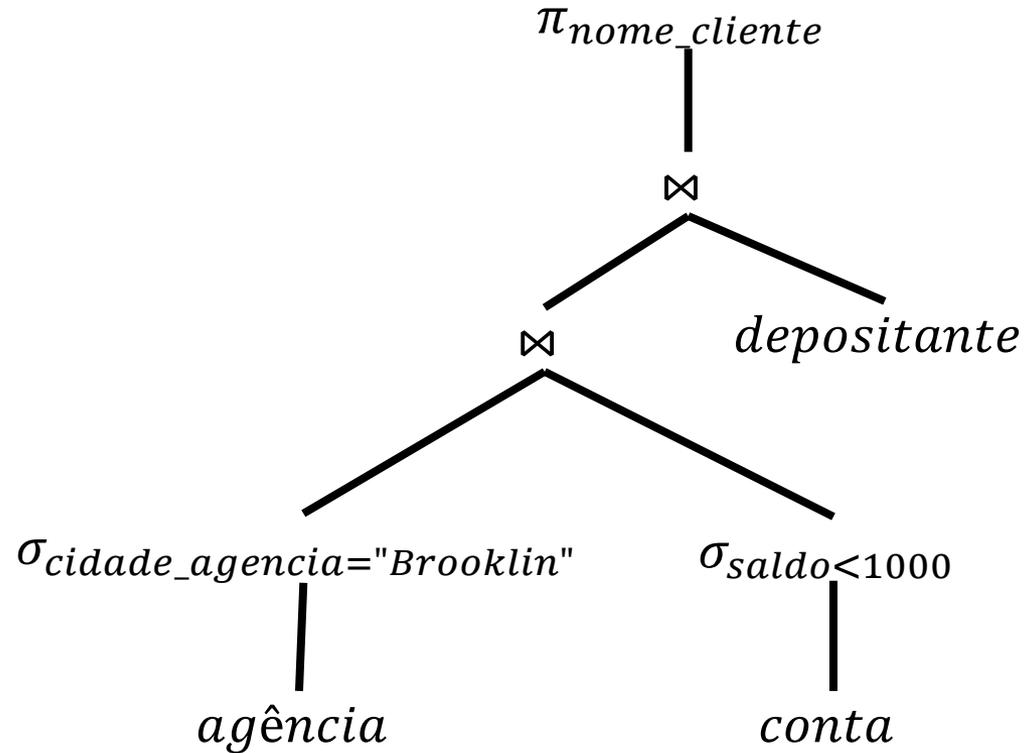
$$\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"} \wedge \text{saldo} > 1000} (\text{agencia} \bowtie \text{conta})$$
$$\equiv$$

$$\sigma_{\text{cidadeAgencia}=\text{"Brooklyn"}} (\text{agencia}) \bowtie \sigma_{\text{saldo} > 1000} (\text{conta})$$

Múltiplas transformações



(a) árvore da expressão inicial



(b) árvore após múltiplas transformações

Exemplo de operação de projeção

- Quando calculamos

$\sigma_{cidadeAgencia="Brooklyn"}(agencia) \bowtie conta$

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

- Empurre projeções usando as regras de equivalência e elimine atributos indesejados dos resultados intermediários para obter:

$\pi_{nomeCliente}(\pi_{numeroConta}(\sigma_{cidadeAgencia="Brooklyn"}(agencia) \bowtie conta) \bowtie depositante)$

- Realize a projeção o mais cedo possível reduz o tamanho da relação intermediária
 - Atenção que os atributos necessários as operações seguintes devem ser preservados

Exemplo de ordenação de junção

- Para todas as relações R_1, R_2, R_3
 - $(R_1 \bowtie R_2) \bowtie R_3 = R_1 \bowtie (R_2 \bowtie R_3)$
 - Se $R_2 \bowtie R_3$ for muito grande e $R_1 \bowtie R_2$ for pequeno, escolhemos $(R_1 \bowtie R_2) \bowtie R_3$ de modo que calculamos e armazenamos uma relação temporária menor

Exemplo de ordenação de junção (cont.)

- Considere a expressão

$\pi_{\text{nomeCliente}}((\textit{depositante} \bowtie \textit{conta}) \bowtie \sigma_{\text{cidadeAgencia}=\textit{Brooklyn}}(\textit{agência}))$

- Poderia calcular $\textit{conta} \bowtie \textit{depositante}$ primeiro e depois o resultado da junção como $\sigma_{\text{cidadeAgencia}=\textit{Brooklyn}}(\textit{agência})$
- Entretanto, $\textit{conta} \bowtie \textit{depositante}$ provavelmente será uma relação grande
- Como é mais provável que somente uma pequena fração dos clientes do banco tenha contas em agências localizadas no Brooklyn, é melhor calcular primeiro

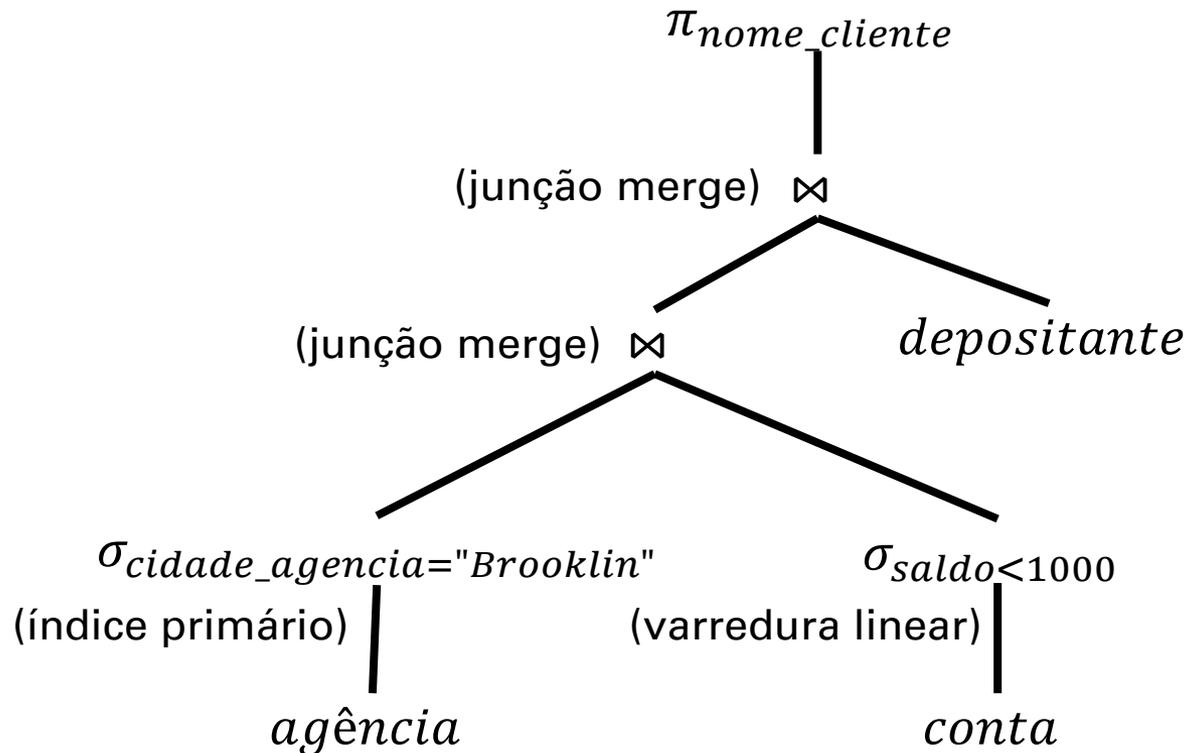
$\pi_{\text{nomeCliente}}((\sigma_{\text{cidadeAgencia}=\textit{Brooklyn}}(\textit{agência}) \bowtie \textit{conta}) \bowtie \textit{depositante})$

Enumeração de expressões equivalentes

- Os otimizadores de consulta usam regras de equivalência para gerar sistematicamente expressões equivalente à expressão indicada
- Conceitualmente, gera-se todas as expressões equivalentes executando repetidamente as transformações até que nenhuma outra expressão seja encontrada:
 - para cada expressão encontrada até aqui, use todas as regras de equivalência aplicáveis, e acrescente expressões recém geradas ao conjunto de expressões encontradas até aqui
- A técnica acima é muito dispendiosa em espaço e tempo

Plano de avaliação

- Um plano de avaliação define exatamente qual algoritmo é usado para cada operação, e como a execução das operações é coordenada



Escolha dos planos de avaliação

- Precisa-se considerar a interação das técnicas de avaliação quando escolher planos de avaliação:
 - a escolha do algoritmo de menor custo para cada operação independentemente pode não gerar o melhor algoritmo geral.
 - Por exemplo:
 - junção merge pode ser mais dispendiosa que a junção de hash, mas pode oferecer uma saída classificada que reduz o custo para um nível de agregação externo
 - a junção de loop aninhado oferece oportunidade para pipelining
- As abordagens adotadas pelos otimizadores de consulta incorporam elementos das duas técnicas gerais a seguir:
 - 1. Pesquisar todos os planos e escolher o melhor plano com base no custo
 - 2. Usar heurística para escolher um plano

Otimização baseada em custo

- Considere encontrar a melhor ordem de junção para
$$R_1 \bowtie R_2 \bowtie R_3 \bowtie \dots \bowtie R_n$$
- Existem $\frac{C_{n-1}^{2(n-1)}}{n}$ ordens de junção diferentes para a expressão acima
 - Com $n = 7$, o número é 665280, com $n = 10$, o número é maior que 176 bilhões!
- Não é preciso gerar todas as ordens de junção.
 - Usando a programação dinâmica, a ordem de junção de menor custo para qualquer subconjunto de $\{R_1, R_2, \dots, R_n\}$ é calculada apenas uma vez e armazenada para uso futuro

Programação dinâmica na otimização

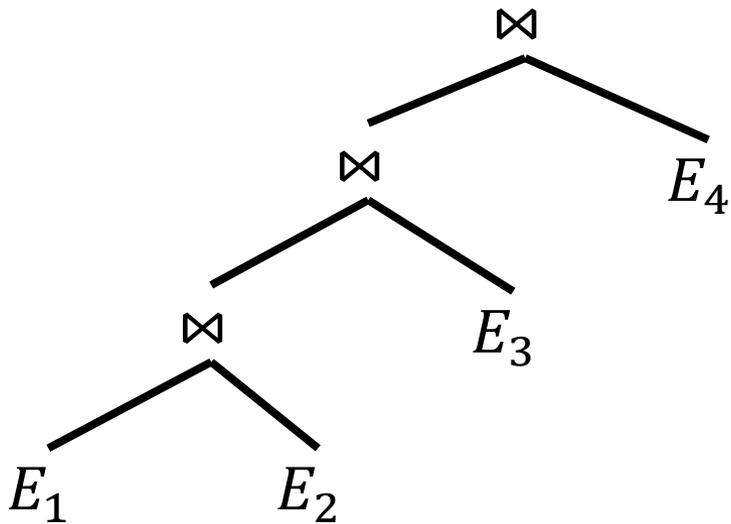
- Para encontrar a melhor árvore de junção para um conjunto de n relações:
 - Para encontrar o melhor plano para um conjunto S de n relações, considere todos os planos possíveis da forma: $S_1 \bowtie (S - S_1)$ onde S_1 é qualquer subconjunto não vazio de S
 - Recursivamente, calcule os custos para juntar subconjuntos de S para encontrar o custo de cada plano. Escolha a mais barata das $2^n - 1$ alternativas
 - Quando o plano para qualquer subconjunto for calculado, armazene-o e reutilize-o quando for necessário novamente, em vez de recalculá-lo
 - Programação dinâmica

Algoritmo de otimização da ordem de junção

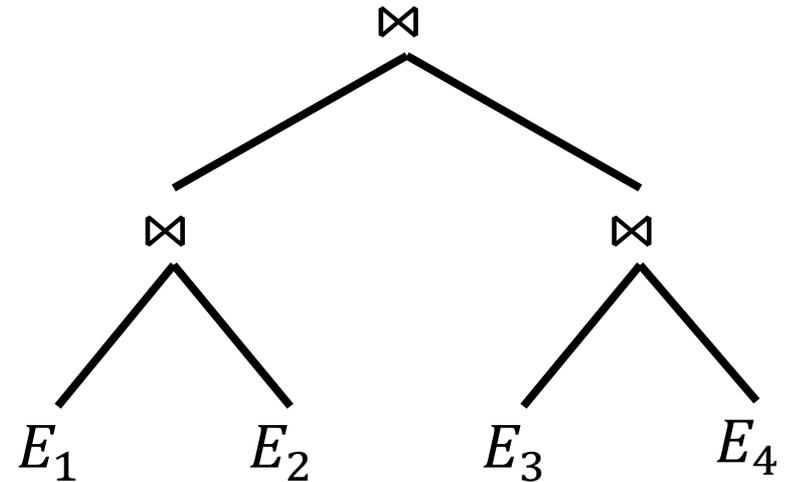
```
procedure acharMelhorPlano(S)
  if (melhorplano[S].custo  $\neq$   $\infty$ )
    return melhorplano[S]
  // Se não estava calculado o melhorplano[S]; calcula-se agora
  for each subconjunto não vazio  $S_1$  de S de modo que  $S_1 \neq S$ 
     $P_1 = \text{acharMelhorPlano}(S_1)$ 
     $P_2 = \text{acharMelhorPlano}(S - S_1)$ 
    A = melhor alg. para junção de 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 =
        "executar  $P_1$ .plano;
        executar  $P_2$ .plano;
        juntar resultados de  $P_1$  e  $P_2$  usando A"
  return melhorplano[S]
```

Árvores de junção esquerda profunda

- Em árvores de junção profunda esquerda, a entrada do lado direito para cada junção é uma relação, e não o resultado de uma junção intermediária



(a) árvore de junção esquerda profunda



(b) árvore de junção

Custo da otimização

- Com o tempo de programação dinâmico, a complexidade da otimização com árvores de muitas folhas é $O(3^n)$
 - Com $n = 10$, esse número é 59000 em vez de 176 bilhões!
- A complexidade de espaço é $O(2^n)$

Custo da otimização

- Para encontrar a melhor árvore de junção profunda esquerda para um conjunto de n relações
 - Considere n alternativas com uma relação como a entrada do lado direito e as outras relações como a entrada do lado esquerdo
 - Usando a ordem de junção de menor custo (calculada e armazenada recursivamente) para cada alternativa no lado esquerdo, escolha a mais barata das n alternativas
- Se apenas árvores profundas esquerdas forem consideradas, a complexidade de tempo para encontrar a melhor ordem de junção é **$O(n2^n)$**
- A complexidade de espaço permanece em **$O(2^n)$**
- A otimização baseada em custo é dispendiosa, mas compensa para consultas sobre datasets grandes
(consultas típicas possuem n pequeno, geralmente < 10)

Ordens interessantes na otimização baseada em custo

- Considere a expressão $(R_1 \bowtie R_2 \bowtie R_3) \bowtie R_4 \bowtie R_5$
- Uma ordem de classificação interessante é uma ordem de classificação as tuplas para as próximas operações
 - O uso da junção merge para calcular $R_1 \bowtie R_2 \bowtie R_3$ pode ser mais dispendioso, mas pode oferecer uma saída classificada em uma ordem interessante para o processamento de R_4 ou R_5
- Não é suficiente encontrar a melhor ordem de junção para cada subconjunto do conjunto de n relações dadas
 - É preciso encontrar a melhor ordem de junção para cada subconjunto, para cada ordem de classificação interessante
 - Extensão simples dos algoritmos de programação dinâmica mais antigos
 - Normalmente, o número de ordens interessantes é muito pequeno e não afeta a complexidade de tempo/espço de modo significativo

Otimização heurística

- A otimização baseada em custo é dispendiosa, mesmo com a programação dinâmica
- Os sistemas podem usar heurísticas para reduzir o número de escolhas que precisam ser feitas com base no custo
- A otimização heurística transforma a árvore de consulta usando um conjunto de regras que normalmente (mas não em todos os casos) melhoram o desempenho da execução:
 - Realizar a seleção cedo (reduz o número de tuplas)
 - Realizar a projeção cedo (reduz o número de atributo)
 - Realizar as operações de seleção e junção mais restritivas antes de outras operações semelhantes
 - Alguns sistemas utilizam apenas heurísticas, outros combinam heurísticas com otimização parcial baseada em custo.

Etapas na otimização heurística típica

- Desconectar seleções conjuntivas em uma sequência de operações de seleção exclusivas
- Mover operações de seleção para baixo na árvore de consulta, para executar o mais cedo possível (push-down selection)
- Executar primeiro as operações de seleção e junção que produzirão as menores relações
- Substituir operações de produto cartesiano que são seguidas por uma condição de seleção por operações de junção
- Desconstruir e mover para o mais baixo possível na árvore as listas de atributos de projeção, criando novas projeções onde for necessário
- Identificar as sub-árvores cujas operações podem ser canalizadas, e executá-las usando pipelining

Referências

