CEFET/RJ

Otimização de Consultas

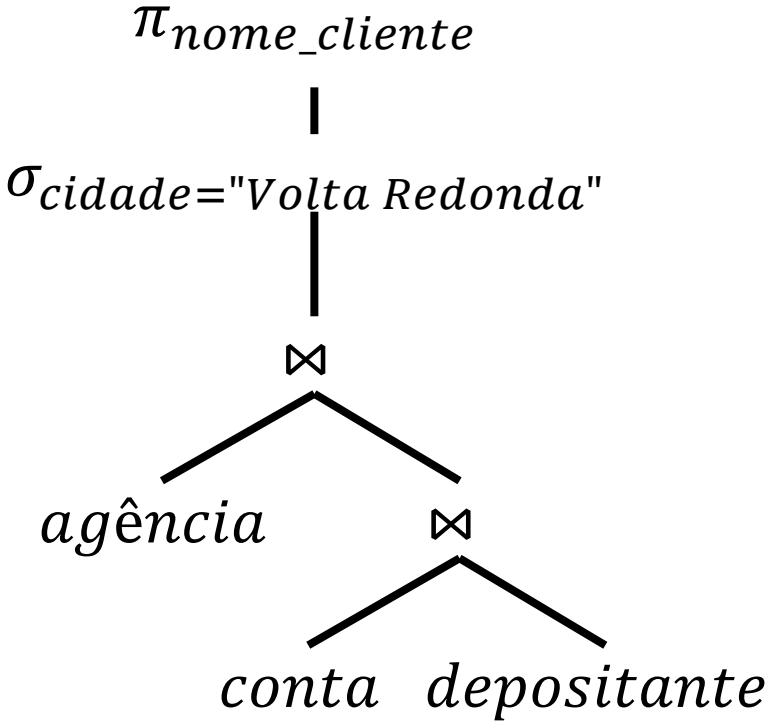
Eduardo Ogasawara
eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

Introdução

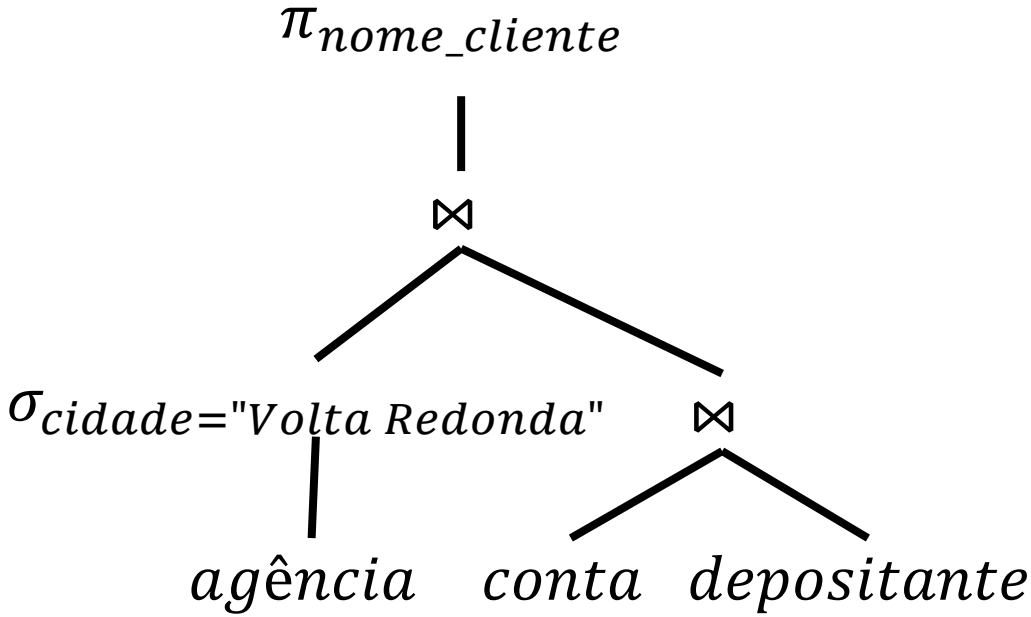
- Modos alternativos de avaliar determinada consulta
  - Expressões equivalentes
  - Diferentes algoritmos para cada operação
- A diferença de custo entre uma maneira boa e ruim para avaliar uma consulta pode ser enorme
  - Exemplo: realizar um $R \times S$ seguido por uma seleção $\boldsymbol{\sigma}_{R.A = S.B}$ 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

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

- Operações de seleção são comutativas

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

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

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

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

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

Regras de equivalência

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

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

- Operações de junção natural são associativas

$$\boldsymbol{(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)}$$

- Operações de junção teta são associativas

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

  onde $\boldsymbol{\theta_2}$ envolve atributos apenas de $\boldsymbol{E_2}$ e $\boldsymbol{E_3}$

Regras de equivalência

- A operação de seleção tem distribuição com a junção teta sob as duas condições a seguir:
  - Quando todos os atributos em $\boldsymbol{\theta_0}$ envolvem apenas os atributos de uma das expressões ($\boldsymbol{E_1}$) sendo juntadas

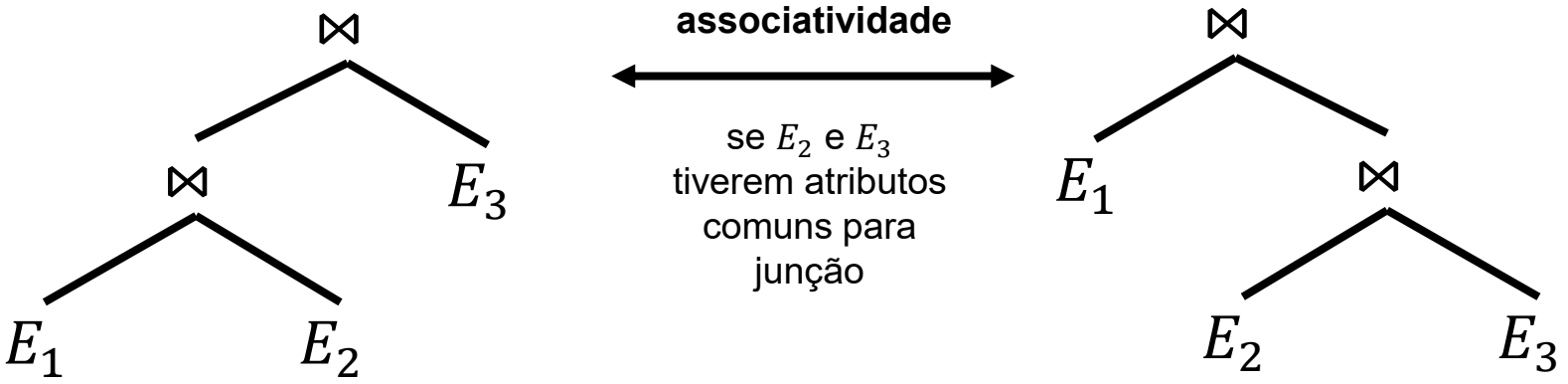
$$\boldsymbol{\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2}$$

  - Quando $\boldsymbol{\theta_1}$ envolve apenas os atributos de $\boldsymbol{E_1}$ e $\boldsymbol{\theta_2}$ envolve apenas os atributos de $\boldsymbol{E_2}$

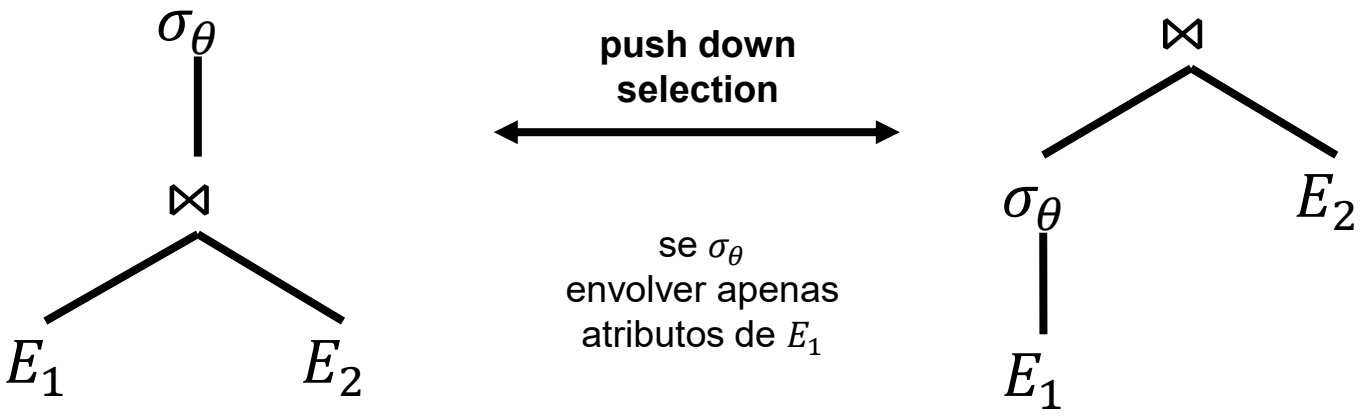
$$\boldsymbol{\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = \sigma_{\theta_1}(E_1) \bowtie_\theta (\sigma_{\theta_2}(E_2))}$$

- Essas transformações são conhecidas como push-down selection

Representação das regras de equivalência



Representação das regras de equivalência



Regras de equivalência

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

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

Regras de equivalência

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

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

Regras de equivalência

- A união e interseção de conjuntos são comutativas

$$\boldsymbol{E_1 \cup E_2 = E_2 \cup E_1}$$
$$\boldsymbol{E_1 \cap E_2 = E_2 \cap E_1}$$

  (a diferente de conjunto não é comutativa)

- A união e a interseção de conjuntos são associativas

$$\boldsymbol{(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)}$$
$$\boldsymbol{(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)}$$

Regras de equivalência

- A operação de seleção tem distribuição por ∪, ∩ e −

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

Também:

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

A operação de projeção tem distribuição pela união

$$\boldsymbol{\pi_L(E_1 \cup E_2) = \pi_L(E_1) \cup \pi_L(E_2)}$$

Exemplo de transformação

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

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

$$\equiv$$

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

  - 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

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

$$\equiv$$

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

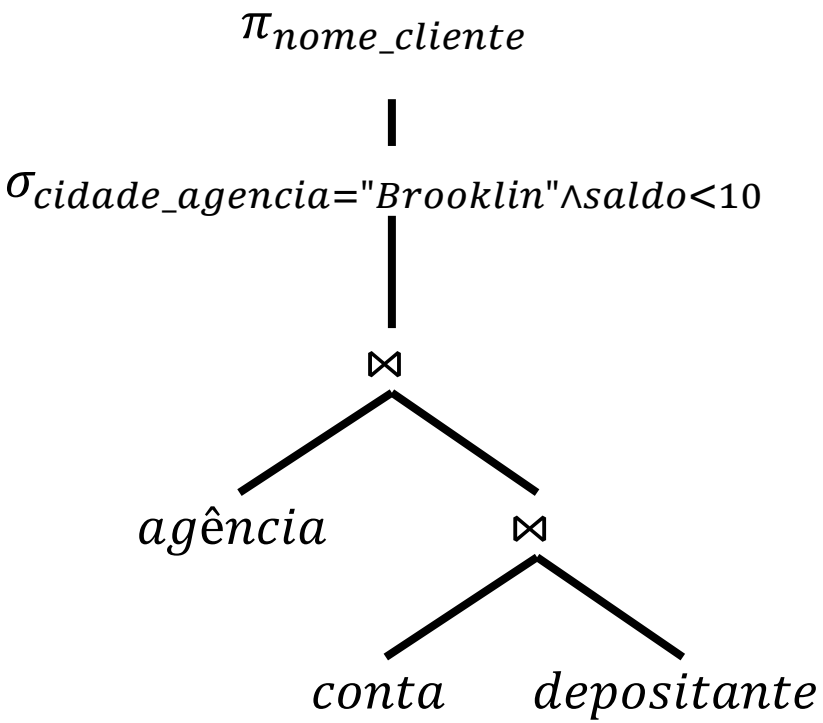
- A segunda forma oferece uma oportunidade para aplicar a regra "realizar seleções cedo", resultando na subexpressão

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

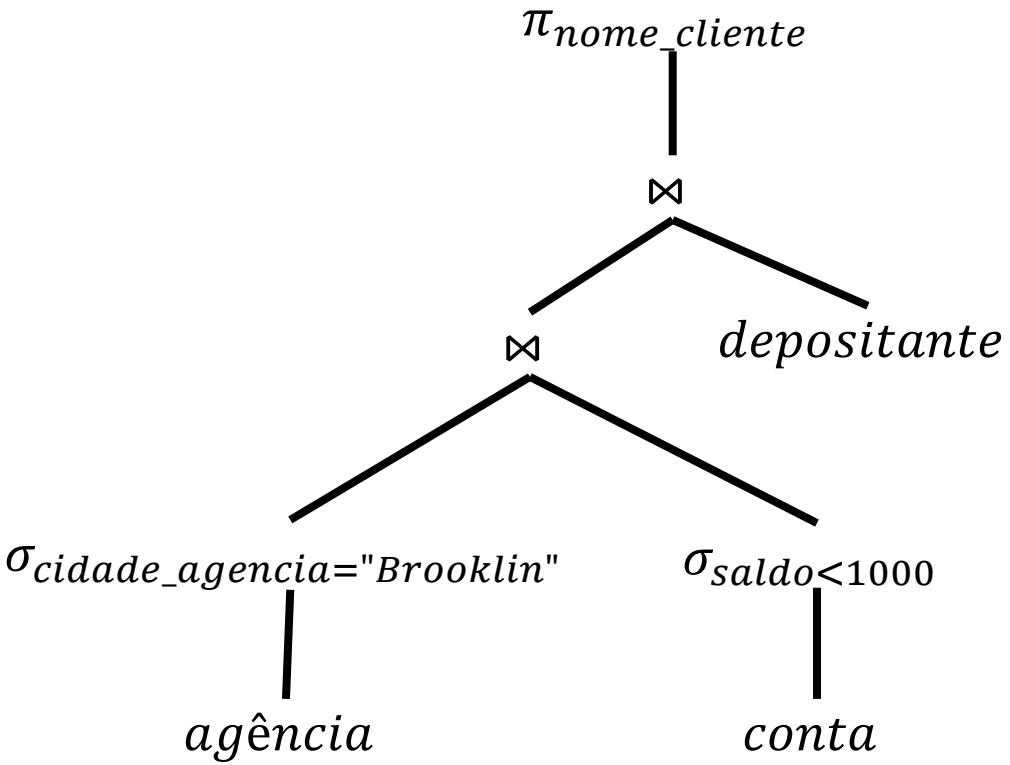
$$\equiv$$

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

Múltiplas transformações



(a) árvore da expressão inicial



(b) árvore após múltiplas transformações

Exemplo de operação de projeção

- Quando calculamos

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

  - 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:

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

- 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 $\boldsymbol{R_1, R_2, R_3}$
  - $(\boldsymbol{R_1} \bowtie \boldsymbol{R_2}) \bowtie \boldsymbol{R_3} = \boldsymbol{R_1} \bowtie (\boldsymbol{R_2} \bowtie \boldsymbol{R_3})$
  - Se $\boldsymbol{R_2} \bowtie \boldsymbol{R_3}$ for muito grande
    e $\boldsymbol{R_1} \bowtie \boldsymbol{R_2}$ for pequeno,
    escolhemos $(\boldsymbol{R_1} \bowtie \boldsymbol{R_2}) \bowtie \boldsymbol{R_3}$
    de modo que calculamos e armazenamos uma relação temporária menor

Exemplo de ordenação de junção (cont.)

- Considere a expressão

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

  - Poderia calcular $\boldsymbol{conta \bowtie depositante}$ primeiro e depois o resultado da junção como $\boldsymbol{\sigma_{cidadeAgencia="Brooklyn"}(agência)}$
  - Entretanto, $\boldsymbol{conta \bowtie depositante}$ 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

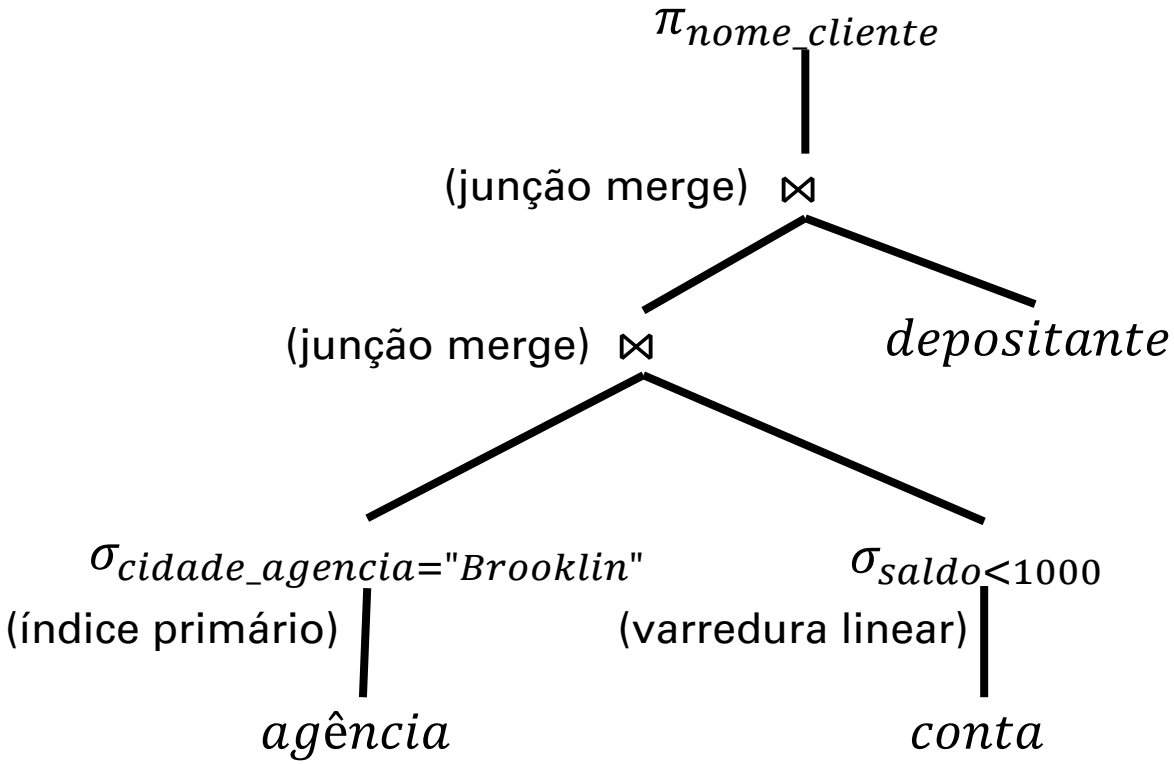
  $\boldsymbol{\pi_{nomeCliente}((\sigma_{cidadeAgencia="Brooklyn"}(agência) \bowtie conta) \bowtie depositante)}$

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

$$\boldsymbol{R_1 \bowtie R_2 \bowtie R_3 \bowtie \cdots \bowtie R_n}$$

- Existem $\frac{\boldsymbol{C_{n-1}^{2^n-1}}}{\boldsymbol{n}}$ 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 $\{\boldsymbol{R_1}, \boldsymbol{R_2}, \ldots, \boldsymbol{R_n}\}$ 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: $\boldsymbol{S_1} \bowtie (S - \boldsymbol{S_1})$ onde $\boldsymbol{S_1}$ 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 $\boldsymbol{2^n - 1}$ 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 $achar\boldsymbol{M}elhor\boldsymbol{P}lano(S)$

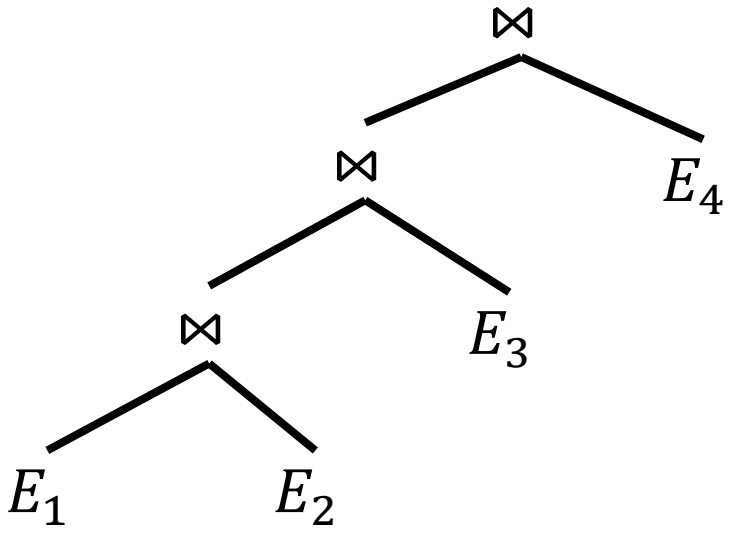
if (melhorplano[S].custo $\neq \infty$)
    return melhorplano[S]
// Se não estava calculado o melhorplano[S]; calcula-se agora
for each subconjunto não vazio $\boldsymbol{S_1}$ de S de modo que $\boldsymbol{S_1} \neq S$
    $\boldsymbol{P_1} = achar\boldsymbol{M}elhor\boldsymbol{P}lano(\boldsymbol{S_1})$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

Custo da otimização

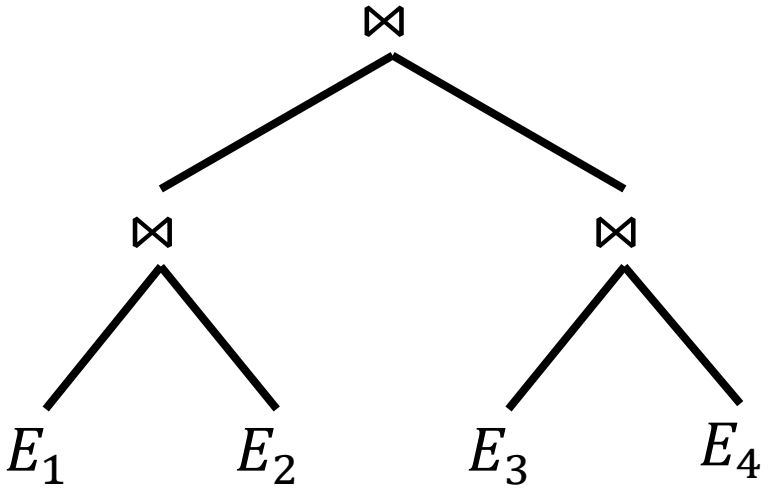
- Com o tempo de programação dinâmico, a complexidade da otimização geral é $\boldsymbol{O(3^n)}$
  - Com $n = 10$, esse número é 59000 em vez de 176 bilhões!
- A complexidade de espaço é $\boldsymbol{O(2^n)}$

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

- 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 $\boldsymbol{O(n2^n)}$
- A complexidade de espaço permanece em $\boldsymbol{O(2^n)}$
- A otimização baseada em custo é dispendiosa, mas compensa para consultas sobre datasets grandes

  (consultas típicas possuem n pequeno, geralmente < 10)

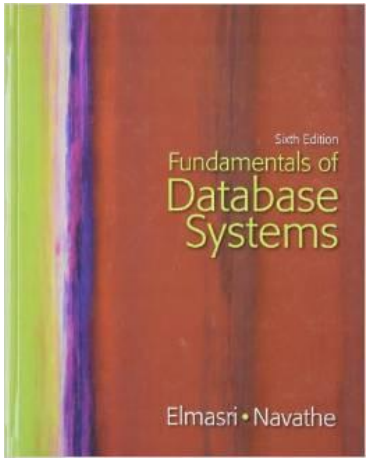
Ordens interessantes na otimização baseada em custo

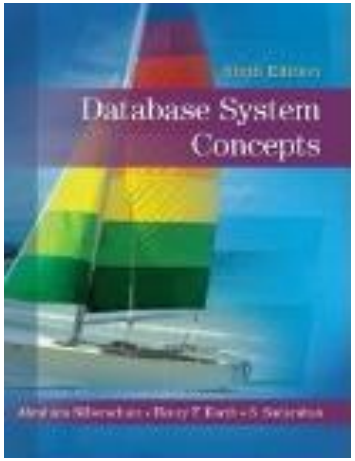
- Considere a expressão $(\boldsymbol{R_1} \bowtie \boldsymbol{R_2} \bowtie \boldsymbol{R_3}) \bowtie \boldsymbol{R_4} \bowtie \boldsymbol{R_5}$
- Uma ordem de classificação interessante é uma ordem de classificação as tuplas para as próximas operações
  - O uso da junção merge para calcular $\boldsymbol{R_1} \bowtie \boldsymbol{R_2} \bowtie \boldsymbol{R_3}$ pode ser mais dispendioso, mas pode oferecer uma saída classificada em uma ordem interessante para o processamento de $\boldsymbol{R_4}$ ou $\boldsymbol{R_5}$
- 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
  - Normalmente, o número de ordens interessantes é muito pequeno e não afeta a complexidade de tempo/espaço 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:
  - Substituir operações de produto cartesiano que são seguidas por uma condição de seleção por operações de junção
  - 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
  - Identificar as sub-árvores cujas operações podem ser canalizadas, e executá-las usando pipelining
  - Alguns sistemas utilizam apenas heurísticas, outros combinam heurísticas com otimização parcial baseada em custo.

Referências





Elmasri, R.; Navathe, S. B. Fundamentals of Database Systems. Pearson, 2016.
Korth, H. F.; Sudarshan, S.; Silberschatz, A. Database System Concepts. McGraw-Hill, 2019.
Özsu, M. T.; Valduriez, P. Principles of Distributed Database Systems. [s.l.] Springer Nature, 2019.

A maioria dos slides apresenta exemplos do livro Database System Concepts