Processamento e Otimização de Consultas em Bancos de Dados

Disciplina Bancos de Dados II

Prof. Renato Fileto

INE/CTC/UFSC

Tópicos

- Introdução ao processamento e otimização de consultas
- Otimização heurística de consultas
  - Regras de transformação
  - Estimativas de distribuição dos valores de dados
  - Algoritmo para otimização heurística
- Algoritmos para operadores relacionais
  - Seleção
  - Junção, união e diferença
  - Pipeline de operações
- Bibliografia e leituras recomendadas
- Exercícios

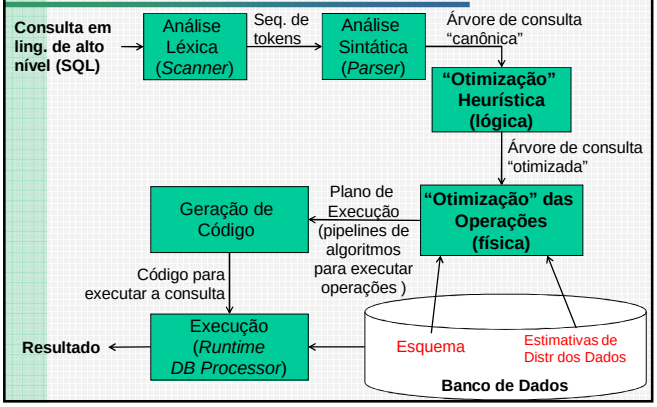
Introdução

- Linguagens de alto nível (ex.SQL) podem ter consultas com alto tempo de processamento
- Expressões e estratégias de execução diferentes, com desempenhos distintos e resultados idênticos, para solucionar uma consulta
  - Oportunidade para otimização lógica (converter expressões de consulta em expressões mais eficientes)
- SQL não define os algoritmos a serem utilizados para executar as operações constituintes de uma expressão de consulta
  - Oportunidade para otimização física (escolher os algoritmos e estruturas de dados a serem utilizados)

Introdução

- "Otimizadores" de consultas de SGBDs procuram determinar uma estratégia de execução eficiente para as consultas
- Geram estratégias otimizadas, mas sem necessariamente ser as melhores possíveis
- O processo de otimização leva em consideração:
  - A teoria relacional de dados (álgebra relacional)
  - A organização de dados e estruturas de acesso em disco
  - O tamanho das tabelas (número e tamanho dos registros)
  - Estimativas de distribuição dos dados
  - Seletividades das operações das consultas

Processamento de Consulta



Otimização Heurística de Consultas

- Também conhecida como otimização algébrica e otimização lógica
- Faz manipulações algébricas, aplicando regras de transformação de seqüências de operações
- Não considera o modo como as relações estão armazenadas
- Visa reduzir os tamanhos dos resultados intermediários

Regras de Otimização Heurística

- Principais regras de otimização de consultas:

1. Executar seleções e projeções o tão logo quanto possível (reduz o tamanho dos resultados intermediários)
2. Executar primeiro operações cuja seletividade resulte em menor quantidade de blocos a serem lidos do disco
3. Combinar, quando possível, seleção com produto cartesiano formando junção
4. Combinar seqüências de operações (e.g., seleção, projeção, junção)
5. Procurar sub-expressões comuns (a serem executadas repetidas vezes) e guardá-las caso seja mais eficiente ler seus resultados do que reprocessá-las

Regras de Transformação
(Elmasri / Navathe 2009)

1. Cascade of σ: A conjunctive selection condition can be broken up into a cascade (that is, a sequence) of individual σ operations:

   $\sigma_{c1 \text{ AND } c2 \text{ AND } \ldots \text{ AND } cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots(\sigma_{cn}(R))\ldots))$

2. Commutativity of σ: The σ operation is commutative:

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

3. Cascade of π: In a cascade (sequence) of π operations, all but the last one can be ignored:

   $\pi_{List1}(\pi_{List2}(\ldots(\pi_{Listn}(R))\ldots)) \equiv \pi_{List1}(R)$

4. Commuting σ with π: If the selection condition c involves only those attributes A1, ..., An in the projection list, the two operations can be commuted:

   $\pi_{A1, A2, \ldots, An}(\sigma_c(R)) \equiv \sigma_c(\pi_{A1, A2, \ldots, An}(R))$

5. Commutativity of $\bowtie$ (and $\times$): The $\bowtie$ operation is commutative, as is the $\times$ operation:

   $R \bowtie_c S \equiv S \bowtie_c R$
   $R \times S \equiv S \times R$

Regras de Transformação
(Elmasri / Navathe 2009)

6. Commuting σ with $\bowtie$ (or $\times$): If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows:

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

   Alternatively, if the selection condition c can be written as (c1 AND c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:

   $\sigma_c(R \bowtie S) \equiv (\sigma_{c1}(R)) \bowtie (\sigma_{c2}(S))$

   The same rules apply if the $\bowtie$ is replaced by a $\times$ operation.

7. Commuting π with $\bowtie$ (or $\times$): Suppose that the projection list is $L = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$, where $A_1, \ldots, A_n$ are attributes of R and $B_1, \ldots, B_m$ are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:

   $\pi_L(R \bowtie_c S) \equiv (\pi_{A1, \ldots, An}(R)) \bowtie_c (\pi_{B1, \ldots, Bm}(S))$

   $\pi_L(R \bowtie_c S) \equiv \pi_L((\pi_{A1, \ldots, An, An+1, \ldots, An+k}(R)) \bowtie_c (\pi_{B1, \ldots, Bm, Bm+1, \ldots, Bm+p}(S)))$

Regras de Transformação
(Elmasri / Navathe 2009)

8. Commutativity of set operations: The set operations $\cup$ and $\cap$ are commutative but $-$ is not.
9. Associativity of $\bowtie$, $\times$, $\cup$, and $\cap$: These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have:

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

10. Commuting σ with set operations: The σ operation commutes with $\cup$, $\cap$, and $-$. If θ stands for any one of these three operations (throughout the expression), we have:

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

11. The π operation commutes with $\cup$:

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

12. Converting a $(\sigma, \times)$ sequence into $\bowtie$: If the condition c of a σ that follows a $\times$ corresponds to a join condition, convert the $(\sigma, \times)$ sequence into a $\bowtie$ as follows:

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

Esboço do Algoritmo de Otimização Heurística de Consultas

1. Usando a regra 1, quebre condições conjuntivas.
2. Usando as regras 2, 4, 6 e 10, empurre seleções para tão perto quanto possível dos nós folhas da árvore de consulta.
3. Usando as regras 5 e 9, reorganize a árvore para executar condições mais seletivas primeiro.
4. Usando a regra 12, combine seleção com produto cartesiano formando junção, quando possível.
5. Usando as regras 3, 4, 7 e 11, empurre projeções para tão perto quanto possível dos nós folhas da árvore de consulta.
6. Combinar seqüências de operações (e.g., seleção, projeção, junção) que possam ser usadas por um único algoritmo (e.g., *pipeline*).
7. Procure sub-expressões comuns, avalie-as e guarde os resultados caso seja mais eficiente lê-los.

Algoritmos para executar operações

- Complexidade mínima inerente aos problemas
  - **Seleção (busca) – O(log(N))**
    - Hashing (aplicável sob restrições) – O(1)
  - **Produto Cartesiano – $O(N^2)$**
    - O tamanho da saída de *R X S* é $|R|*|S|$
  - **Junção, União, Interseção e Diferença – O(N)**
    - Para fazer *R [x] S* precisa carregar todos os registros de *R* e *S* ao menos uma vez, i.e., $|R|+|S|$ registros/páginas
  - **Projeção** - complexidade da operação conjugada

  Obs.: A execução de operações pode ser combinada

Leituras recomendadas

- Elmasri, R.; Navathe, S.B. *Fundamentals of database Systems. 4 ed.,* Benjamin Cummings, 2003.
  - **Chapter 15 – Algorithms for Query Processing and Optimization**
  - **Chapter 16 – Practical Database Design and Tuning**
- Ramakrishnan, R. *Database management systems. 2nd edition.* McGraw-Hill, 1998.
  - **Part 4 – Query Evaluation**
    - **Chapter 11 – External Sorting**
    - **Chapter 12 – Evaluation of Relational Operators**
    - **Chapter 13 – Introduction to Query Optimization**
    - **Chapter 14 – A Typical Relational Query Optimizer**