Chapter 5: Overview of Query Processing

- Query Processing Overview
- Query Optimization
- Distributed Query Processing Steps

Acknowledgements: I am indebted to Arturas Mazeika for providing me his slides of this course.
Query Processing Overview

- **Query processing**: A 3-step process that transforms a high-level query (of relational calculus/SQL) into an *equivalent* and *more efficient* lower-level query (of relational algebra).

1. **Parsing and translation**
   - Check syntax and verify relations.
   - Translate the query into an equivalent relational algebra expression.

2. **Optimization**
   - Generate an optimal evaluation plan (with lowest cost) for the query plan.

3. **Evaluation**
   - The query-execution engine takes an (optimal) evaluation plan, executes that plan, and returns the answers to the query.
• The success of RDBMSs is due, in part, to the availability
  – of declarative query languages that allow to easily express complex queries without knowing about the details of the physical data organization and
  – of advanced query processing technology that transforms the high-level user/application queries into efficient lower-level query execution strategies.

• The query transformation should achieve both correctness and efficiency
  – The main difficulty is to achieve the efficiency
  – This is also one of the most important tasks of any DBMS

• Distributed query processing: Transform a high-level query (of relational calculus/SQL) on a distributed database (i.e., a set of global relations) into an equivalent and efficient lower-level query (of relational algebra) on relation fragments.

• Distributed query processing is more complex
  – Fragmentation/replication of relations
  – Additional communication costs
  – Parallel execution
Query Processing Example

- **Example:** Transformation of an SQL-query into an RA-query.
  Relations: EMP(ENO, ENAME, TITLE), ASG(ENO, PNO, RESP, DUR)
  Query: *Find the names of employees who are managing a project?*

  - High level query
    ```
    SELECT ENAME
    FROM EMP, ASG
    WHERE EMP.ENO = ASG.ENO AND DUR > 37
    ```

  - Two possible transformations of the query are:
    * Expression 1: \( \Pi_{ENAME}({\sigma}_{DUR>37 \land EMP.ENO=ASG.ENO}(EMP \times ASG)) \)
    * Expression 2: \( \Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{DUR>37}(ASG))) \)

  - Expression 2 avoids the expensive and large intermediate Cartesian product, and therefore typically is better.
We make the following assumptions about the data fragmentation:

- Data is (horizontally) fragmented:
  - Site1: $ASG1 = \sigma_{ENO \leq \text{"E3"}}(ASG)$
  - Site2: $ASG2 = \sigma_{ENO > \text{"E3"}}(ASG)$
  - Site3: $EMP1 = \sigma_{ENO \leq \text{"E3"}}(EMP)$
  - Site4: $EMP2 = \sigma_{ENO > \text{"E3"}}(EMP)$
  - Site5: Result

- Relations ASG and EMP are fragmented in the same way

- Relations ASG and EMP are locally clustered on attributes RESP and ENO, respectively
• Now consider the expression $\Pi_{ENAME}(EMP \times_{ENO} (\sigma_{DUR>37}(ASG)))$

• Strategy 1 (partially parallel execution):
  – Produce $ASG'_1$ and move to Site 3
  – Produce $ASG'_2$ and move to Site 4
  – Join $ASG'_1$ with $EMP_1$ at Site 3 and move the result to Site 5
  – Join $ASG'_2$ with $EMP_2$ at Site 4 and move the result to Site 5
  – Union the result in Site 5

• Strategy 2:
  – Move $ASG_1$ and $ASG_2$ to Site 5
  – Move $EMP_1$ and $EMP_2$ to Site 5
  – Select and join at Site 5

• For simplicity, the final projection is omitted.
Query Processing Example . . .

- Calculate the cost of the two strategies under the following assumptions:
  - Tuples are uniformly distributed to the fragments; 20 tuples satisfy $\text{DUR} > 3.7$
  - size(EMP) = 400, size(ASG) = 1000
  - tuple access cost = 1 unit; tuple transfer cost = 10 units
  - ASG and EMP have a local index on DUR and ENO

- Strategy 1
  - Produce ASG’s: $(10+10) \times \text{tuple access cost}$ 20
  - Transfer ASG’s to the sites of EMPs: $(10+10) \times \text{tuple transfer cost}$ 200
  - Produce EMP’s: $(10+10) \times \text{tuple access cost} \times 2$ 40
  - Transfer EMP’s to result site: $(10+10) \times \text{tuple transfer cost}$ 200
  - Total cost 460

- Strategy 2
  - Transfer EMP₁, EMP₂ to site 5: $400 \times \text{tuple transfer cost}$ 4,000
  - Transfer ASG₁, ASG₂ to site 5: $1000 \times \text{tuple transfer cost}$ 10,000
  - Select tuples from ASG₁ $\cup$ ASG₂: $1000 \times \text{tuple access cost}$ 1,000
  - Join EMP and ASG’: $400 \times 20 \times \text{tuple access cost}$ 8,000
  - Total cost 23,000
Query optimization is a crucial and difficult part of the overall query processing. The objective of query optimization is to minimize the following cost function:

\[ \text{I/O cost} + \text{CPU cost} + \text{communication cost} \]

Two different scenarios are considered:

- Wide area networks
  - Communication cost dominates
    - low bandwidth
    - low speed
    - high protocol overhead
  - Most algorithms ignore all other cost components
- Local area networks
  - Communication cost not that dominant
  - Total cost function should be considered
 Ordering of the operators of relational algebra is crucial for efficient query processing

• Rule of thumb: move expensive operators at the end of query processing

• Cost of RA operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select, Project</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>(without duplicate elimination)</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>(with duplicate elimination)</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Join</td>
<td></td>
</tr>
<tr>
<td>Semi-join</td>
<td></td>
</tr>
<tr>
<td>Division</td>
<td></td>
</tr>
<tr>
<td>Set Operators</td>
<td></td>
</tr>
<tr>
<td>Cartesian Product</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>
Several issues have to be considered in query optimization

- **Types of query optimizers**
  - wrt the search techniques (exhaustive search, heuristics)
  - wrt the time when the query is optimized (static, dynamic)

- **Statistics**

- **Decision sites**

- **Network topology**

- **Use of semijoins**
Query Optimization Issues

- Types of Query Optimizers wrt Search Techniques
  - Exhaustive search
    * Cost-based
    * Optimal
    * Combinatorial complexity in the number of relations
  - Heuristics
    * Not optimal
    * Regroups common sub-expressions
    * Performs selection, projection first
    * Replaces a join by a series of semijoins
    * Reorders operations to reduce intermediate relation size
    * Optimizes individual operations
Query Optimization Issues . . .

- Types of Query Optimizers wrt Optimization Timing
  - Static
    * Query is optimized prior to the execution
    * As a consequence it is difficult to estimate the size of the intermediate results
    * Typically amortizes over many executions
  - Dynamic
    * Optimization is done at run time
    * Provides exact information on the intermediate relation sizes
    * Have to re-optimize for multiple executions
  - Hybrid
    * First, the query is compiled using a static algorithm
    * Then, if the error in estimate sizes greater than threshold, the query is re-optimized at run time
Query Optimization Issues ... 

- **Statistics**
  - Relation/fragments
    - Cardinality
    - Size of a tuple
    - Fraction of tuples participating in a join with another relation/fragment
  - Attribute
    - Cardinality of domain
    - Actual number of distinct values
    - Distribution of attribute values (e.g., histograms)
  - Common assumptions
    - Independence between different attribute values
    - Uniform distribution of attribute values within their domain
Query Optimization Issues . . .

- Decision sites
  - Centralized
    * Single site determines the "best" schedule
    * Simple
    * Knowledge about the entire distributed database is needed
  - Distributed
    * Cooperation among sites to determine the schedule
    * Only local information is needed
    * Cooperation comes with an overhead cost
  - Hybrid
    * One site determines the global schedule
    * Each site optimizes the local sub-queries
• **Network topology**
  
  - Wide area networks (WAN) point-to-point
    
    * Characteristics
      - Low bandwidth
      - Low speed
      - High protocol overhead
    
    * Communication cost dominate; all other cost factors are ignored
    
    * Global schedule to minimize communication cost
    
    * Local schedules according to centralized query optimization

  - Local area networks (LAN)
    
    * Communication cost not that dominant
    
    * Total cost function should be considered
    
    * Broadcasting can be exploited (joins)
    
    * Special algorithms exist for star networks
Query Optimization Issues . . .

- **Use of Semijoins**
  - Reduce the size of the join operands by first computing semijoins
  - Particularly relevant when the main cost is the communication cost
  - Improves the processing of distributed join operations by reducing the size of data exchange between sites
  - However, the number of messages as well as local processing time is increased
Distributed Query Processing Steps

CONTROL SITE

Calculus Query on Distributed Relations

Query Decomposition

GLOBAL SCHEMA

Algebraic Query on Distributed Relations

Data Localization

FRAGMENT SCHEMA

Fragment Query

Global Optimization

STATS ON FRAGMENTS

CONTROL SITE

Optimized Fragment Query with Communication Operations

LOCAL SITES

Local Optimization

LOCAL SCHEMAS

Optimized Local Queries
• Query processing transforms a high level query (relational calculus) into an equivalent lower level query (relational algebra). The main difficulty is to achieve the efficiency in the transformation

• Query optimization aims to minimize the cost function:

\[ \text{I/O cost} + \text{CPU cost} + \text{communication cost} \]

• Query optimizers vary by search type (exhaustive search, heuristics) and by type of the algorithm (dynamic, static, hybrid). Different statistics are collected to support the query optimization process

• Query optimizers vary by decision sites (centralized, distributed, hybrid)

• Query processing is done in the following sequence: query decomposition→data localization→global optimization→local optimization