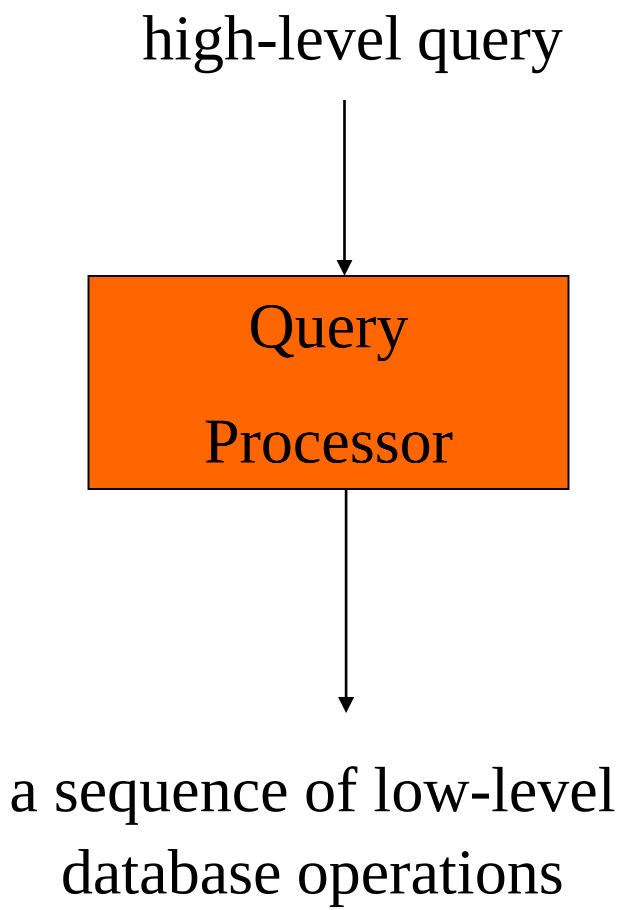
Distributed Data Processing

- Introduction
- Distributed DBMS Architecture
- Distributed DB Design
- Semantic Data Control
- **Query Processing**
- Transaction Management

Query Processing



5.Overview of Query Processing

- Query Processing Problem
- Objectives of Query Processing
- Complexity of Relational Algebra Operations
- Characterization of Query Processing
- Layers of Query Processing

5.1 Query Processing Problem

Query Processing Problem

- The main **function** of a relational query processor is to transform a high-level query (typically, in relational calculus) into an equivalent lower-level query (typically, in some variation of relational algebra)
- The low-level query actually implements the execution strategy for the query
- The transformation must achieve both **correctness** and **efficiency**
  - **Correct**: if the low-level query has the same semantics as the original query – produce the same result
  - **Efficient**: minimize resource consumption

Example of Efficiency

SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO
AND DUR >37

Strategy 1

$$\Pi_{\text{ENAME}} \left(\delta_{\text{DUR>37} \wedge \text{EMP.ENO=ASG.ENO}} (\text{EMP x ASG}) \right)$$

Strategy 2

$$\Pi_{\text{ENAME}} (\text{EMP} \infty_{\text{ENO}} (\delta_{\text{DUR>37}} (\text{ASG})))$$

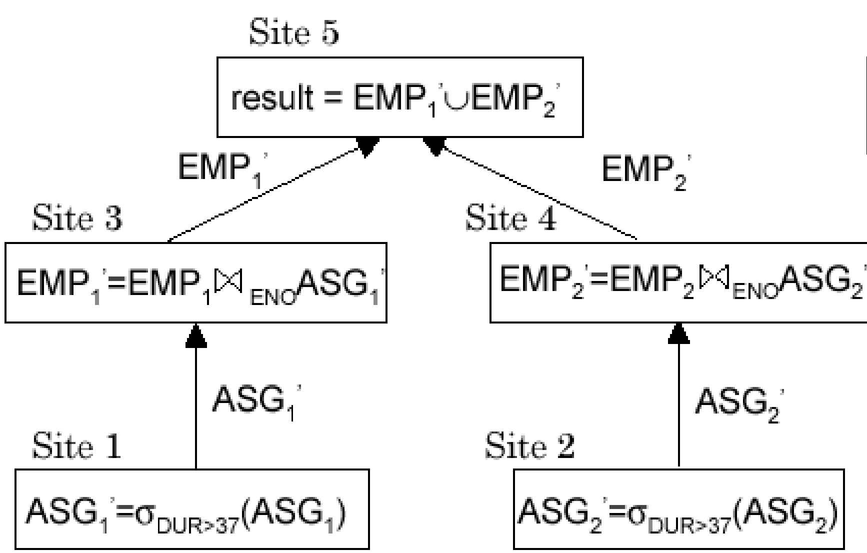
Strategy 2 avoids Cartesian product, so is "better"

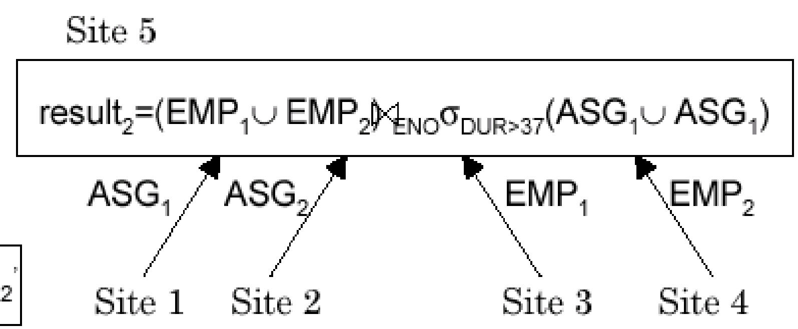
In a distributed system

- In a distributed system, relational algebra is not enough to express execution strategy. It must be supplemented with operations for exchanging data between sites
  - **Ordering** relational algebra operations
  - Selecting the best **sites** to process data and Possibly the **way** data should be transformed

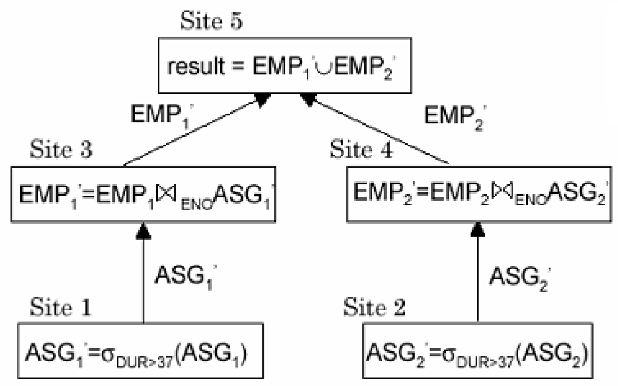
Example in a distributed environment

Site 1	Site 2	Site 3	Site 4	Site 5
$ASG_1=\sigma_{ENO\leq\text{"E3"}}(ASG)$	$ASG_2=\sigma_{ENO>\text{"E3"}}(ASG)$	$EMP_1=\sigma_{ENO\leq\text{"E3"}}(EMP)$	$EMP_2=\sigma_{ENO>\text{"E3"}}(EMP)$	Result





Cost of alternatives



- Assume:
  - *size*(EMP) = 400, *size*(ASG) = 1000, *size*(result)=20
  - tuple access cost = 1 unit; tuple transfer cost = 10 units
  - Uniformly distributed; locally clustered (ASG—DUR,EMP—ENO)

- **Strategy 1**

produce ASG': (10*tuple access cost) *2	20
transfer ASG' to the sites of EMP: (10*tuple transfer cost) *2	200
produce EMP': (10+10) ∗tuple access cost*2	40
transfer EMP' to result site: (10*tuple transfer cost) *2	200
Total cost	460

- **Strategy 2**

transfer EMP to site 5: 400∗tuple transfer cost	4,000
transfer ASG to site 5: 1000∗tuple transfer cost	10,000
produce ASG': 1000∗tuple access cost	1,000
join EMP and ASG': 400∗20∗tuple access cost	8,000
Total cost	23,000

5.2 Objectives of Query Processing

Objectives of Query Processing

An important aspect of query processing is
query optimization

Minimize a cost function

I/O cost + CPU cost + communication cost

These might have different weights in different distributed environments

Wide area networks

- communication cost will dominate
  - low bandwidth
  - low speed
  - high protocol overhead
- most algorithms ignore all other cost components

Local area networks

- communication cost is not dominant
- total cost function should be considered – a weighted combination

Can also minimize response time or maximize throughput

5.3 Complexity of Relational Algebra Operations

Complexity of Relational Algebra Operations

Assume

- relations of cardinality *n*
- sequential scan

Two principles:

1. The most selective operations that **reduce cardinalities** (e.g.,selection) should be performed first.

2. Operations should be **ordered** by increasing complexity so that Cartesian products can be avoided or delayed.

Operation	Complexity
Select **Project** **(without duplicate elimination)**	**O(n)**
Project **(with duplicate elimination)** **Group**	**O(n*log n)**
Join **Semi-join** **Division** **Set Operators**	**O(n*log n)**
Cartesian Product	**$O(n^2)$**

5.4 Characterization of Query Processing

Characterization of Query Processing

- Language
- Types of optimization
- Optimization timing
- Optimization granularity
- Statistics
- Decision sites
- Exploitation of the network topology
- Exploitation of replicated fragments
- Use of semijoins

Language

- **Input language** to the query processor
  - Based on relational calculus or relational algebra
- **Output language** from the query processor
  - Generally some internal form of relational algebra augmented with communication primitives
  - The operations of the output language are implemented directly in the system

Types of Optimization

- **Exhaustive search**
  - optimal
  - cost-based
  - combinatorial complexity in the number of relations
  - Only if query optimization is done once for many subsequent executions of the query
- **Heuristics**
  - not optimal
    - Restrict the solution space so that only a few strategies are considered
  - regroup common sub-expressions
  - perform selection, projection first
  - replace a join by a series of semijoins
  - reorder operations to reduce intermediate relation size
  - optimize individual operations

Optimization timing

- **Static**
  - compilation optimize prior to the execution
  - difficult to estimate the size of the intermediate results – estimated using database statistics
    - error propagation
  - cost can amortize over many executions
  - R*
- **Dynamic**
  - run time optimization
  - exact information on the intermediate relation sizes
  - have to reoptimize for multiple executions
  - Distributed INGRES
- **Hybrid**
  - compile using a static algorithm
  - if the error in estimate sizes > threshold, reoptimize at run time
  - MERMAID

Optimization granularity

- Single query at a time
  - cannot use common intermediate results
- Multiple queries at a time
  - efficient if many similar queries
  - decision space is much larger

Statistics

- Relation
  - cardinality
  - size of a tuple
  - fraction of tuples participating in a join with another relation
- Attribute
  - cardinality of domain
  - actual number of distinct values
- Common assumptions
  - **independence** between different attribute values
  - **uniform distribution** of attribute values within their domain

Decision sites

- **Centralized**
  - single site determines the “best” schedule
  - simple
  - need knowledge about the entire distributed database
- **Distributed**
  - cooperation among sites to determine the schedule
  - need only local information
  - cost of cooperation
- **Hybrid**
  - one site determines the global schedule
  - each site optimizes the local subqueries

Exploitation of the network topology

- Wide area networks (WAN) – point-to-point
  - characteristics
    - low bandwidth, low speed, high protocol overhead
  - communication cost will dominate; ignore all other cost factors
  - global schedule to minimize communication cost
  - local schedules according to centralized query optimization
- Local area networks (LAN)
  - communication cost is not dominant
  - total cost function should be considered
  - Increase parallel execution at the expense of communication cost
  - Take advantage of the network topology
    - broadcasting can be exploited (joins)
    - special algorithms exist for star networks

Exploitation of Replicated fragments

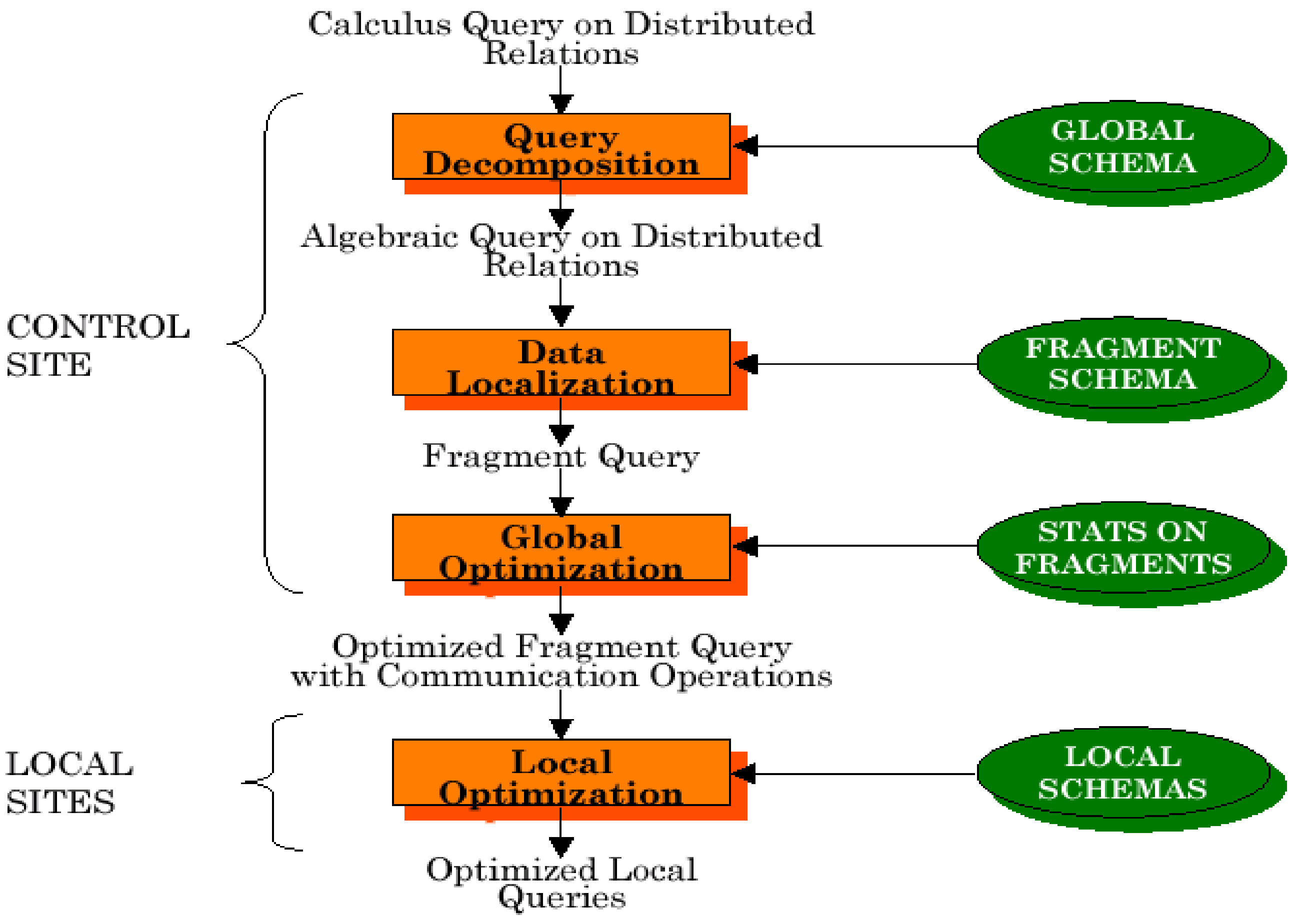
- Most optimization algorithms consider the localization process independently of optimization
- Some algorithms have exploited the existence of replicated fragments at run time in order to minimize communication times
  - The optimization algorithms is then more complex

Use of semijoins

- **Semijoin**
  - Reducing the size of the operand relation
  - May result in an increase in the number of message and in the local processing time
- Selecting an optimal combination of joins and semijoins.

5.5 Layers of Query Processing

Layers of Query Processing



Step 1: Query Decomposition

Input : **Calculus query on global relations**

- Normalization
  - manipulate query quantifiers and qualification
- Analysis
  - detect and reject “incorrect” queries
  - possible for only a subset of relational calculus
- Simplification
  - eliminate redundant predicates
- Restructuring
  - calculus query -> algebraic query
  - more than one translation is possible
  - use transformation rules

Step 2: Data Localization

- Input: **Algebraic query on distributed relations**
- Localize the query's data using data distributed information
  - Determine which fragments are involved
  - Transform the distributed (globe) query into a fragment query
- Step1: Localization program
  - substitute for each global query its materialization (reconstruction) program
- Step2: Optimize
  - simplification and restructuring

Step 3: Global Query Optimization

- Input: **Fragment query**
- Find the *best* (not necessarily optimal) global schedule, that is to find the best ordering of operations in the fragment query, including communication operations which minimize a cost function
  - Minimize a cost function
  - Available statistics on fragments
  - Distributed join processing
    - Bushy vs. linear trees
    - Which relation to ship where?
    - Ship-whole vs ship-as-needed
  - Decide on the use of semijoins
    - Semijoin saves on communication at the expense of more local processing.
  - Join methods
    - nested loop vs ordered joins (merge join or hash join)

Step 4: Local Optimization

Input: **Best global execution schedule**

- Select the best access path by all the site
- Use the centralized optimization techniques

References

- J.C. Freytag, D. Maier, and G. Vossen. *Query Processing for Advanced Database Systems*. Morgan-Kaufmann, 1994.
- W. Kim, D.S. Reiner and D.S. Batory. *Query Processing in Database Systems*. Springer-Verlag, 1985.
- G. Graefe. “Query Evaluation Techniques for Large Databases”, *ACM Computing Surveys*, 25(2): 73-170, June 1993.