Distributed Databases

by Farnoush Banaei-Kashani

Excerpt from “Principles of Distributed Database Systems”
by M. Tamer Özsu and Patrick Valduriez
Topics

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Semantic Data Control
- **Distributed Query Processing**
- Distributed Transaction Management
- Parallel Database Systems
- Distributed Object DBMS
- Database Interoperability
- Current Issues
Outline

- Problem Definition
- Issues to Consider
- Methodology
  - Step 1: Query Decomposition
  - Step 2: Data Localization
  - Step 3: Global Optimization
  - Step 4: Local Optimization
Query Processing

high level user query

query processor

low level data manipulation commands
Problem?

\[
\begin{align*}
\text{SELECT} & \quad \text{ENAME} \\
\text{FROM} & \quad \text{EMP, ASG} \\
\text{WHERE} & \quad \text{EMP.ENO} = \text{ASG.ENO} \\
\text{AND} & \quad \text{DUR} > 37
\end{align*}
\]

Strategy 1
\[
\Pi_{\text{ENAME}}(\sigma_{\text{DUR}>37 \land \text{EMP.ENO} = \text{ASG.ENO}} (\text{EMP} \times \text{ASG}))
\]

Strategy 2
\[
\Pi_{\text{ENAME}}(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{DUR}>37} (\text{ASG})))
\]

Strategy 2 avoids Cartesian product, so is “better”
## Problem in DDBS?

### Site 1

\[ \text{ASG}_1 = \sigma_{\text{ENo} \leq 3}(\text{ASG}) \]

### Site 2

\[ \text{ASG}_2 = \sigma_{\text{ENo} > 3}(\text{ASG}) \]

### Site 3

\[ \text{EMP}_1 = \sigma_{\text{ENo} \leq 3}(\text{EMP}) \]

### Site 4

\[ \text{EMP}_2 = \sigma_{\text{ENo} > 3}(\text{EMP}) \]

### Site 5

Result

---

**Diagram:**

1. **Site 1**
   - \( \text{ASG}_1 = \sigma_{\text{DUR} > 37}(\text{ASG}) \)

2. **Site 2**
   - \( \text{ASG}_2 = \sigma_{\text{ENo} > 3}(\text{ASG}) \)

3. **Site 3**
   - \( \text{EMP}_1 = \sigma_{\text{ENo} \leq 3}(\text{EMP}) \)

4. **Site 4**
   - \( \text{EMP}_2 = \sigma_{\text{ENo} > 3}(\text{EMP}) \)

5. **Site 5**
   - **Result**

   - \( \text{result} = \text{EMP}_1 \cup \text{EMP}_2 \)

   - **Site 3**
     - \( \text{EMP}_1 = \sigma_{\text{ENo} \leq 3}(\text{EMP}) \)

   - **Site 4**
     - \( \text{EMP}_2 = \sigma_{\text{ENo} > 3}(\text{EMP}) \)

---
Problem in DDBS?

■ Assume:
  - \( \text{size}(\text{EMP}) = 400, \text{size}(\text{ASG}) = 1000 \)
  - tuple access cost = 1 unit; tuple transfer cost = 10 units

■ Strategy 1
  1. produce ASG': (10+10) * tuple access cost 20
  2. transfer ASG' to the sites of EMP: (10+10) * tuple transfer cost 200
  3. produce EMP': (10+10) * tuple access cost * 2 40
  4. transfer EMP' to result site: (10+10) * tuple transfer cost 200
  Total cost 460

■ Strategy 2
  1. transfer EMP to site 5: 400 * tuple transfer cost 4,000
  2. transfer ASG to site 5: 1000 * tuple transfer cost 10,000
  3. produce ASG': 1000 * tuple access cost 1,000
  4. join EMP and ASG': 400 * 20 * tuple access cost 8,000
  Total cost 23,000
Query Optimization Objectives

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 not that dominant

- total cost function should be considered

Can also maximize throughput
## Complexity of Relational Operations

- **Assume**
  - relations of cardinality $n$
  - sequential scan

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Project (without duplicate elimination)</td>
<td></td>
</tr>
<tr>
<td>Project (with duplicate elimination) Group</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Join</td>
<td>$O(n \log n)$</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>
Outline

- Problem Definition
- Issues to Consider
- Methodology
  - Step 1: Query Decomposition
  - Step 2: Data Localization
  - Step 3: Global Optimization
  - Step 4: Local Optimization
Query Processing Issues – Types of Optimizers

- **Exhaustive search**
  - cost-based
  - optimal
  - combinatorial complexity in the number of relations

- **Heuristics**
  - not optimal
  - 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
Query Processing Issues – 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
Query Processing Issues – Optimization Timing

- **Static**
  - compilation optimize prior to the execution
  - difficult to estimate the size of the intermediate results
    - error propagation
  - 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
Query Processing Issues – 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
Query Processing Issues – 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
Query Processing Issues – 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 not that dominant
  - total cost function should be considered
  - broadcasting can be exploited (joins)
  - special algorithms exist for star networks
Outline

- Problem Definition
- Issues to Consider
- Methodology
  - Step 1: Query Decomposition
  - Step 2: Data Localization
  - Step 3: Global Optimization
  - Step 4: Local Optimization
Distributed Query Processing Methodology

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

Optimized Fragment Query with Communication Operations

LOCAL SITES

Local Optimization

LOCAL SCHEMAS

Optimized Local Queries
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 $\rightarrow$ algebraic query
  - more than one translation is possible
  - use transformation rules
Normalization

- Lexical and syntactic analysis
  - check validity (similar to compilers)
  - check for attributes and relations
  - type checking on the qualification

- Put into normal form
  - Conjunctive normal form
    \[(p_{i1} \lor p_{i2} \lor \ldots \lor p_{i\ell}) \land \ldots \land (p_{m1} \lor p_{m2} \lor \ldots \lor p_{m\ell})\]
  - Disjunctive normal form
    \[(p_{i1} \land p_{i2} \land \ldots \land p_{i\ell}) \lor \ldots \lor (p_{m1} \land p_{m2} \land \ldots \land p_{m\ell})\]
  - OR's mapped into union
  - AND's mapped into join or selection
Analysis

- Refute incorrect queries
- **Type incorrect**
  - If any of its attribute or relation names are not defined in the global schema
  - If operations are applied to attributes of the wrong type
- **Semantically incorrect**
  - Components do not contribute in any way to the generation of the result
  - Only a subset of relational calculus queries can be tested for correctness
  - Those that do not contain disjunction and negation
  - To detect
    - connection graph (query graph)
    - join graph
Analysis - Example

```
SELECT ENAME, RESP
FROM EMP, ASG, PROJ
WHERE EMP.ENO = ASG.ENO
AND ASG.PNO = PROJ.PNO
AND PNAME = "CAD/CAM"
AND DUR \geq 36
AND TITLE = "Programmer"
```
If the query graph is not connected, the query is wrong.

```
SELECT ENAME, RESP
FROM EMP, ASG, PROJ
WHERE EMP.ENO = ASG.ENO
AND PNAME = "CAD/CAM"
AND DUR ≥ 36
AND TITLE = "Programmer"
```
Simplification

- Why simplify?

- How? Use transformation rules
  - elimination of redundancy
    - idempotency rules
      \[ p_1 \land \neg(p_1) \Leftrightarrow \text{false} \]
      \[ p_1 \land (p_1 \lor p_2) \Leftrightarrow p_1 \]
      \[ p_1 \lor \text{false} \Leftrightarrow p_1 \]
      ...
  - application of transitivity

1. \( p \land p \Leftrightarrow p \)
2. \( p \lor p \Leftrightarrow p \)
3. \( p \land \text{true} \Leftrightarrow p \)
4. \( p \lor \text{false} \Leftrightarrow p \)
5. \( p \land \text{false} \Leftrightarrow \text{false} \)
6. \( p \lor \text{true} \Leftrightarrow \text{true} \)
7. \( p \land \neg p \Leftrightarrow \text{false} \)
8. \( p \lor \neg p \Leftrightarrow \text{true} \)
9. \( p_1 \land (p_1 \lor p_2) \Leftrightarrow p_1 \)
10. \( p_1 \lor (p_1 \land p_2) \Leftrightarrow p_1 \)
Simplification - Example

```
SELECT  TITLE
FROM    EMP
WHERE   EMP.ENAME = “J. Doe”
OR      (NOT(EMP.TITLE = “Programmer”))
AND     (EMP.TITLE = “Programmer”)
OR      EMP.TITLE = “Elect. Eng.”)
AND     NOT(EMP.TITLE = “Elect. Eng.”))

SELECT  TITLE
FROM    EMP
WHERE   EMP.ENAME = “J. Doe”
```
Restructuring

- Convert relational calculus to relational algebra
- Make use of query trees
- Example
  
  Find the names of employees other than J. Doe who worked on the CAD/CAM project for either 1 or 2 years.

  ```
  SELECT ENAME
  FROM EMP, ASG, PROJ
  WHERE EMP.ENO = ASG.ENO
  AND ASG.PNO = PROJ.PNO
  AND ENAME ≠ "J. Doe"
  AND PNAME = "CAD/CAM"
  AND (DUR = 12 OR DUR = 24)
  ```
Restructuring - Transformation Rules

- Commutativity of binary operations
  - $R \times S \Leftrightarrow S \times R$
  - $R \bowtie S \Leftrightarrow S \bowtie R$
  - $R \cup S \Leftrightarrow S \cup R$

- Associativity of binary operations
  - $(R \times S) \times T \Leftrightarrow R \times (S \times T)$
  - $(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T')$

- Idempotence of unary operations
  - $\Pi_{A'}(\Pi_{A'}(R)) \Leftrightarrow \Pi_{A'}(R)$
  - $\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) = \sigma_{p_1(A_1) \wedge p_2(A_2)}(R)$
  - where $R[A]$ and $A' \subseteq A$, $A'' \subseteq A$ and $A' \subseteq A''$

- Commuting selection with projection
Restructuring - Transformation Rules

- Commuting selection with binary operations
  \[ \sigma_{p(A)}(R \times S) \iff (\sigma_{p(A)}(R)) \times S \]
  \[ \sigma_{p(A_j)}(R \bowtie_{(A_j,B_k)} S) \iff (\sigma_{p(A_j)}(R)) \bowtie_{(A_j,B_k)} S \]
  \[ \sigma_{p(A_j)}(R \cup T) \iff \sigma_{p(A_j)}(R) \cup \sigma_{p(A_j)}(T) \]
  where \(A_i\) belongs to \(R\) and \(T\)

- Commuting projection with binary operations
  \[ \Pi_{C}(R \times S) \iff \Pi_{A'}(R) \times \Pi_{B}(S) \]
  \[ \Pi_{C}(R \bowtie_{(A_j,B_k)} S) \iff \Pi_{A'}(R) \bowtie_{(A_j,B_k)} \Pi_{B}(S) \]
  \[ \Pi_{C}(R \cup S) \iff \Pi_{C}(R) \cup \Pi_{C}(S) \]
  where \(R[A]\) and \(S[B]\); \(C = A' \cup B'\) where \(A' \subset A, B' \subset B\)
Example

Recall the previous example:
Find the names of employees other than J. Doe who worked on the CAD/CAM project for either one or two years.

```
SELECT  ENAME
FROM     PROJ, ASG, EMP
WHERE    ASG.ENO=EMP.ENO
AND      ASG.PNO=PROJ.PNO
AND      ENAME≠“J. Doe”
AND      PROJ.PNAME=“CAD/CAM”
AND      (DUR=12 OR DUR=24)
```
Equivalent Query

\[ \Pi_{ENAME} \]

\[ \sigma_{PNAME=\text{"CAD/CAM"}} \land (DUR=12 \lor DUR=24) \land ENAME \neq \text{"J. DOE"} \]

\[ \bowtie_{PNO \land ENO} \]

\[ \times \]

\[ \text{ASG} \]

\[ \text{PROJ} \]

\[ \text{EMP} \]
Restructuring

\[ \sigma_{\text{PNAME} = \text{"CAD/CAM"}} \]
\[ \sigma_{\text{DUR} = 12, \text{DUR} = 24} \]
\[ \sigma_{\text{ENAME} \neq \text{"J. Doe"}} \]
Distributed Query Processing Methodology

- **CONTROL SITE**
  - Calculus Query on Distributed Relations
  - Algebraic Query on Distributed Relations
    - Data Localization
      - Fragment Query
      - Global Optimization
        - Optimized Fragment Query with Communication Operations
          - Local Optimization
            - Optimized Local Queries
            - LOCAL SCHEMAS
    - GLOBAL SCHEMA
    - FRAGMENT SCHEMA
    - STATS ON FRAGMENTS

- **LOCAL SITES**
  - LOCAL SCHEMAS
Step 2 – Data Localization

**Input:** Algebraic query on distributed relations

- Determine which fragments are involved

**Localization program**

- substitute for each global query its materialization program
- optimize
Example

Assume

- EMP is fragmented into EMP₁, EMP₂, EMP₃ as follows:
  - EMP₁ = \( \sigma_{\text{ENO} \leq \text{E}3} \)(EMP)
  - EMP₂ = \( \sigma_{\text{E}3 < \text{ENO} \leq \text{E}6} \)(EMP)
  - EMP₃ = \( \sigma_{\text{ENO} \leq \text{E}6} \)(EMP)

- ASG fragmented into ASG₁ and ASG₂ as follows:
  - ASG₁ = \( \sigma_{\text{ENO} \leq \text{E}3} \)(ASG)
  - ASG₂ = \( \sigma_{\text{ENO} > \text{E}3} \)(ASG)

Replace EMP by \((\text{EMP}_1 \cup \text{EMP}_2 \cup \text{EMP}_3)\) and ASG by \((\text{ASG}_1 \cup \text{ASG}_2)\) in any query.
Provides Parallelism
Eliminates Unnecessary Work
Reduction for PHF

- **Reduction with selection**

  - Relation \( R \) and \( F_R = \{ R_1, R_2, \ldots, R_w \} \) where \( R_j = \sigma_{p_j}(R) \)

  \[ \sigma_{p_i}(R_j) = \phi \text{ if } \forall x \text{ in } R: \neg(p_i(x) \land p_j(x)) \]

  - Example

    ```sql
    SELECT *
    FROM EMP
    WHERE ENO = "E5"
    ```

    ![Diagram showing the reduction process with selection]

    ```sql
    \sigma_{ENO="E5"} \left( \bigcup \text{EMP}_i \right)
    ```
Reduction for PHF

- Reduction with join
  - Possible if fragmentation is done on join attribute
  - Distribute join over union
    \[(R_1 \cup R_2) \bowtie S \Leftrightarrow (R_1 \bowtie S) \cup (R_2 \bowtie S)\]
  - Given \( R_i = \sigma_{p_i}(R) \) and \( R_j = \sigma_{p_j}(R) \)
    \[R_i \bowtie R_j = \emptyset \text{ if } \forall x \text{ in } R_i, \forall y \text{ in } R_j: \neg(p_i(x) \land p_j(y))\]
Reduction for PHF

Reduction with join - Example

- Assume EMP is fragmented as before and

  \[ \text{ASG}_1: \sigma_{\text{ENO} \leq \text{"E3"}}(\text{ASG}) \]

  \[ \text{ASG}_2: \sigma_{\text{ENO} > \text{"E3"}}(\text{ASG}) \]

- Consider the query

  \[
  \text{SELECT}^* \\
  \text{FROM} \quad \text{EMP, ASG} \\
  \text{WHERE} \quad \text{EMP.ENO} = \text{ASG.ENO}
  \]
Reduction for PHF

- Reduction with join - Example
  - Distribute join over unions
  - Apply the reduction rule
Reduction for VF

- Find useless (not empty) intermediate relations

Relation $R$ defined over attributes $A = \{A_1, ..., A_n\}$ vertically fragmented as $R_i = \Pi_{A'}(R)$ where $A' \subseteq A$:

$\Pi_{D,K}(R_i)$ is useless if the set of projection attributes $D$ is not in $A'$

Example: $\text{EMP}_1 = \Pi_{\text{ENO,ENAME}}(\text{EMP})$; $\text{EMP}_2 = \Pi_{\text{ENO,TITLE}}(\text{EMP})$

\[
\begin{align*}
\text{SELECT} & \quad \text{ENAME} \\
\text{FROM} & \quad \text{EMP} \\
\end{align*}
\]

\[
\begin{align*}
\Pi_{\text{ENAME}} & \quad \rightarrow \\
\text{ENO} & \quad \rightarrow \\
\text{EMP}_1 & \quad \rightarrow \\
\text{EMP}_2 & \quad \rightarrow \\
\text{EMP}_1 & \quad \rightarrow \\
\end{align*}
\]
Reduction for DHF

■ Rule:
  ➔ Distribute joins over unions
  ➔ Apply the join reduction for horizontal fragmentation

■ Example

\[
\begin{align*}
\text{ASG}_1 &: \text{ASG} \bowtie_{\text{ENO}} \text{EMP}_1 \\
\text{ASG}_2 &: \text{ASG} \bowtie_{\text{ENO}} \text{EMP}_2 \\
\text{EMP}_1 &: \sigma_{\text{TITLE}=\text{"Programmer"}} \left( \text{EMP} \right) \\
\text{EMP}_2 &: \sigma_{\text{TITLE} \neq \text{"Programmer"}} \left( \text{EMP} \right)
\end{align*}
\]

Query

\[
\begin{align*}
\text{SELECT} & \quad * \\
\text{FROM} & \quad \text{EMP, ASG} \\
\text{WHERE} & \quad \text{ASG.ENO} = \text{EMP.ENO} \\
\text{AND} & \quad \text{EMP.TITLE} = \text{"Mech. Eng."}
\end{align*}
\]
Reduction for DHF

Generic query

\[ \bigcup_{\text{ASG}_1} \bigcup_{\text{ASG}_2} \bigcup_{\text{EN}_0} \]

\[ \sigma_{\text{TITLE}=\text{"Mech. Eng."}} \]

Selections first

\[ \bigcup_{\text{ASG}_1} \bigcup_{\text{ASG}_2} \bigcup_{\text{EN}_0} \]

\[ \sigma_{\text{TITLE}=\text{"Mech. Eng."}} \]
Reduction for DHF

Joins over unions

Elimination of the empty intermediate relations
(left sub-tree)
Reduction for HF

- Combine the rules already specified:

  - Remove empty relations generated by contradicting selections on horizontal fragments;

  - Remove useless relations generated by projections on vertical fragments;

  - Distribute joins over unions in order to isolate and remove useless joins.
Reduction for HF

Example

Consider the following hybrid fragmentation:

\[ \text{EMP}_1 = \sigma_{\text{ENO} \leq \text{E}4} (\Pi_{\text{ENO, ENAME}} (\text{EMP})) \]

\[ \text{EMP}_2 = \sigma_{\text{ENO} > \text{E}4} (\Pi_{\text{ENO, ENAME}} (\text{EMP})) \]

\[ \text{EMP}_3 = \Pi_{\text{ENO, TITLE}} (\text{EMP}) \]

and the query

\[ \text{SELECT} \quad \text{ENAME} \]
\[ \text{FROM} \quad \text{EMP} \]
\[ \text{WHERE} \quad \text{ENO} = \text{E}5 \]