Distributed Databases

by Farnoush Banaei-Kashani

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

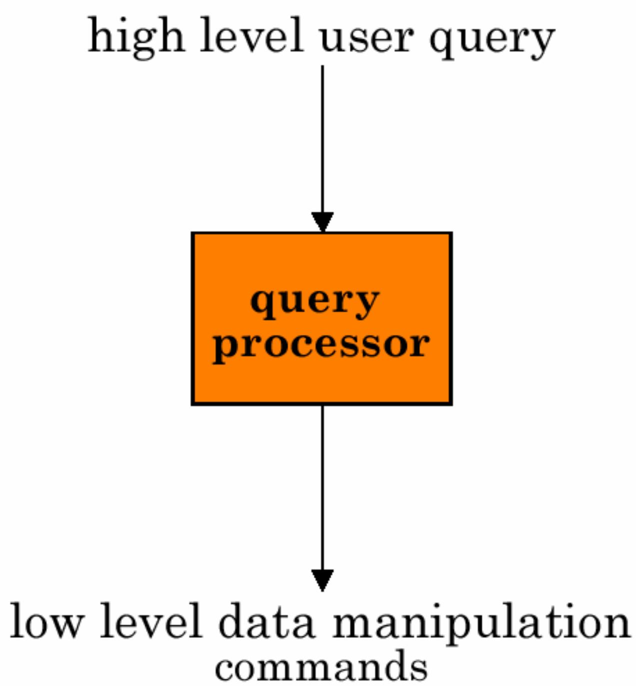
Topics

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Semantic Data Control
- $\boxed{\text{Distributed Query Processing}}$
- 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



Problem?

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

Strategy 1

$$\Pi_{ENAME}(\sigma_{DUR>37 \wedge EMP.ENO=ASG.ENO}(EMP \times ASG))$$

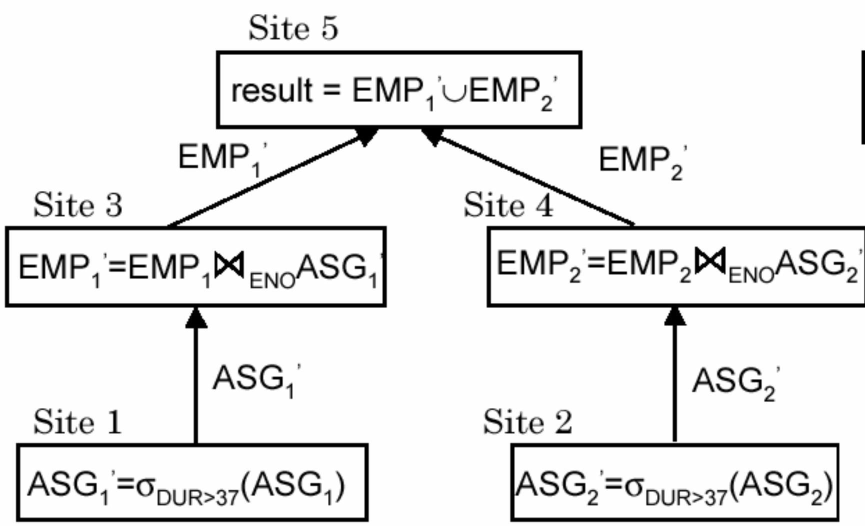
Strategy 2

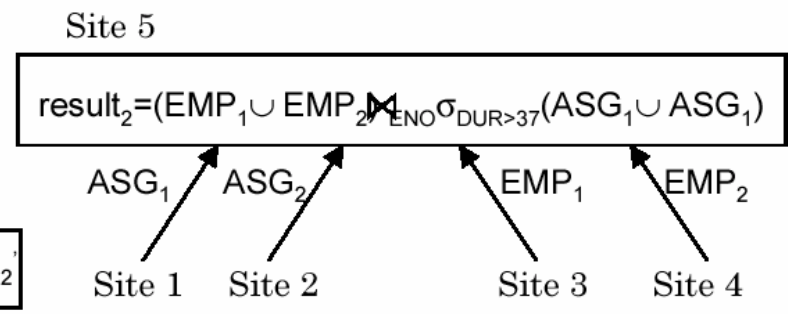
$$\Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{DUR>37}(ASG)))$$

Strategy 2 avoids Cartesian product, so is "better"

Problem in DDBS?

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





Problem in DDBS?

- Assume:
  - *size*(EMP) = 400, *size*(ASG) = 1000
  - tuple access cost = 1 unit; tuple transfer cost = 10 units
- Strategy 1

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

- Strategy 2

❶	transfer EMP to site 5:400∗tuple transfer cost	4,000
❷	transfer ASG to site 5 :1000∗tuple transfer cost	10,000
❸	produce ASG':1000∗tuple access cost	1,000
❹	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

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	$O(n^2)$

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

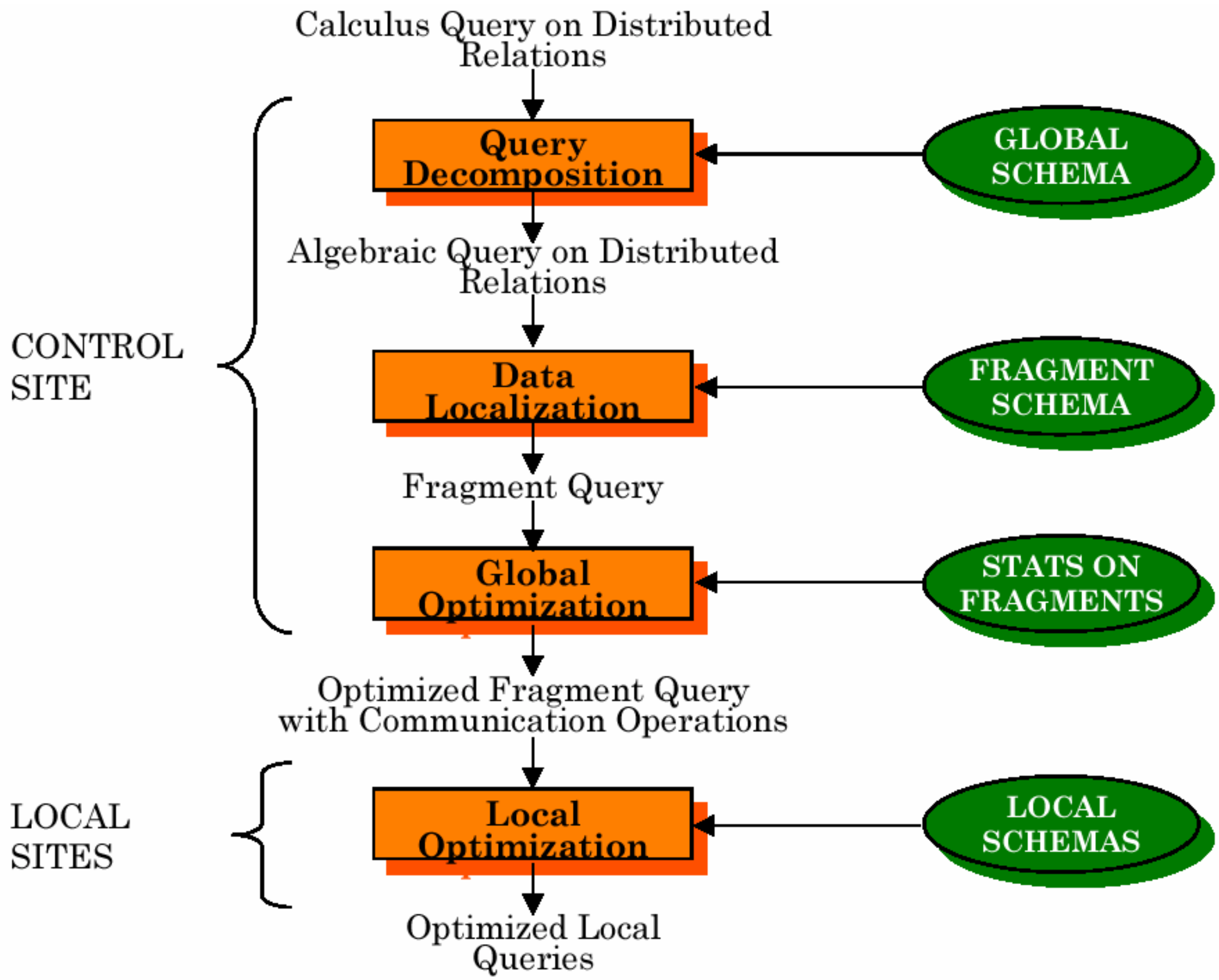
Query Processing Issues – Network Topology

- 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



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

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

    $$(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$$
  - Disjunctive normal form

    $$(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$$
  - OR's mapped into union
  - AND's mapped into join or selection

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $p_1 \wedge (p_2 \wedge p_3) \Leftrightarrow (p_1 \wedge p_2) \wedge p_3$
4. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
5. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
6. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$
7. $\neg(p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$
8. $\neg(p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$
9. $\neg(\neg p) \Leftrightarrow p$

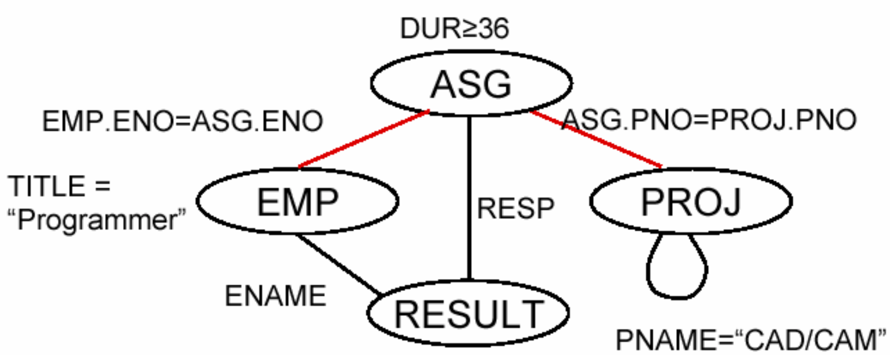
Analysis

- 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 ≥ 36
AND      TITLE = "Programmer"
```

Query graph

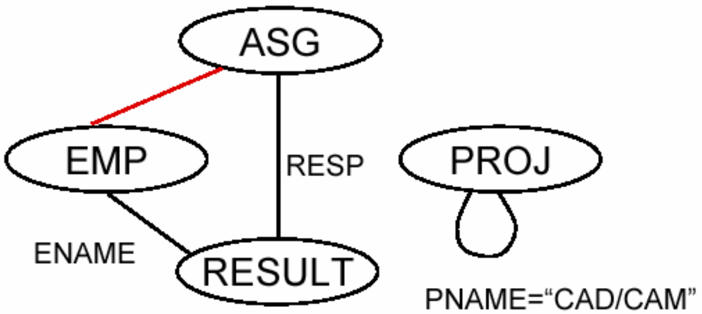


Join graph

Analysis - Example

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

      $p_1 \wedge \neg(p_1) \Leftrightarrow \text{false}$

      $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$

      $p_1 \vee \text{false} \Leftrightarrow p_1$

      …
  - application of transitivity

1. $p \wedge p \Leftrightarrow p$
2. $p \vee p \Leftrightarrow p$
3. $p \wedge true \Leftrightarrow p$
4. $p \vee false \Leftrightarrow p$
5. $p \wedge false \Leftrightarrow false$
6. $p \vee true \Leftrightarrow true$
7. $p \wedge \neg p \Leftrightarrow false$
8. $p \vee \neg p \Leftrightarrow true$
9. $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
10. $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

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."))
```

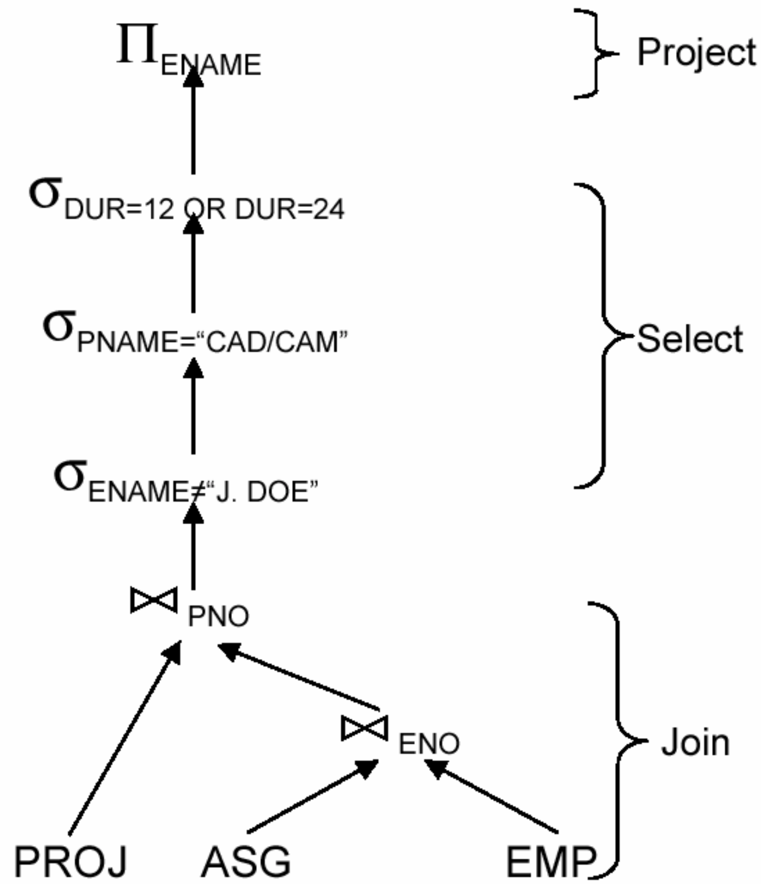
```
SELECT   TITLE
FROM     EMP
WHERE    EMP.ENAME = "J. Doe"
```

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
  - $\Pi_{A'}(\Pi_{A'}(\mathrm{R})) \Leftrightarrow \Pi_{A'}(\mathrm{R})$
  - $\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) = \sigma_{p_1(A_1) \wedge p_2(A_2)}(R)$

  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
  - $\sigma_{p(A)}(R \times S) \Leftrightarrow (\sigma_{p(A)}(R)) \times S$
  - $\sigma_{p(A_i)}(R \bowtie_{(A_j,B_k)} S) \Leftrightarrow (\sigma_{p(A_i)}(R)) \bowtie_{(A_j,B_k)} S$
  - $\sigma_{p(A_i)}(R \cup T) \Leftrightarrow \sigma_{p(A_i)}(R) \cup \sigma_{p(A_i)}(T)$

  where A_i belongs to R and T

- Commuting projection with binary operations
  - $\Pi_C(R \times S) \Leftrightarrow \Pi_{A'}(R) \times \Pi_{B'}(S)$
  - $\Pi_C(R \bowtie_{(A_j,B_k)} S) \Leftrightarrow \Pi_{A'}(R) \bowtie_{(A_j,B_k)} \Pi_{B'}(S)$
  - $\Pi_C(R \cup S) \Leftrightarrow \Pi_C(R) \cup \Pi_C(S)$

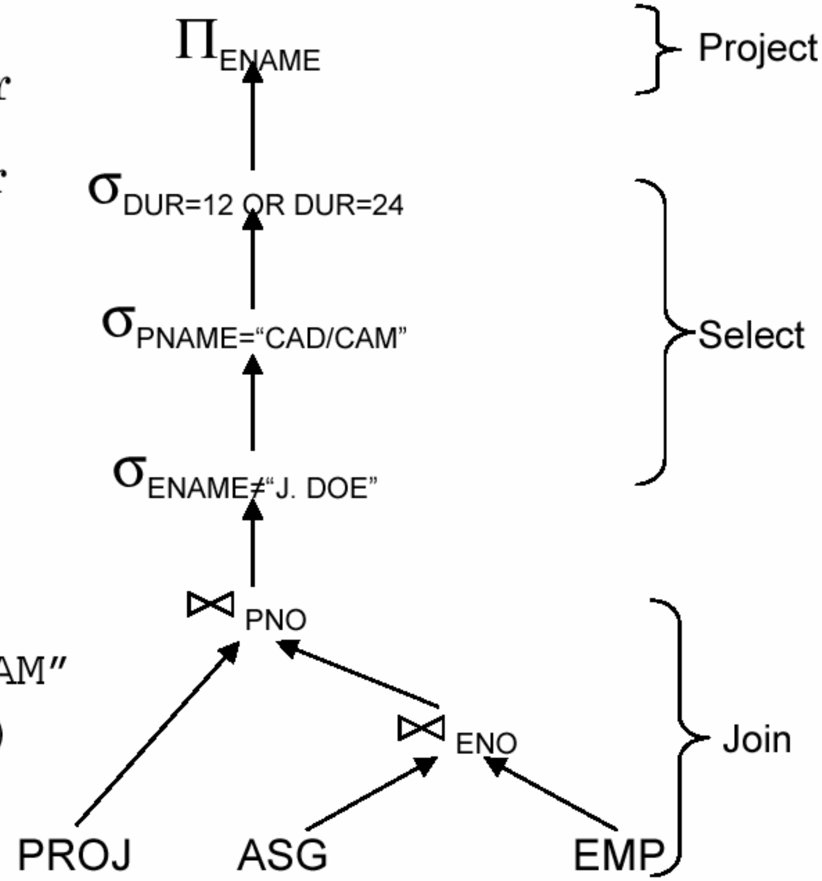
  where $R[A]$ and $S[B]$; $C = A' \cup B'$ where $A' \subseteq A$, $B' \subseteq B$

Example

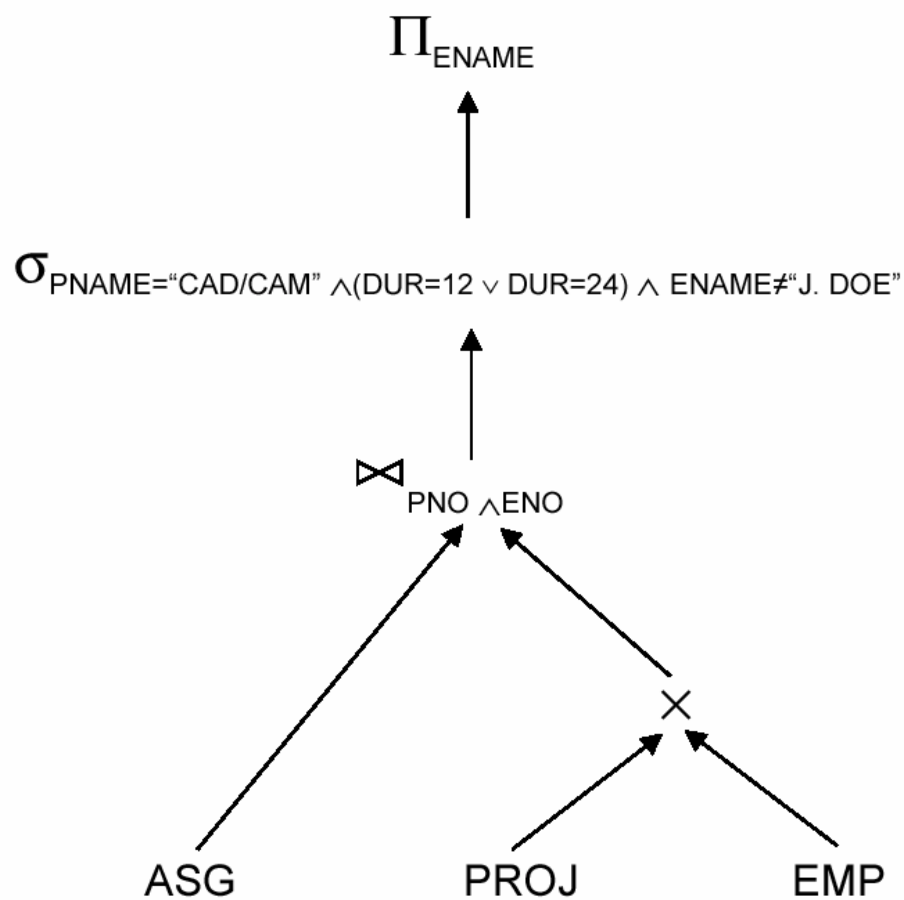
Recall the previous example:

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

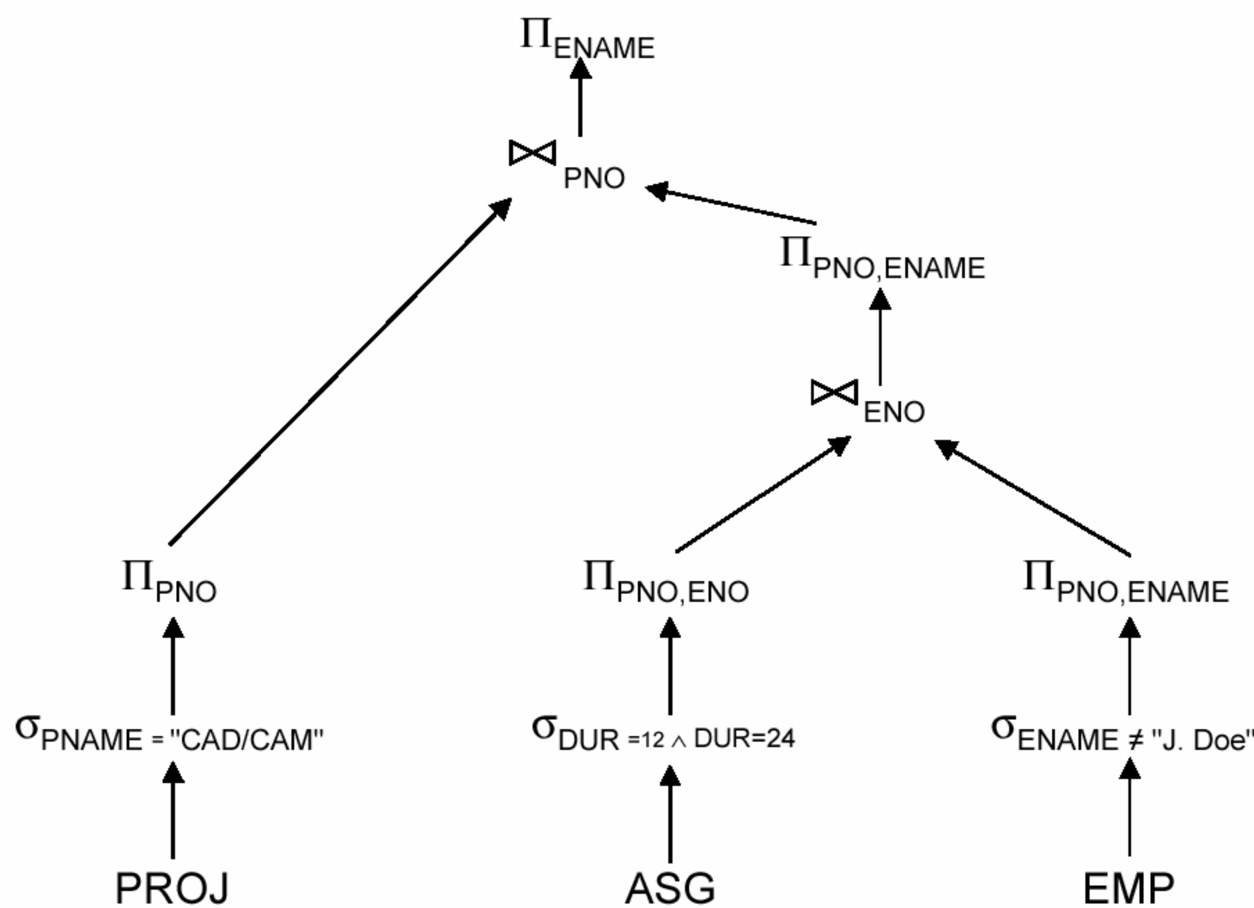
```
SELECT   ENAME
FROM     PROJ, ASG, EMP
WHERE    ASG.ENO=EMP.ENO
AND      ASG.PNO=PROJ.PNO
AND      ENAME≠"J. Doe"
AND      PROJ.PNAME="CAD/CAM"
AND      (DUR=12 OR DUR=24)
```



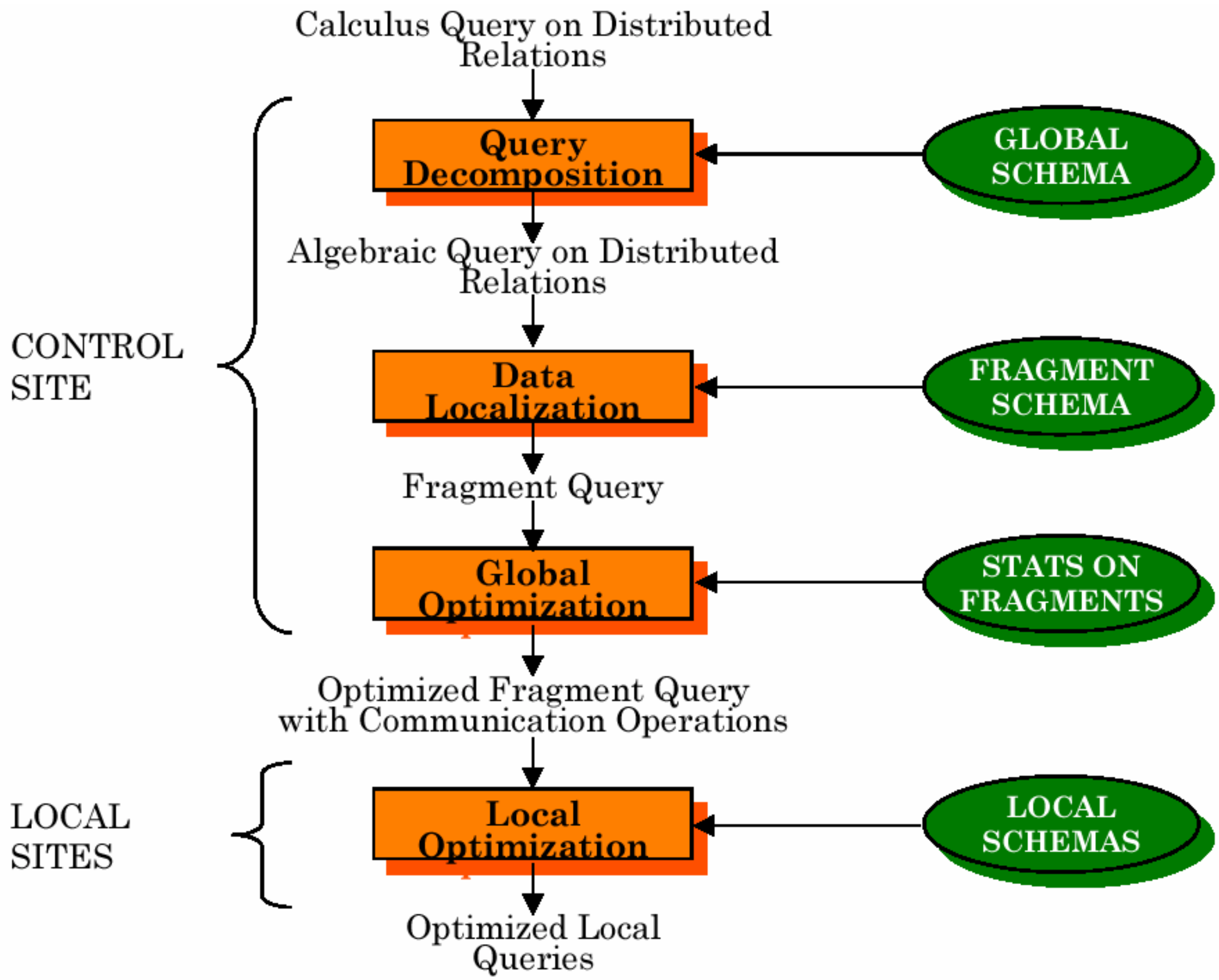
Equivalent Query



Restructuring



Distributed Query Processing Methodology



Step 2 – Data Localization

Input: Algebraic query on distributed relations

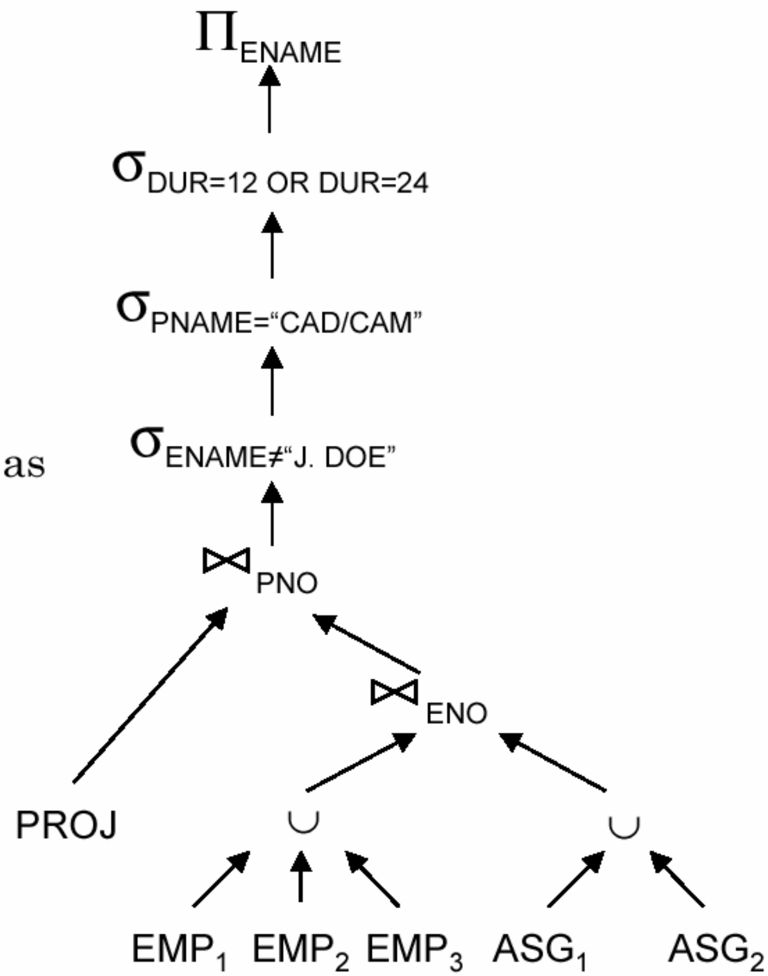
- Determine which fragments are involved
- Localization program
  - substitute for each global query its materialization program
  - optimize

Example

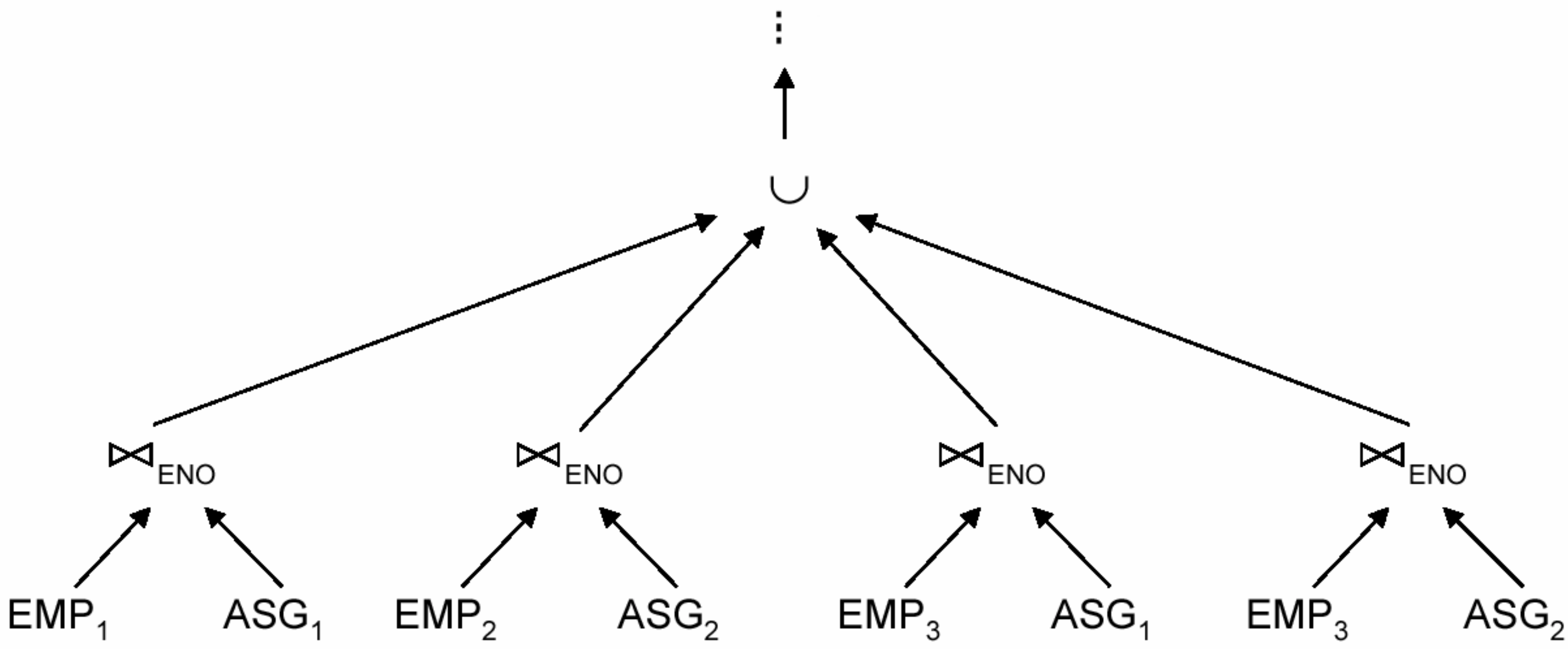
Assume

- EMP is fragmented into EMP_1, EMP_2, EMP_3 as follows:
  - $EMP_1 = \sigma_{ENO \leq \text{"E3"}}(EMP)$
  - $EMP_2 = \sigma_{\text{"E3"} < ENO \leq \text{"E6"}}(EMP)$
  - $EMP_3 = \sigma_{ENO \geq \text{"E6"}}(EMP)$
- ASG fragmented into ASG_1 and ASG_2 as follows:
  - $ASG_1 = \sigma_{ENO \leq \text{"E3"}}(ASG)$
  - $ASG_2 = \sigma_{ENO > \text{"E3"}}(ASG)$

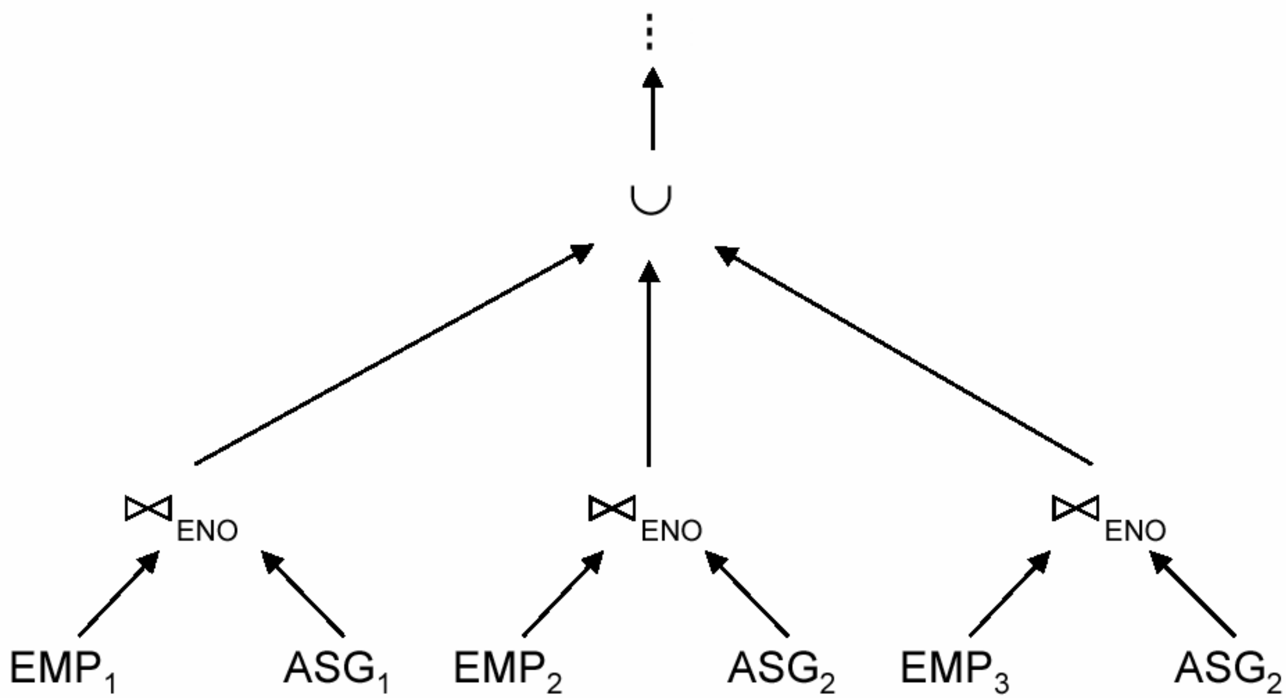
Replace EMP by ($EMP_1 \cup EMP_2 \cup EMP_3$) and ASG by ($ASG_1 \cup ASG_2$) in any query



Provides Parallelism



Eliminates Unnecessary Work

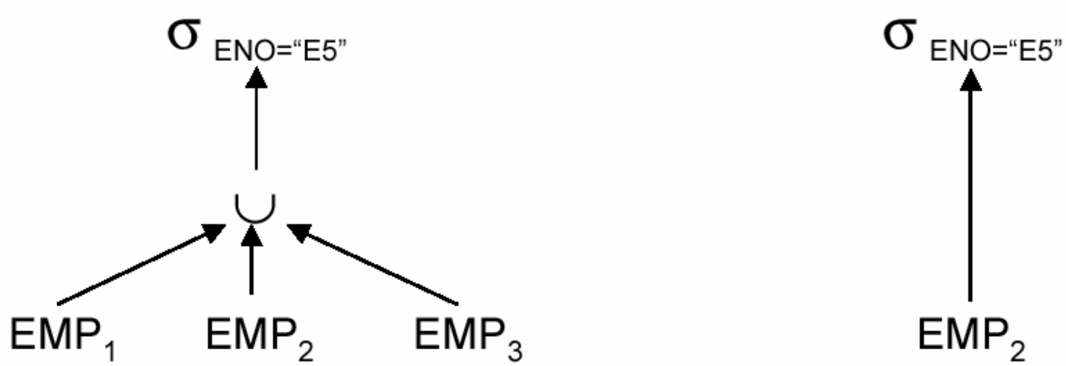


Reduction for PHF

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

    $\sigma_{p_i}(R_j)=\phi$ if $\forall x$ in R: $\neg(p_i(x) \wedge p_j(x))$
  - Example

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



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 $R_j = \sigma_{p_j}(R)$

    $$R_i \bowtie R_j = \phi \text{ if } \forall x \text{ in } R_i, \forall y \text{ in } R_j\text{: } \neg(p_i(x) \wedge p_j(y))$$

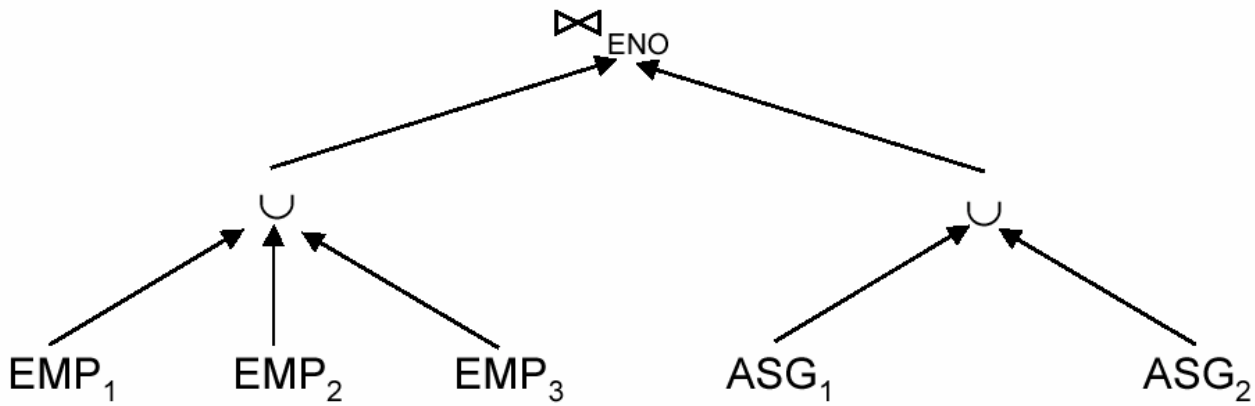
Reduction for PHF

- Reduction with join - Example
  - Assume EMP is fragmented as before and

    ASG_1: $\sigma_{ENO \leq "E3"}(ASG)$

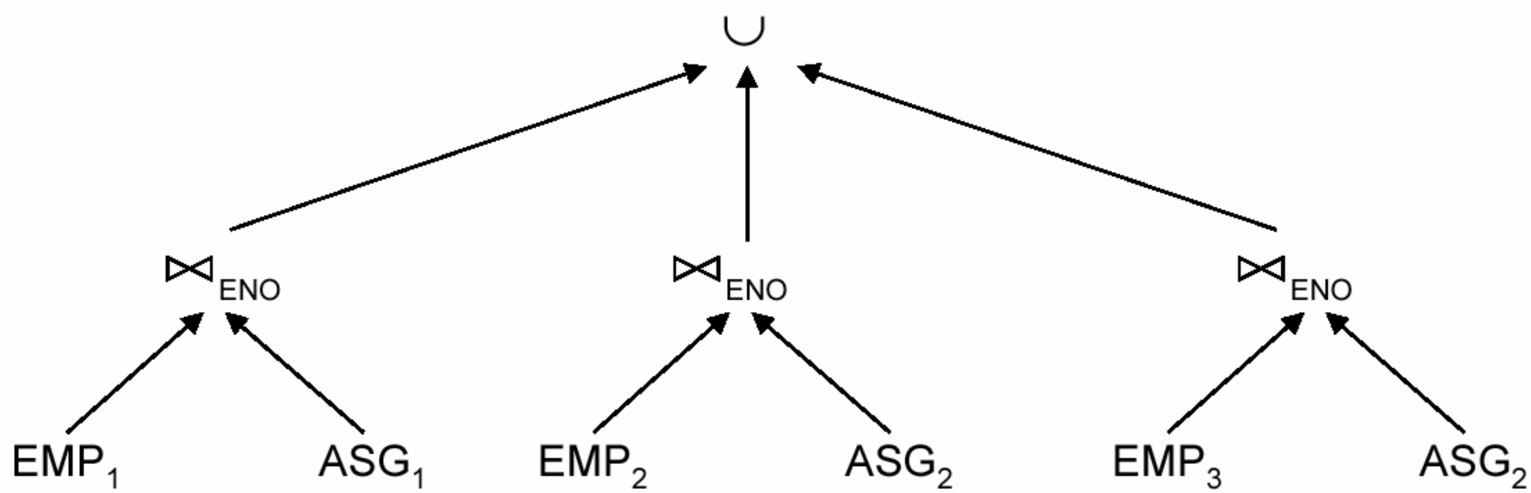
    ASG_2: $\sigma_{ENO > "E3"}(ASG)$
  - Consider the query

```
SELECT*
FROM   EMP, ASG
WHERE  EMP.ENO=ASG.ENO
```



Reduction for PHF

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



Reduction for VF

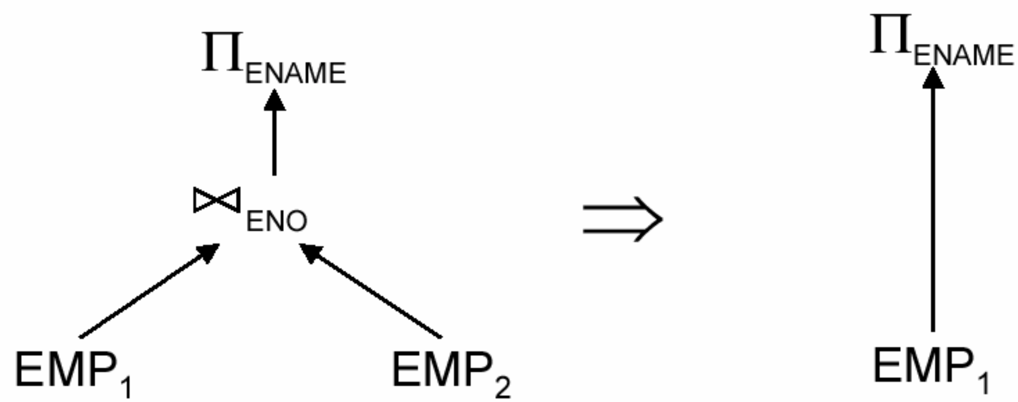
- Find useless (not empty) intermediate relations

  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: $EMP_1 = \Pi_{ENO,ENAME}(EMP)$; $EMP_2 = \Pi_{ENO,TITLE}(EMP)$

```
SELECT  ENAME
FROM    EMP
```



Reduction for DHF

- Rule :
  - Distribute joins over unions
  - Apply the join reduction for horizontal fragmentation
- Example

ASG_1: ASG $\ltimes_{ENO}$ EMP_1

ASG_2: ASG $\ltimes_{ENO}$ EMP_2

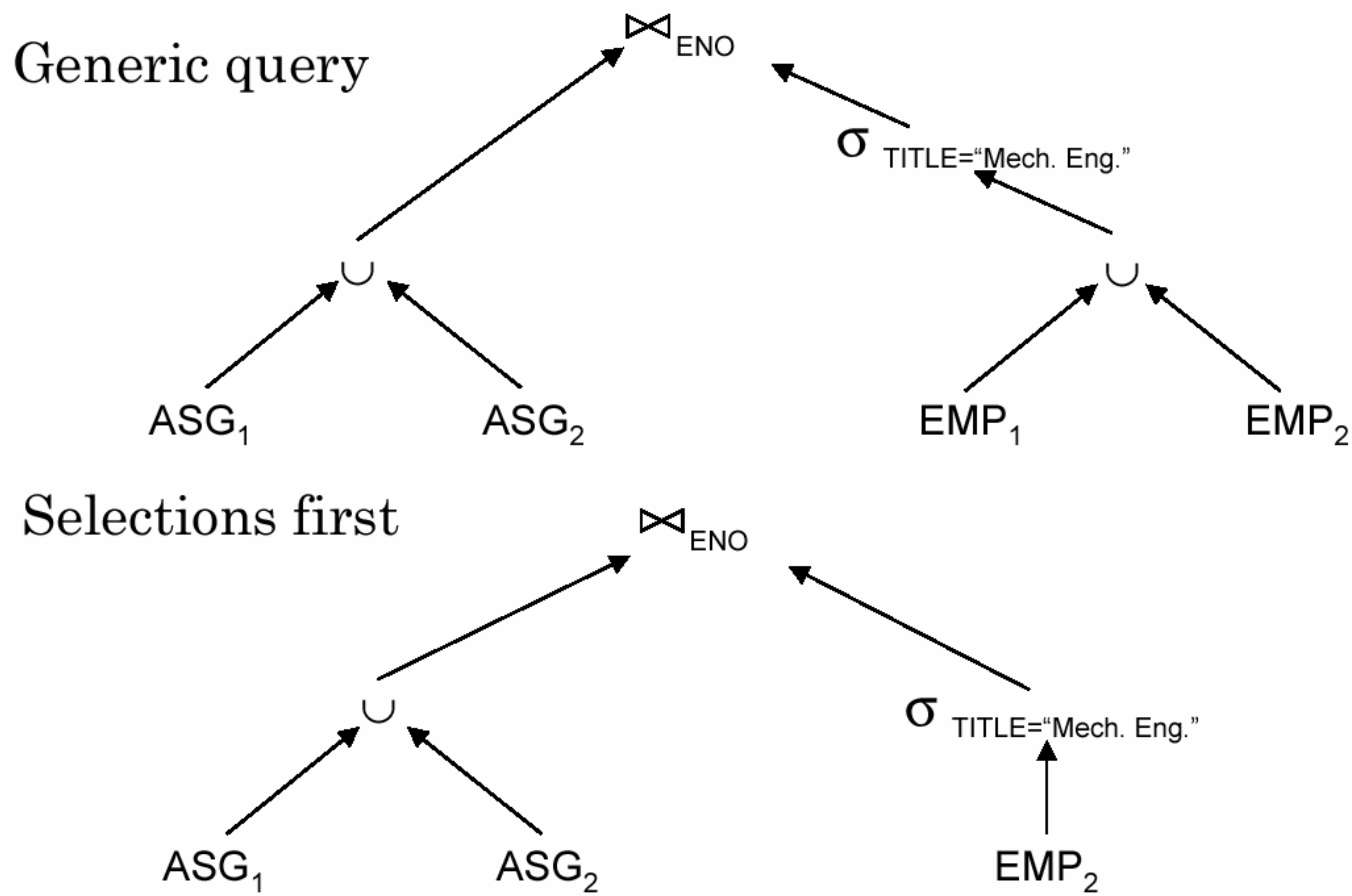
EMP_1: $\sigma_{TITLE=\text{“Programmer”}}(EMP)$

EMP_2: $\sigma_{TITLE\neq\text{“Programmer”}}(EMP)$

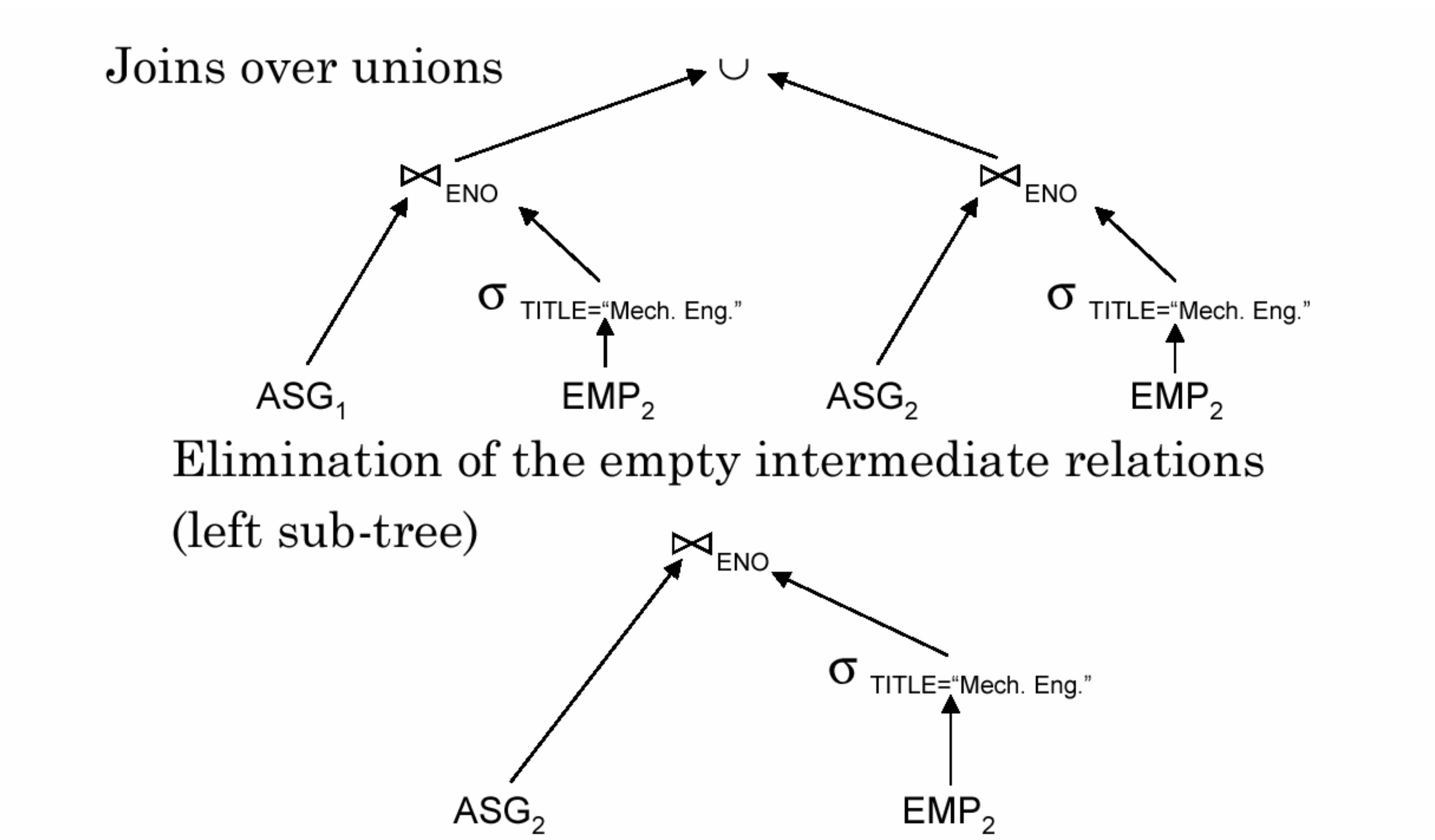
Query

```
SELECT    *
FROM      EMP, ASG
WHERE     ASG.ENO = EMP.ENO
AND       EMP.TITLE = "Mech. Eng."
```

Reduction for DHF



Reduction for DHF



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:

$EMP_1 = \sigma_{ENO \leq "E4"} (\Pi_{ENO,ENAME} (EMP))$

$EMP_2 = \sigma_{ENO > "E4"} (\Pi_{ENO,ENAME} (EMP))$

$EMP_3 = \Pi_{ENO,TITLE} (EMP)$

and the query

```
SELECT    ENAME
FROM      EMP
WHERE     ENO="E5"
```