+

Lecture Notes
Week 3

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

INFS3200 Advanced Database Systems

Semester 1, 2021

Distributed Query Processing

Professor Xue Li

+ Last Week

- Why distributed databases
- Different levels of transparency
- Different levels of schemas
- Three dimensions of distributed database systems
- DDB design
  - Fragmentation, replication, allocation

+ Allocation

- Input
  - Fragments: F={F1,F2,…,Fm} Sites: S={S1,S2,…,Sn}
  - Typical queries: Q={Q1,Q2,…,Qk} detailed with read/write information
- An allocation can be represented by a group of mappings:
  - Xij = 1 if Fi is assigned to Sj; otherwise Xij=0

Total_cost = total_local_processing_cost + total_data_exchange_cost + total_stoarge_cost

- The allocation problem is to find an optimal mapping to minimize the total cost, with challenges: (1) NP-complete, (2) hard to estimate precise costs, (3) changing costs over time
- Solutions: heuristics-based with many simplifications (e.g., communication costs only), supporting dynamic adjustment

+ Distributed Databases

- Distributed Databases Concepts
- Distributed Database Data Storage
- Distributed Query Processing
- Distributed Transaction Management

+ Learning Objectives

- Objectives, overall framework, required information and general strategies of distributed query processing
- Query Processing
  - Query Decomposition
  - Data Localization
  - Processing Optimization

+ Query Optimization Objectives

What is the best we can do?

- 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 may dominate or vary much
    - Bandwidth, speed, high protocol overhead
- Local Area Networks
  - Communication cost not that dominant
  - Total cost function should be considered

+ Example Database

-

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

+ Selecting Alternatives


```
SELECT    ENAME
FROM      EMP,ASG
WHERE     EMP.ENO = ASG.ENO AND
          RESP = "Manager"
```

Strategy 1

$$\Pi_{\text{ENAME}}(\sigma_{\text{RESP=“Manager”} \wedge \text{EMP.ENO=ASG.ENO}}(\text{EMP} \times \text{ASG}))$$

Strategy 2

$$\Pi_{\text{ENAME}}(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{RESP=“Manager”}} (\text{ASG}))$$

+ What is the problem?

```
SELECT ENAME
FROM   EMP,ASG
WHERE  EMP.ENO = ASG.ENO AND
             RESP = "Manager"
```

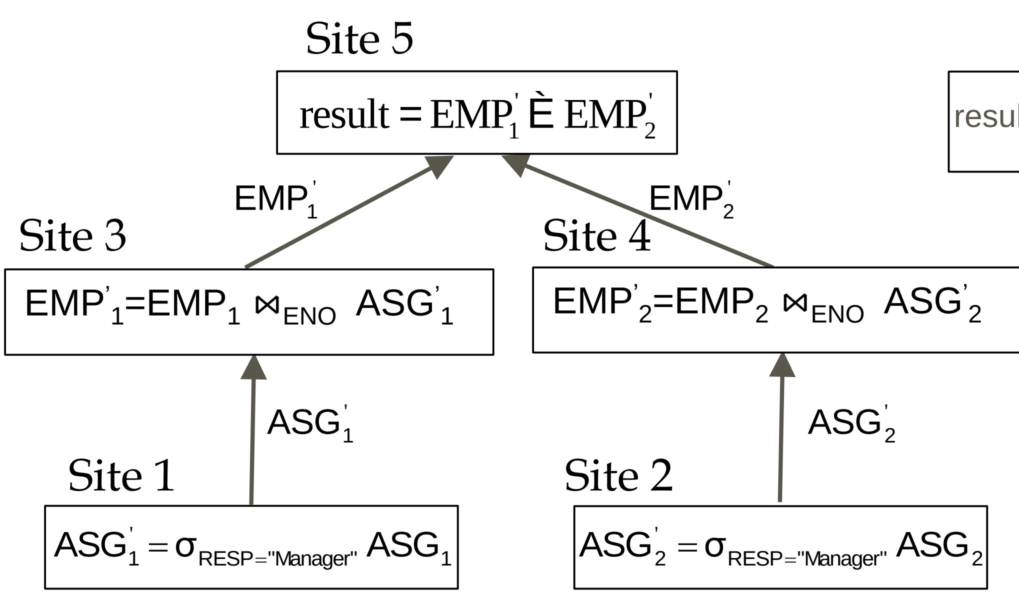
EMP

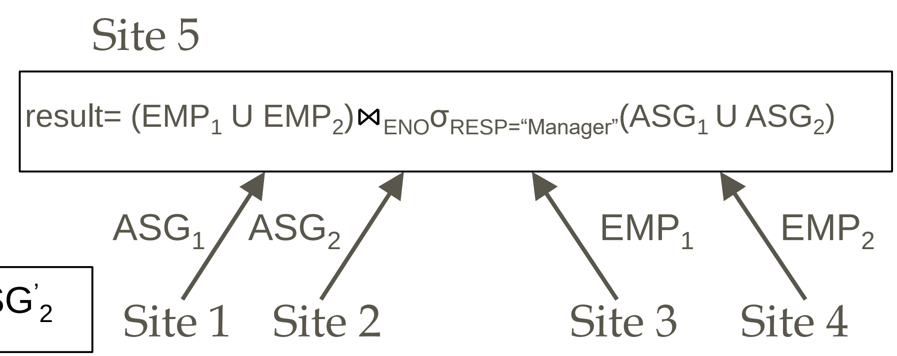
ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

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





+ Types of Optimizers

- Exhaustive search (Model based)
  - Cost model based
  - Optimal
  - Hard to have all system performance data required by cost models
  - Combinatorial complexity in the number of relations
- Heuristics (Rule based)
  - Not optimal
  - Perform selection, projection first, …
  - Reorder operations to reduce intermediate relation size
  - Replace a join by a series of semi-joins?

+ Optimization Granularity

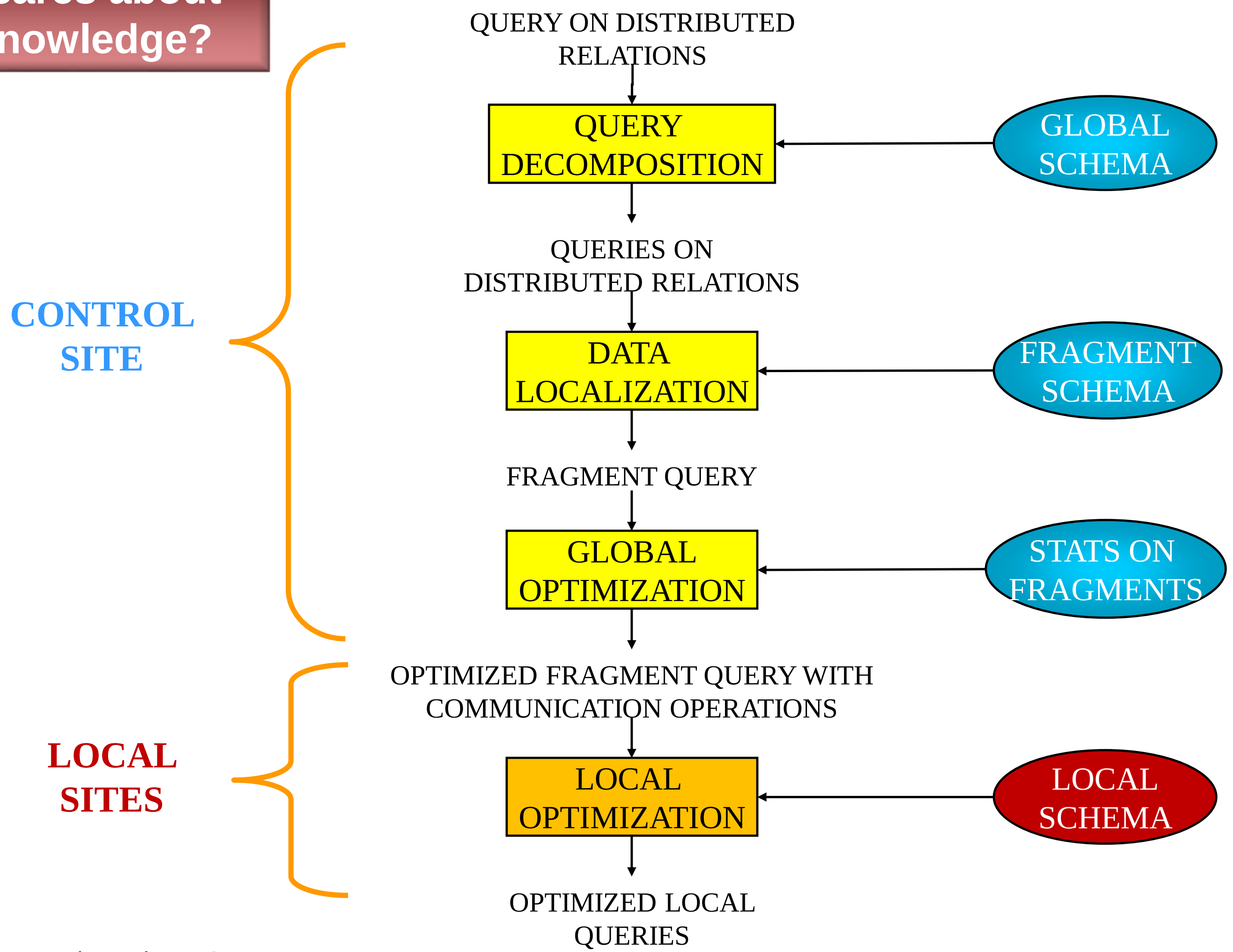
- Single query at a time
  - Cannot use common intermediate results
- Multiple queries at a time
  - Efficient if many similar queries
  - Decision space is much larger

+ Statistics

- Relation
  - Cardinality, size of a tuple…
- Attribute
  - Number of distinct values, selectivity…
- Common assumptions
  - Independence between different attribute values
  - Uniform distribution of attribute values within their domain

+ Layers of Query Processing

Who cares about this knowledge?



+ Query Decomposition

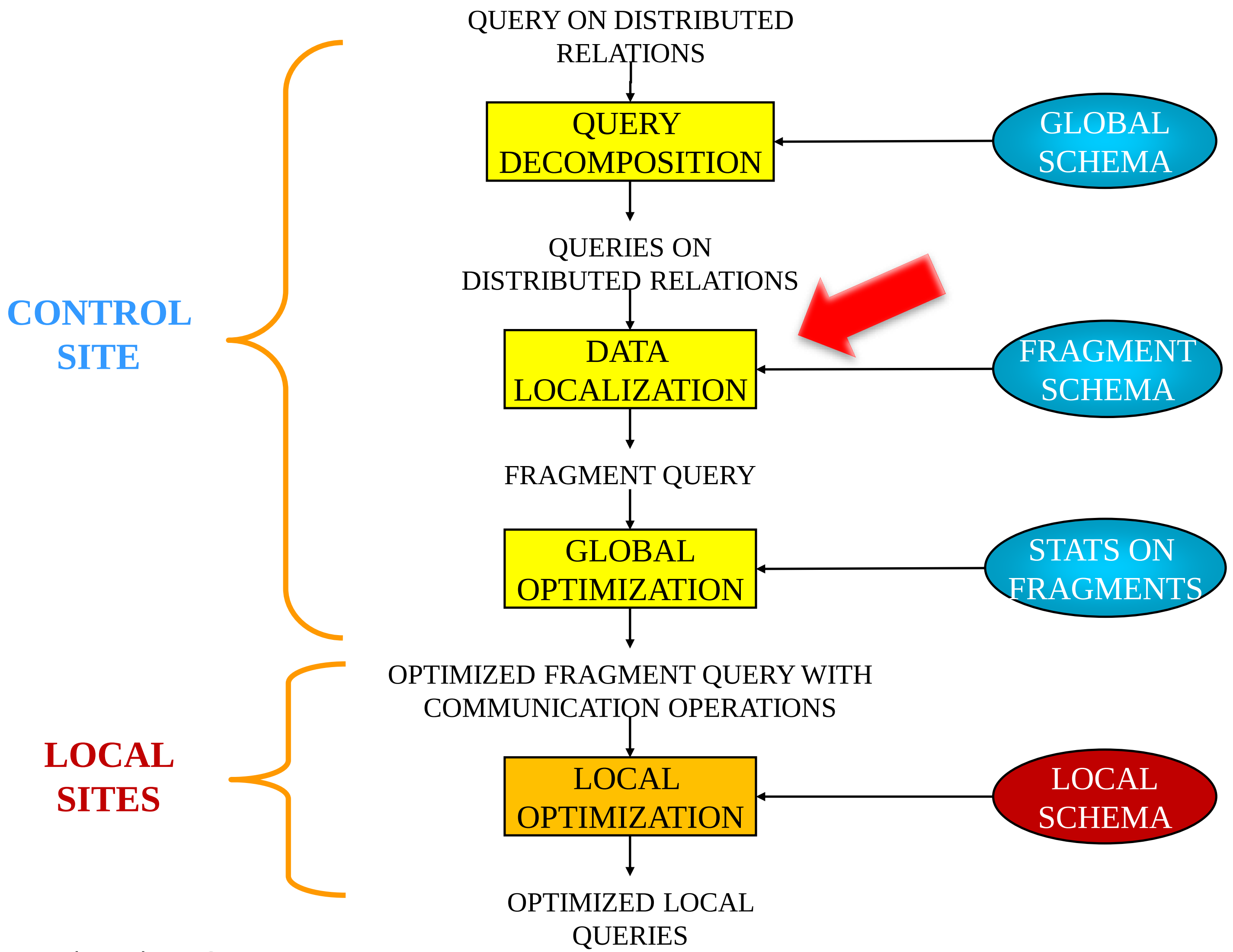
Same as in a centralized system, with 4 steps:

- Normalization: convert query to a standard form
  - SQL is based on relational calculus, which is non-procedural, in the form of {t | F(t)}
  - Relational algebra, a step-by-step program using set operations (σ, π, ⋈, ∩, ∪, x…)
- Analysis: make sure it is semantically correct
- Simplification: remove redundant predicates
- Rewriting: generate a "good" algebraic query

What is the difference between Relational Algebra and Relational Calculus?

https://www.geeksforgeeks.org/difference-between-relational-algebra-and-relational-calculus/

+ Layers of Query Processing



+ Data Localization

Input: Algebraic query on distributed relations

- Two steps:
  - Use fragments to replace relations in the query
  - Simplify and restructure to get another “good” query
- Note
  - Fragmentation is defined using relational operations, thus data localization is similar to view-based query transformation
  - May need to add union ($\cup$) and join ($\bowtie$) operations
    - Reorder operations: push up $\cup$ and $\bowtie$, push down σ (examples later)
    - Simplify operations: eliminate unnecessary operations

+ Example Database

Find the names of employees other than J. Doe who worked on the CAD/CAM project for either 1 or 2 years.

-

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

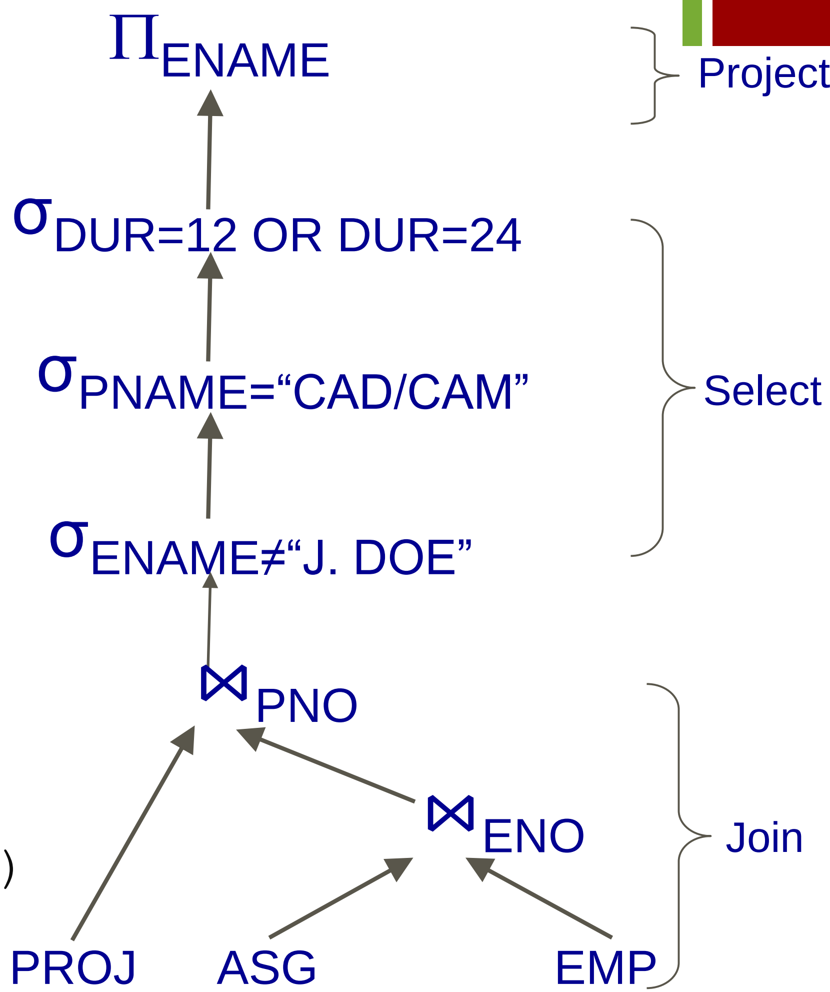
PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

+ Example - Query

Find the names of employees other than J. Doe who worked on the CAD/CAM project for either 1 or 2 years.

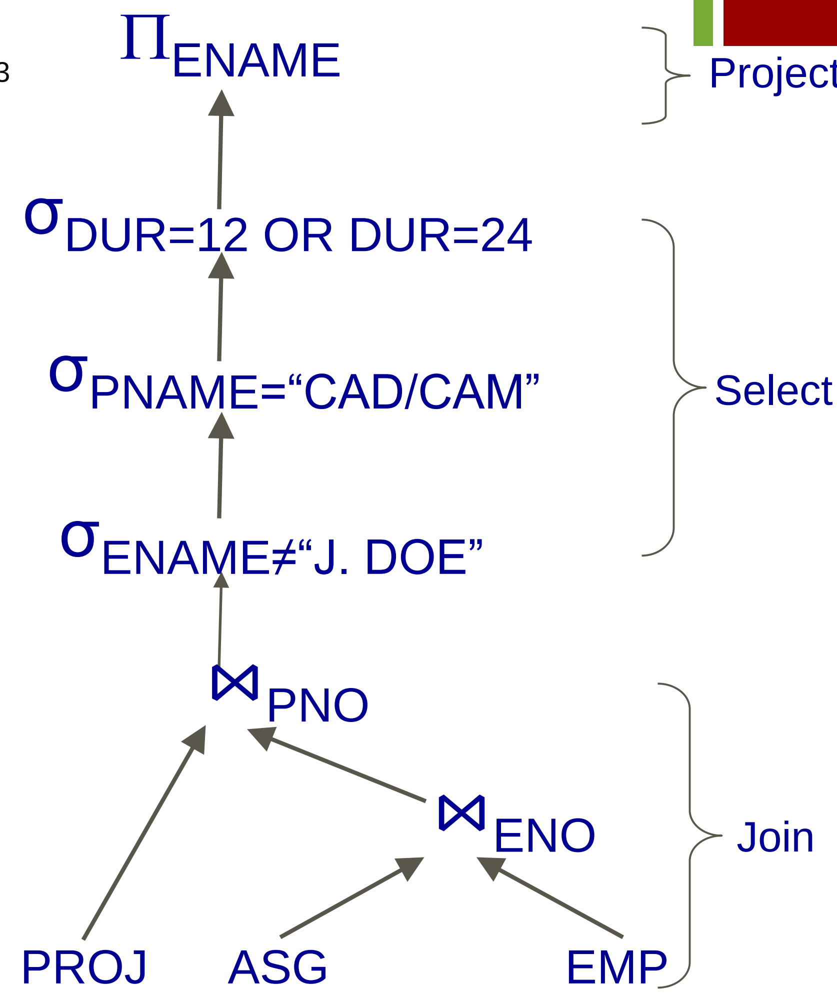
```
SELECT  ENAME
FROM    EMP, ASG, PROJ
WHERE   EMP.ENO = ASG.ENO
AND     ASG.PNO = PROJ.PNO
AND     ENAME ≠ "J. Doe"
AND     PNAME = "CAD/CAM"
AND     (DUR = 12 OR DUR = 24)
```



+ Example - Fragmentation

Assume

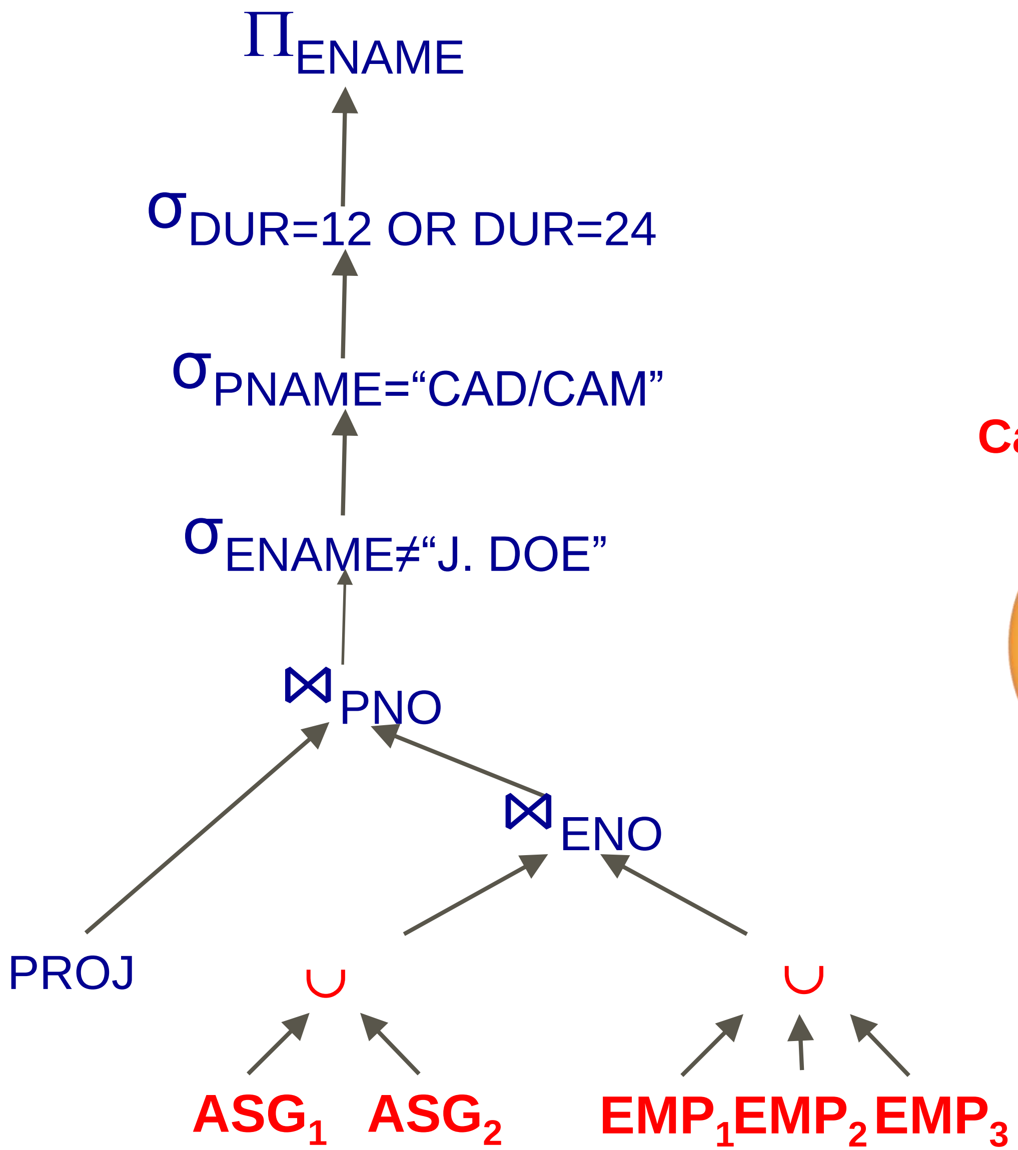
- EMP is fragmented into EMP_1, EMP_2, EMP_3 as follows:
  - $EMP_1 = \sigma_{ENO \leq \text{"E3"}}(EMP)$
  - $EMP_2 = \sigma_{\text{"E3"} < ENO \leq \text{"E6"}}(EMP)$
  - $EMP_3 = \sigma_{ENO > \text{"E6"}}(EMP)$
- ASG fragmented into ASG_1 and ASG_2 as follows:
  - $ASG_1 = \sigma_{ENO \leq \text{"E3"}}(ASG)$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



Replace: EMP by ($EMP_1 \cup EMP_2 \cup EMP_3$)

ASG by ($ASG_1 \cup ASG_2$) in any query

+ Example – Using Fragmentation



Can we do better?

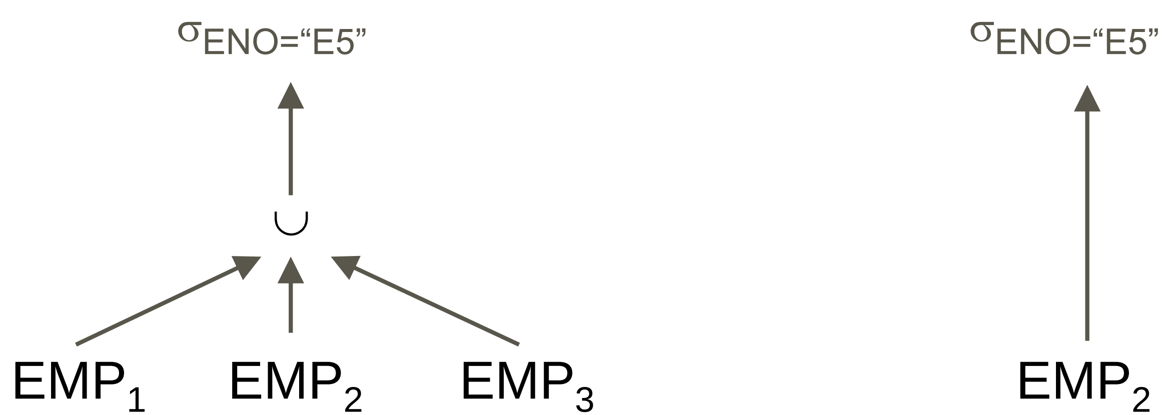
+ Reduction for HF: Selection

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

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

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

- $EMP_1 = \sigma_{ENO \leq \text{“E3”}}(EMP)$
- $EMP_2 = \sigma_{\text{“E3”} < ENO \leq \text{“E6”}}(EMP)$
- $EMP_3 = \sigma_{ENO > \text{“E6”}}(EMP)$



+ Reduction for HF: Join

- Reduction with join
  - Possible when fragmentation is done on join attributes
  - Distribute join over union

$$(R_1 \cup R_2) \bowtie S \Leftrightarrow (R_1 \bowtie S) \cup (R_2 \bowtie S)$$

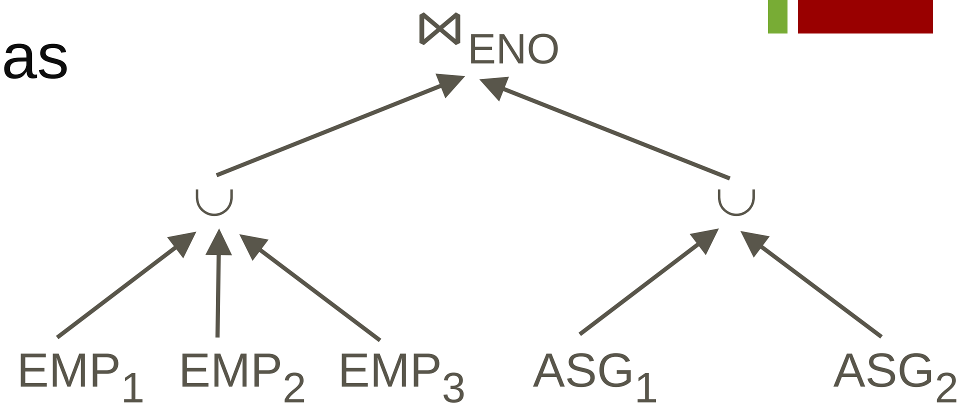
  - Given $R_i = \sigma_{p_i}(R)$ and $R_j = \sigma_{p_j}(R)$

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

+ 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
```



- $EMP_1 = \sigma_{ENO \leq "E3"}(EMP)$
- $EMP_2 = \sigma_{"E3" < ENO \leq "E6"}(EMP)$
- $EMP_3 = \sigma_{ENO > "E6"}(EMP)$

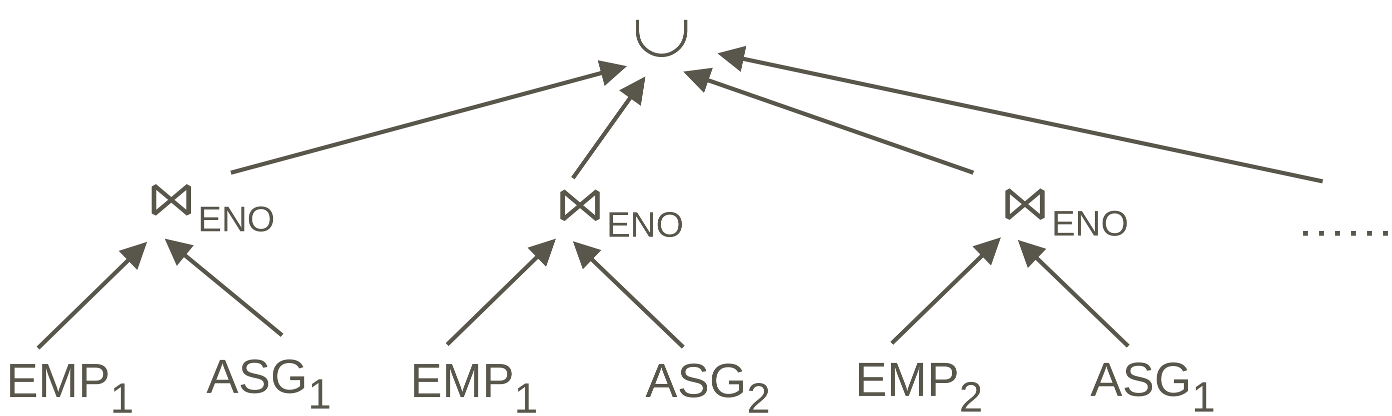
+ Example: Distribute Join over Unions

Rule: $(R1 \cup R2) \bowtie S \Leftrightarrow (R1 \bowtie S) \cup (R2 \bowtie S)$

$$(R_1 \cup R_2) \bowtie (S_1 \cup S_2) \Leftrightarrow (R_1 \bowtie S_1) \cup (R_1 \bowtie S_2) \cup (R_2 \bowtie S_1) \cup (R_2 \bowtie S_2)$$

$$(EMP_1 \cup EMP_2 \cup EMP_3) \bowtie (ASG_1 \cup ASG_2) =$$

$$EMP_1 \bowtie ASG_1 \cup EMP_1 \bowtie ASG_2 \cup$$
$$EMP_2 \bowtie ASG_1 \cup EMP_2 \bowtie ASG_2 \cup$$
$$EMP_3 \bowtie ASG_1 \cup EMP_3 \bowtie ASG_2$$



+ Example: Apply the Reduction Rule

Rule: Given $R_i = \sigma_{p_i}(R)$ and $R_j = \sigma_{p_j}(R)$

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

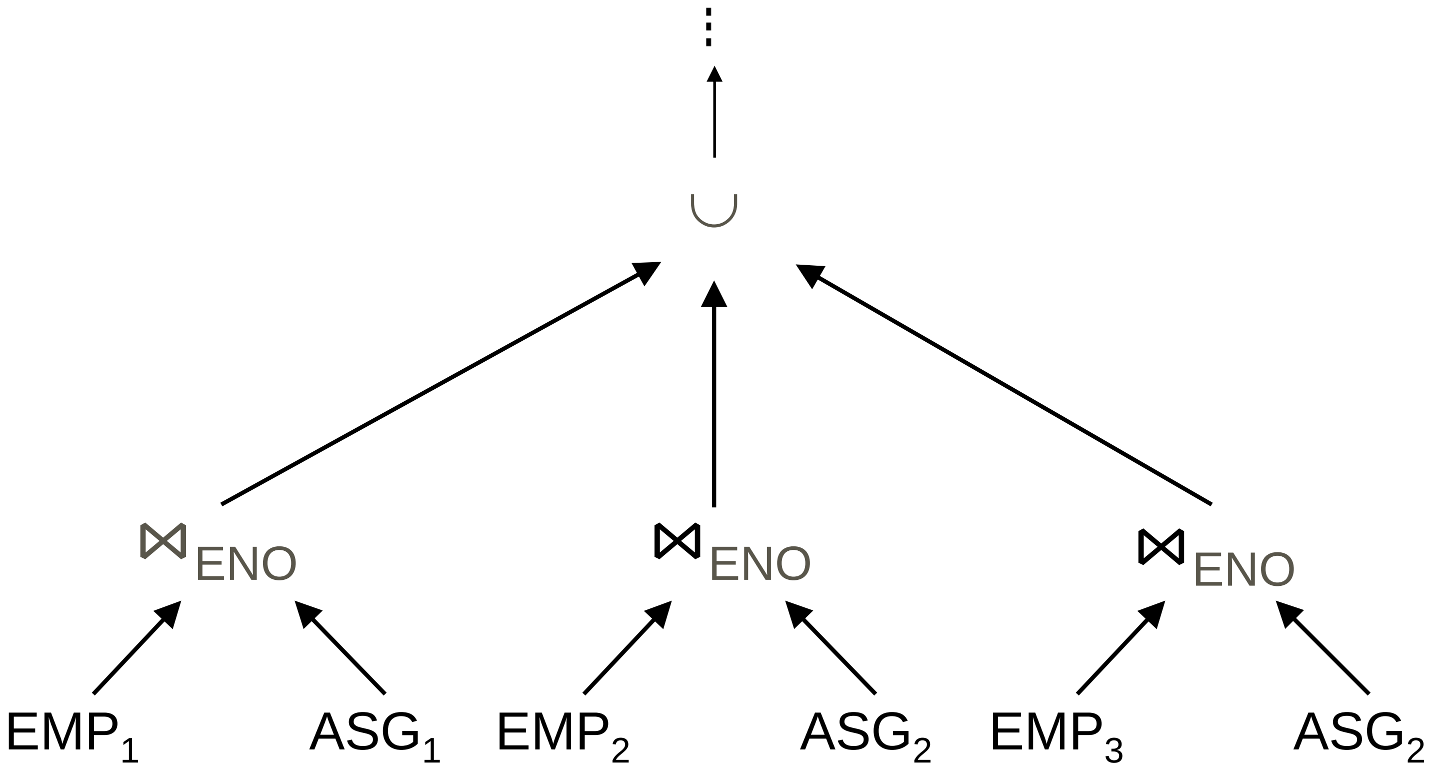
$EMP_1 = \sigma_{ENO \leq \text{"E3"}}(EMP)$
$EMP_2 = \sigma_{\text{"E3"} < ENO \leq \text{"E6"}}(EMP)$
$EMP_3 = \sigma_{ENO \geq \text{"E6"}}(EMP)$

$ASG_1 = \sigma_{ENO \leq \text{"E3"}}(ASG)$
$ASG_2 = \sigma_{ENO > \text{"E3"}}(ASG)$

$EMP_1 \bowtie ASG_1 \cup EMP_1 \bowtie ASG_2 \cup EMP_2 \bowtie ASG_1 \cup$
$EMP_2 \bowtie ASG_2 \cup EMP_3 \bowtie ASG_1 \cup EMP_3 \bowtie ASG_2$
$=$

$EMP_1 \bowtie ASG_1 \cup EMP_2 \bowtie ASG_2 \cup EMP_3 \bowtie ASG_2$

+ Example: Final Result



…also, opportunities for parallel processing!

+ Reduction for VF

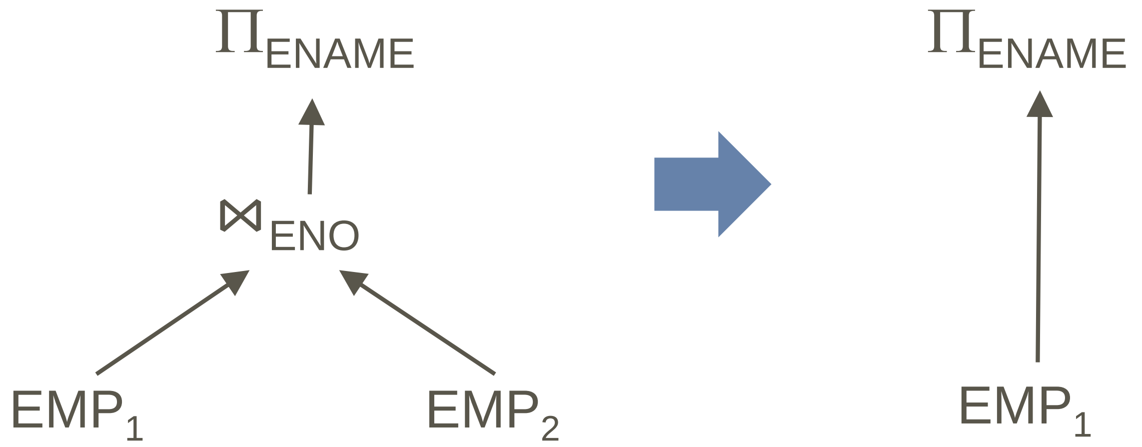
Find useless intermediate relations

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

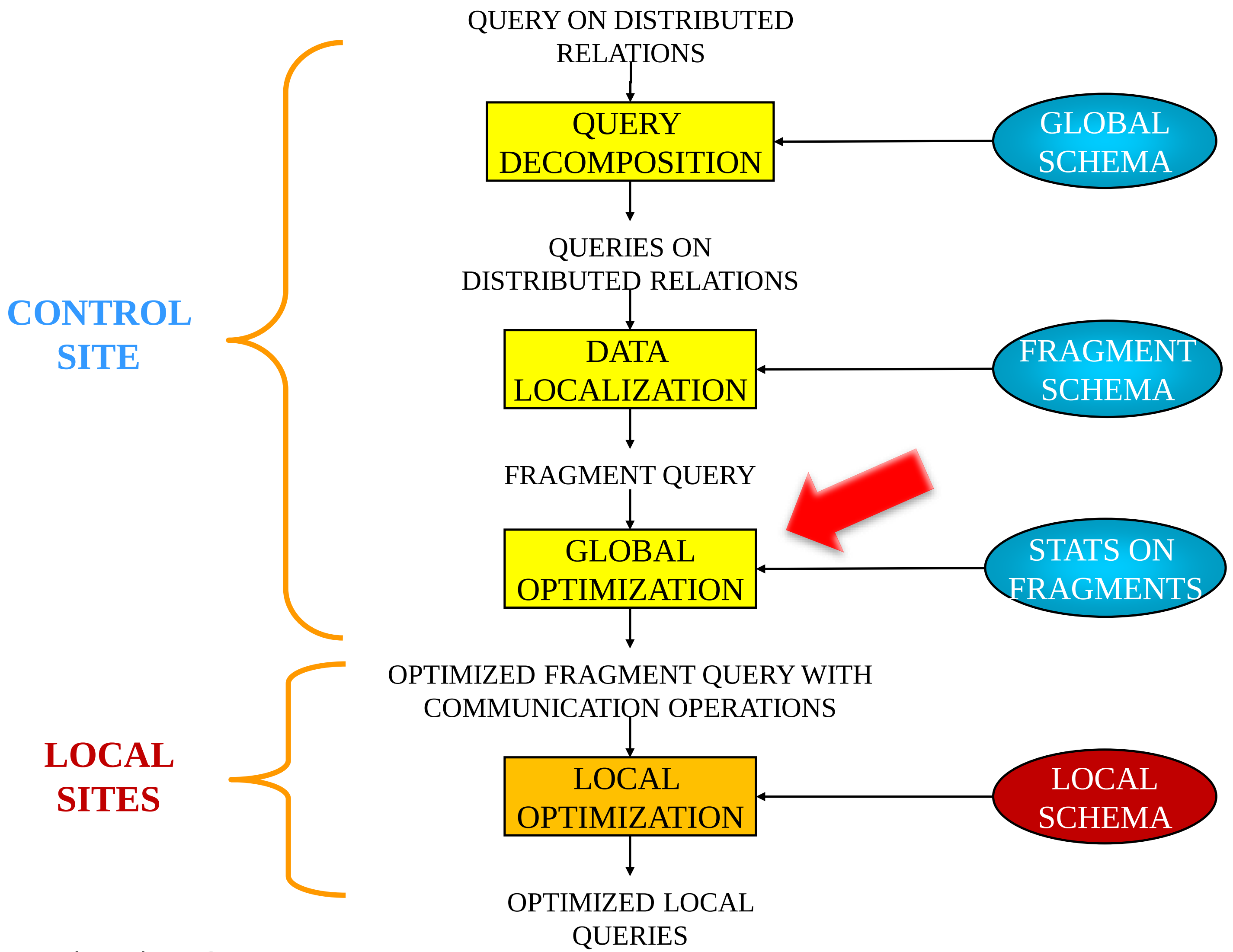
$\Pi_D(R_i)$ is useless if D is not in A'

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

```
SELECT  ENAME
FROM    EMP
```



+ Layers of Query Processing



+ Global Query Optimization

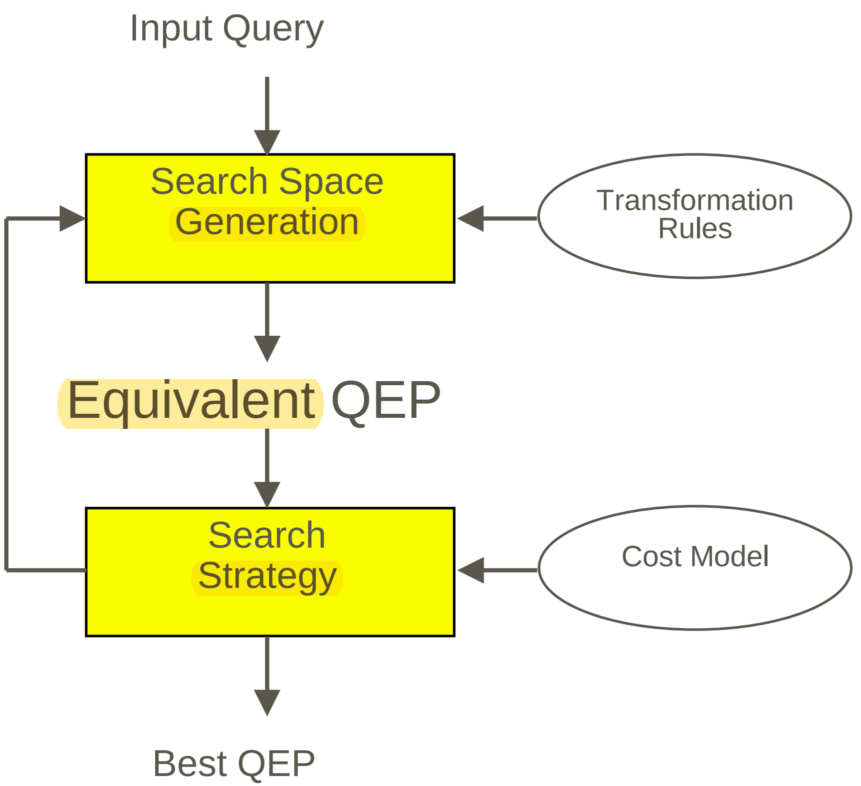
Input: Query on fragments

- Find the best (*not necessarily optimal*) global execution schedule/query plan
  - Minimize a cost function
  - Distributed join processing
    - Bushy vs. linear trees
    - Which relation to ship where?
  - Decide on the use of semijoins
  - …

+ Cost-Based Optimization

- Solution space
  - The set of equivalent algebra expressions (query trees)
- Cost function (in terms of time)
  - I/O cost + CPU cost + communication cost
  - These might have different weights in different distributed environments (LAN vs WAN).
  - Can also maximize system throughput
- Search algorithm
  - How do we move inside the solution space?
  - Exhaustive search, heuristic algorithms (iterative improvement, simulated annealing, genetic,…)

Query Optimization Process



+ Search Space

- Search space characterized by alternative execution plans
- Focus on join trees
- For *N* relations, there are O(*N*!) equivalent left-deep join trees that can be obtained by applying commutativity and associativity rules

```
SELECT  ENAME,RESP,PNAME
FROM    EMP,ASG,PROJ
WHERE   EMP.ENO=ASG.ENO
AND     ASG.PNO=PROJ.PNO
```

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

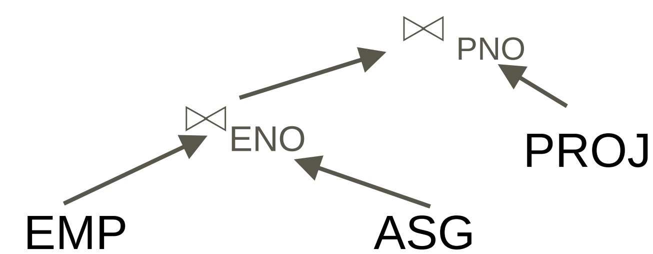
ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

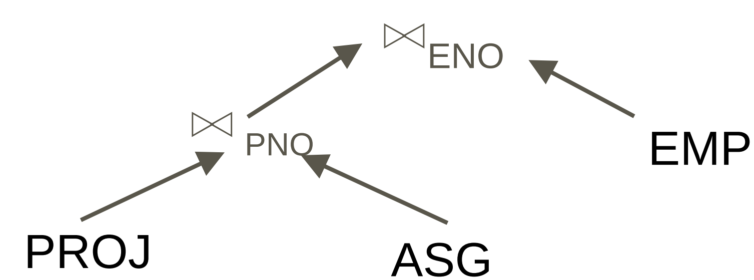
PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

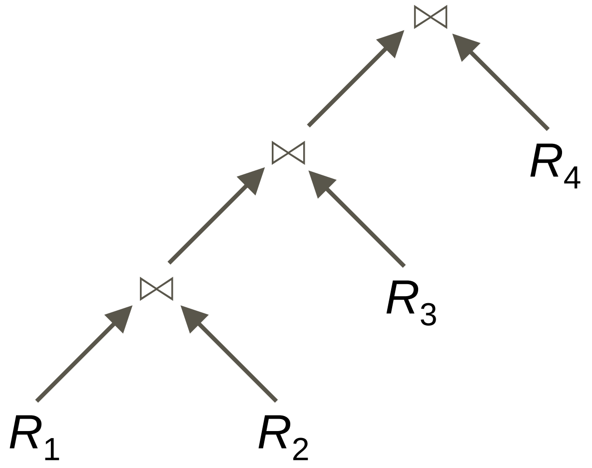




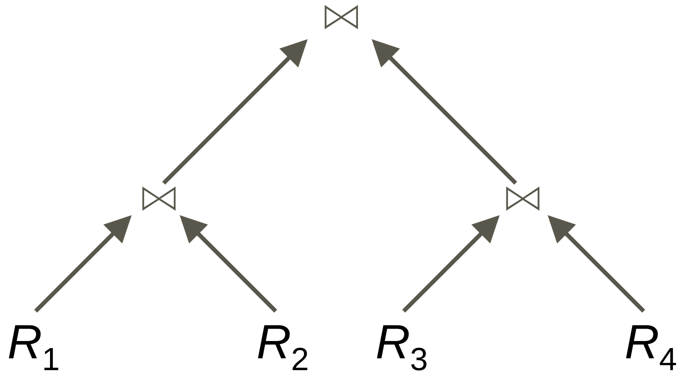
+ Limiting Search Space

- Restrict by means of heuristics
  - Perform unary operations before binary operations
  - …
- Restrict the shape of the join tree
  - Consider only linear trees, ignore bushy ones

Linear Join Tree



Bushy Join Tree



+ Cost Functions

- Total Time (or Total Cost)
  - Reduce each cost (in terms of time) component individually
  - Do as little of each cost component as possible

- Response Time
  - Do as many things as possible in parallel
  - *May increase total time because of increased total activity*

+ Total Cost

Summary of all cost factors

Total Cost = CPU cost + I/O cost + communication cost

- **CPU Cost** = unit instruction cost ∗ **no. of data items**
- **I/O Cost** = unit disk I/O cost ∗ no. of disk I/Os
- **Communication Cost** = message initiation + transmission

So what can we do?

More details on Page 210, Chapter 6, Ozsu & Valduriez

+ Total Cost Factors

- Wide Area Network
  - Message initiation and transmission costs are high
  - Local processing cost is low (fast mainframes or minicomputers)
  - Ratio of communication to I/O costs = 20:1
- Local Area Networks
  - Communication and local processing costs are more or less equal

+ Response Time

Elapsed time between the initiation and the completion (end-to-end) of a query (aka latency)

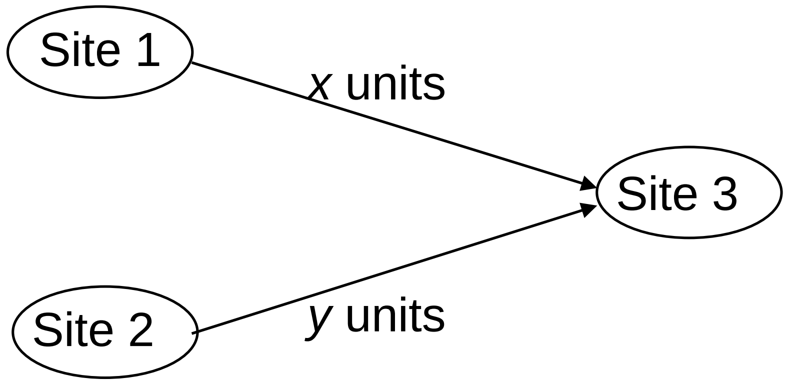
Response time = CPU time + I/O time + Communication time

- **CPU time** = unit instruction time * no. of sequential instructions
- **I/O time** = unit I/O time * no. of sequential I/Os
- **Communication time** = unit msg initiation time * no. of sequential msg + unit transmission time * no. of sequential bytes

So what can we do?

More details on Page 250, Chapter 8, Ozsu & Valduriez

+ Example



Assume that only the communication cost is considered

1. **Total time** = 2 ∗ message initialization time + unit transmission time * (*x*+*y*)

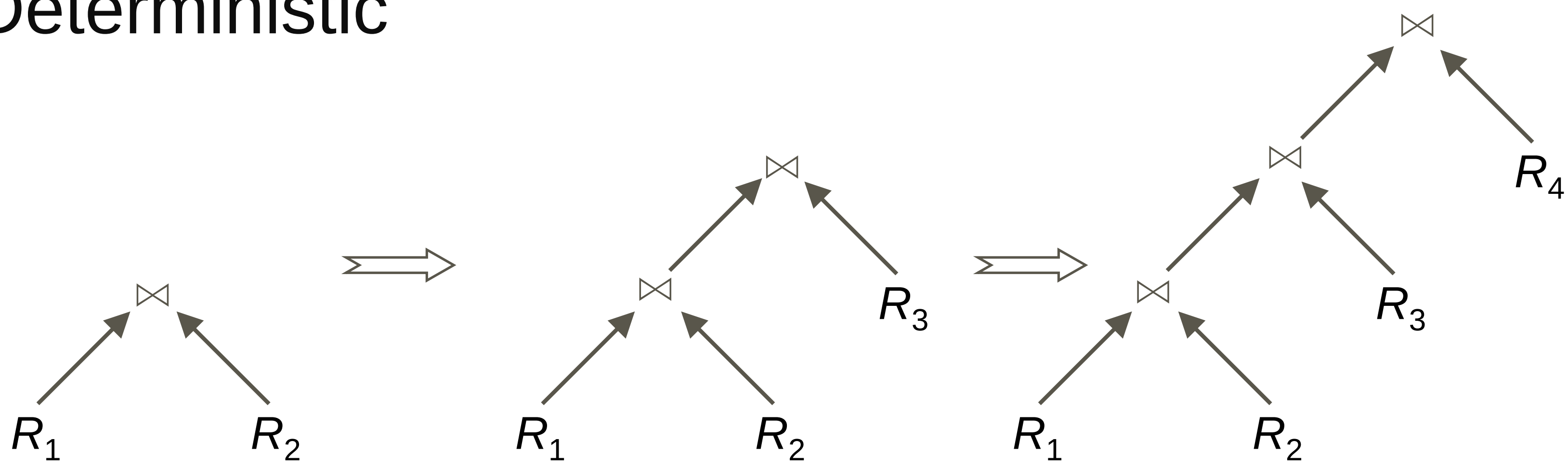
2. **Response time** = max {time to *send x from 1 to 3*, time to *send y from 2 to 3*}
   - time to send *x* from 1 to 3 = message initialization time + unit transmission time * *x*
   - time to send *y* from 2 to 3 = message initialization time + unit transmission time * *y*

+ Search Strategies

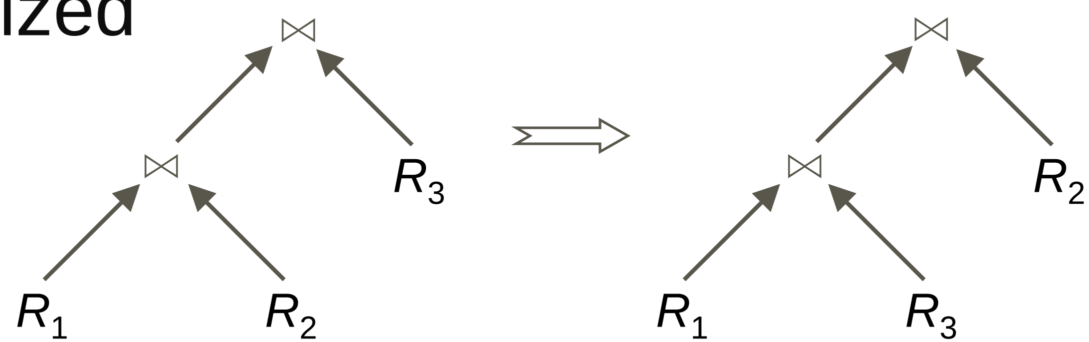
- How to “move” in the search space
- Deterministic
  - Start from base relations and build plans by adding one relation at each step
  - *Dynamic programming*
- Randomized
  - Search for optimality around a particular starting point
  - Trade optimization time for execution time
  - Better when > 10 relations
  - Simulated annealing, hill climbing, etc.
  - Iterative improvement

+ Search Strategies

- Deterministic

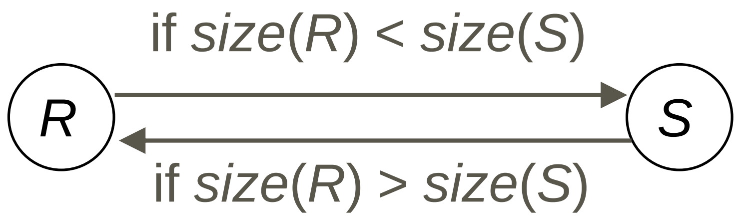


- Randomized



+ Join Ordering

- Consider two relations only



- Multiple relations more difficult because too many alternatives.
  - Compute the cost of all alternatives and select the best one
    - Necessary to compute the size of intermediate relations which is difficult.
  - Use heuristics

+ Join Ordering – Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

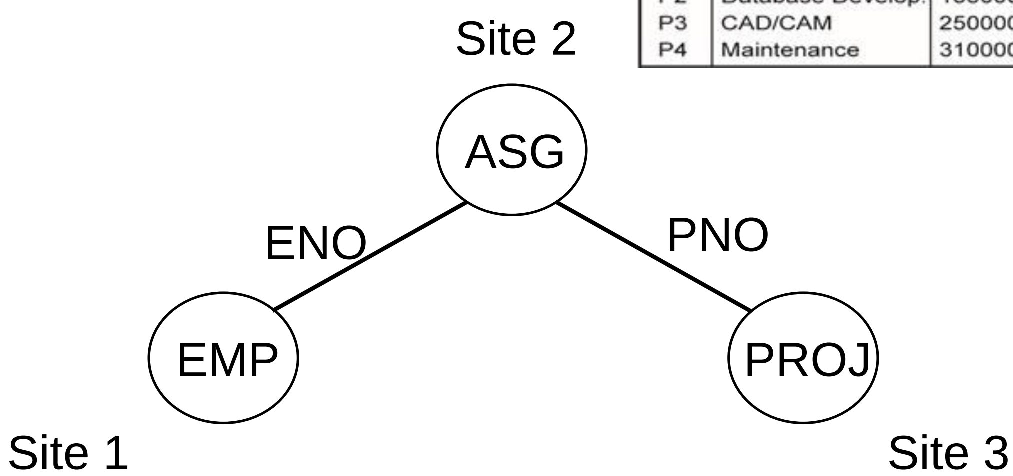
PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

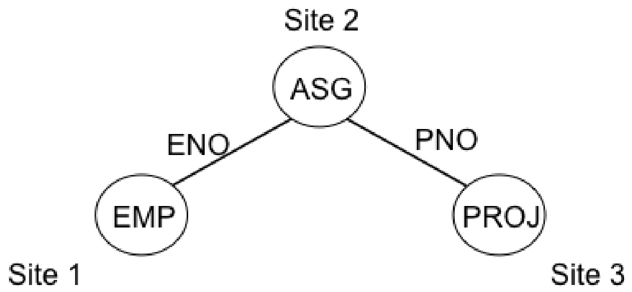
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Consider

$$PROJ \bowtie_{PNO} ASG \bowtie_{ENO} EMP$$



Join Ordering – Example



Execution alternatives:

1. EMP$\rightarrow$ Site 2

   Site 2 computes EMP'=EMP $\bowtie$ ASG
   EMP'$\rightarrow$ Site 3

   Site 3 computes EMP' $\bowtie$ PROJ

2. ASG $\rightarrow$ Site 1

   Site 1 computes EMP'=EMP $\bowtie$ ASG
   EMP' $\rightarrow$ Site 3

   Site 3 computes EMP' $\bowtie$ PROJ

3. ASG $\rightarrow$ Site 3

   Site 3 computes ASG'=ASG $\bowtie$ PROJ
   ASG' $\rightarrow$ Site 1

   Site 1 computes ASG' $\bowtie$ EMP

4. PROJ $\rightarrow$ Site 2

   Site 2 computes PROJ'=PROJ $\bowtie$ASG
   PROJ' $\rightarrow$ Site 1

   Site 1 computes PROJ' $\bowtie$ EMP

5. EMP $\rightarrow$ Site 2
   PROJ $\rightarrow$ Site 2

   Site 2 computes EMP $\bowtie$ PROJ $\bowtie$ ASG

Why do we need to do join operations?

+ Semi-join Algorithm

- Consider the join of two relations (to be joined on A):
  - R[A] (located at site 1)
  - S[A] (located at site 2)
- With two alternatives:
  - Do the join $R \bowtie_A S$ on one site, or
  - Perform a semi-join equivalent

Why do we need Semi-join?
What is it used for?

+ Semi-join Algorithm


```
SELECT ENAME
FROM   EMP,ASG
WHERE  EMP.ENO = ASG.ENO AND
             ASG.ENO <= "E3"
```

- Option 1: Perform the join
  - send R to Site 2
  - Site 2 computes $R \bowtie_A S$
- Option 2: Consider semi-join $(R \ltimes_A S) \bowtie_A S$
  - $S' = \Pi_A(S)$
  - $S' \rightarrow$ Site 1
  - **Site 1** computes $R' = R \ltimes_A S'$
  - $R' \rightarrow$ Site 2
  - **Site 2** computes $R' \bowtie_A S$

Semi-join is better (beneficial) if

$$size(\Pi_A(S)) + size(R \ltimes_A S')) < size(R)$$

Site 2 (S)

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

Site 1 (R)

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

+ Recommended Readings

- Elmasri & Navathe, 6th edition
  - Chapter 25: Distributed Databases
- Elmasri & Navathe, 7th edition
  - Chapter 23: Distributed Database Concepts
- **Ozsu & Valduriez: *Principles of Distributed Database Systems,* 3rd edition, Springer**
  - Chapter 6: Overview of Query Processing
  - Chapter 7: Query Decomposition and Data Localization
  - Chapter 8: Optimization of Distributed Queries

Textbook (PDF): https://link.springer.com/book/10.1007/978-1-4419-8834-8

- Next week: Distributed Transaction Management

https://seu1.org/files/level6/IT344/Fundamentals_of_Database_Systems,_6th_Edition.pdf

https://www.auhd.site/upfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf