Outline

- Introduction
- Background
- Distributed Database Design
- Database Integration
- Semantic Data Control
- Distributed Query Processing
  - → Overview
  - → Query decomposition and localization
  - → Distributed query optimization
- Multidatabase query processing
- Distributed Transaction Management
- Data Replication
- Parallel Database Systems
- Distributed Object DBMS
- Peer-to-Peer Data Management
- Web Data Management
- Current Issues

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

$$(p_{11} \lor p_{12} \lor \ldots \lor p_{1n}) \land \ldots \land (p_{m1} \lor p_{m2} \lor \ldots \lor p_{mn})$$

  - Disjunctive normal form

$$(p_{11} \land p_{12} \land \ldots \land p_{1n}) \lor \ldots \lor (p_{m1} \land p_{m2} \land \ldots \land p_{mn})$$

  - OR's mapped into union
  - AND's mapped into join or selection

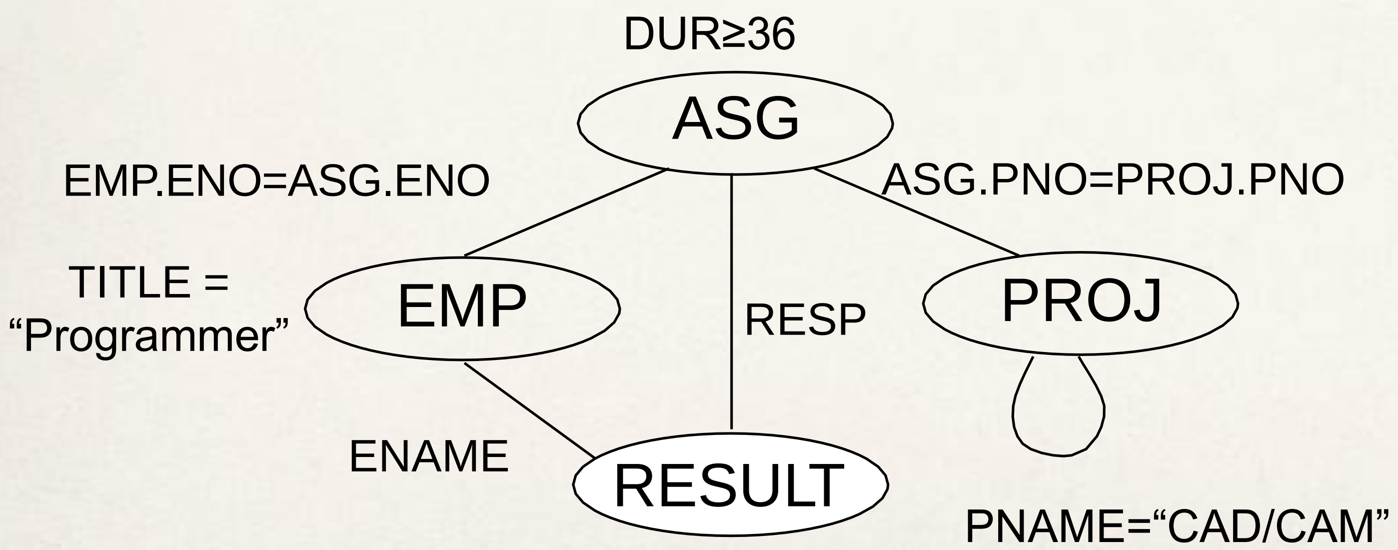
Analysis

- 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

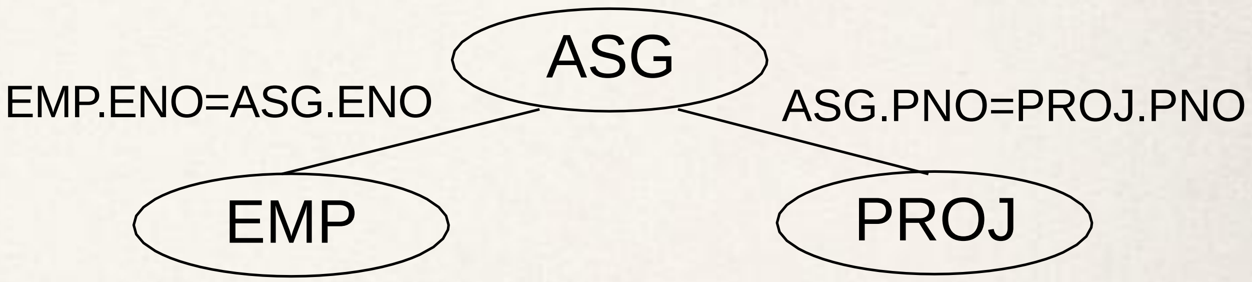
Analysis – Example

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

Query graph



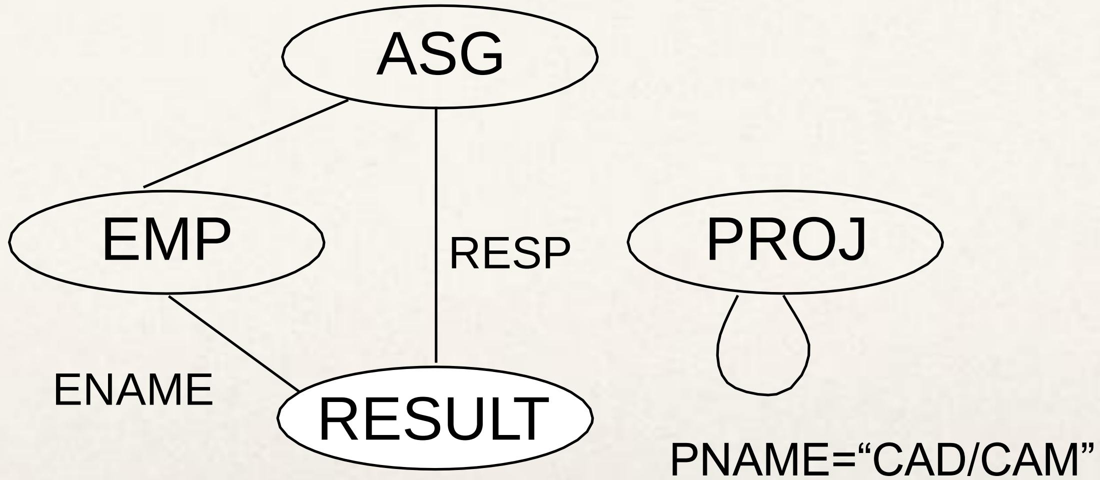
Join graph



Analysis

If the query graph is not connected, the query may be wrong or use Cartesian product

```
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?
  - Remember the example
- How? Use transformation rules
  - Elimination of redundancy
    - idempotency rules

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

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

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

…

  - Application of transitivity
  - Use of integrity rules

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."))
```

⬇

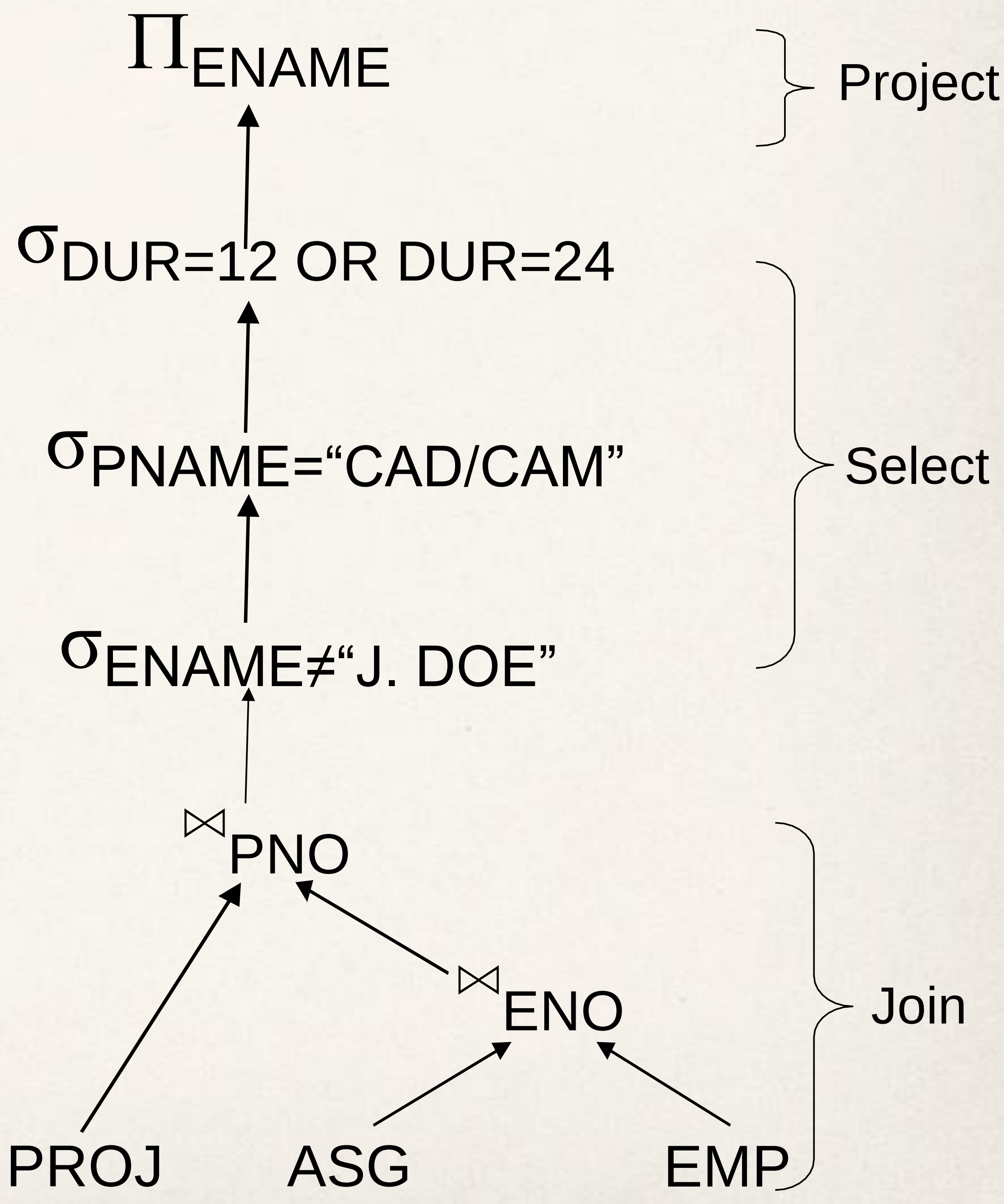
```
SELECT   TITLE
FROM     EMP
WHERE    EMP.ENAME = "J. Doe"
```

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

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

  where A_i belongs to R and T
- Commuting projection with binary operations
  - $\Pi_C(R \times S) \Leftrightarrow \Pi_{A'}(R) \times \Pi_{B'}(S)$
  - $\Pi_C(R \bowtie_{p(A_j,B_k)} S) \Leftrightarrow \Pi_{A'}(R) \bowtie_{p(A_j,B_k)} \Pi_{B'}(S)$
  - $\Pi_C(R \cup S) \Leftrightarrow \Pi_C(R) \cup \Pi_C(S)$

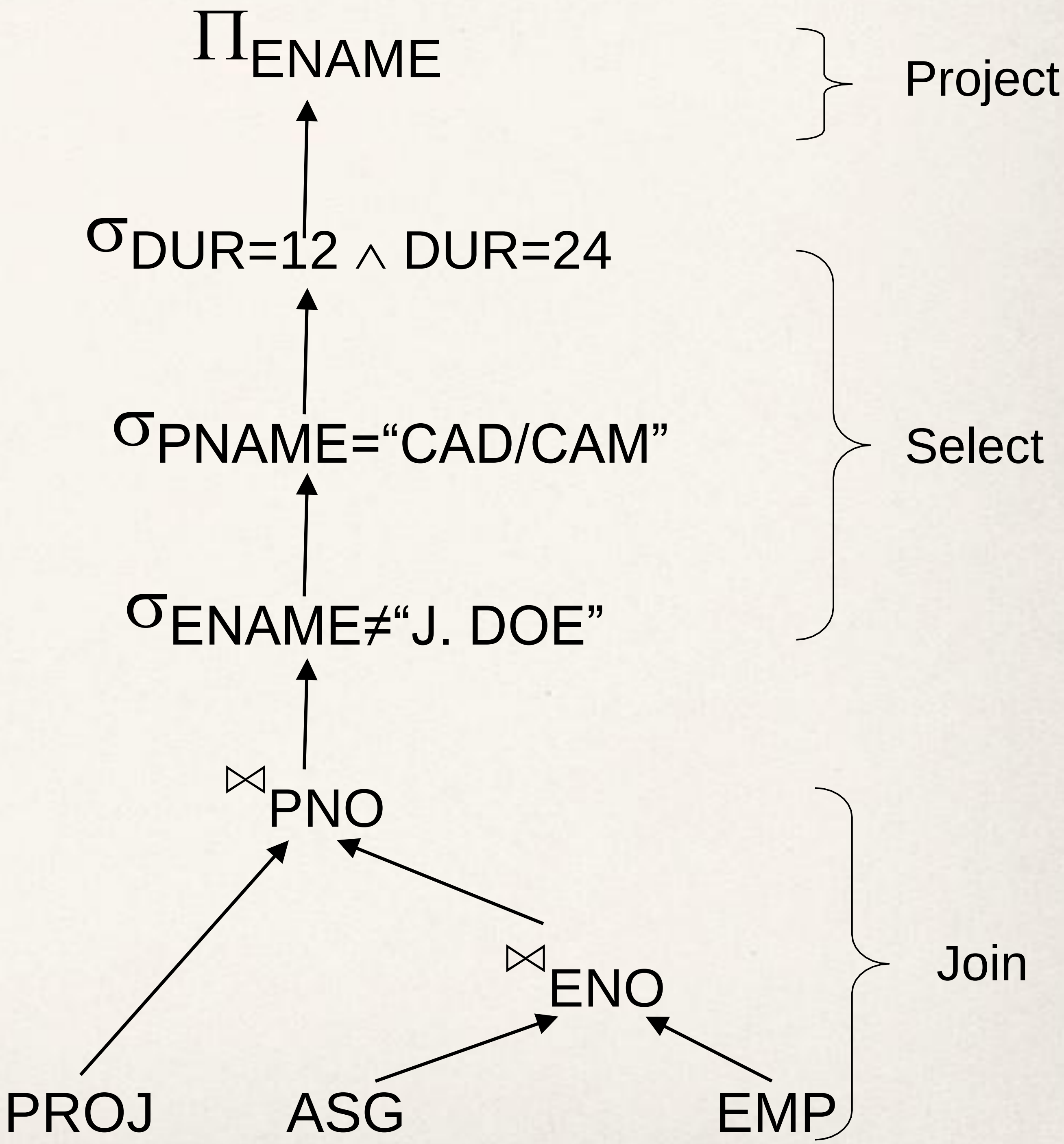
  where $R[A]$ and $S[B]$; $C = A' \cup B'$ where $A' \subseteq A$, $B' \subseteq B$

Example

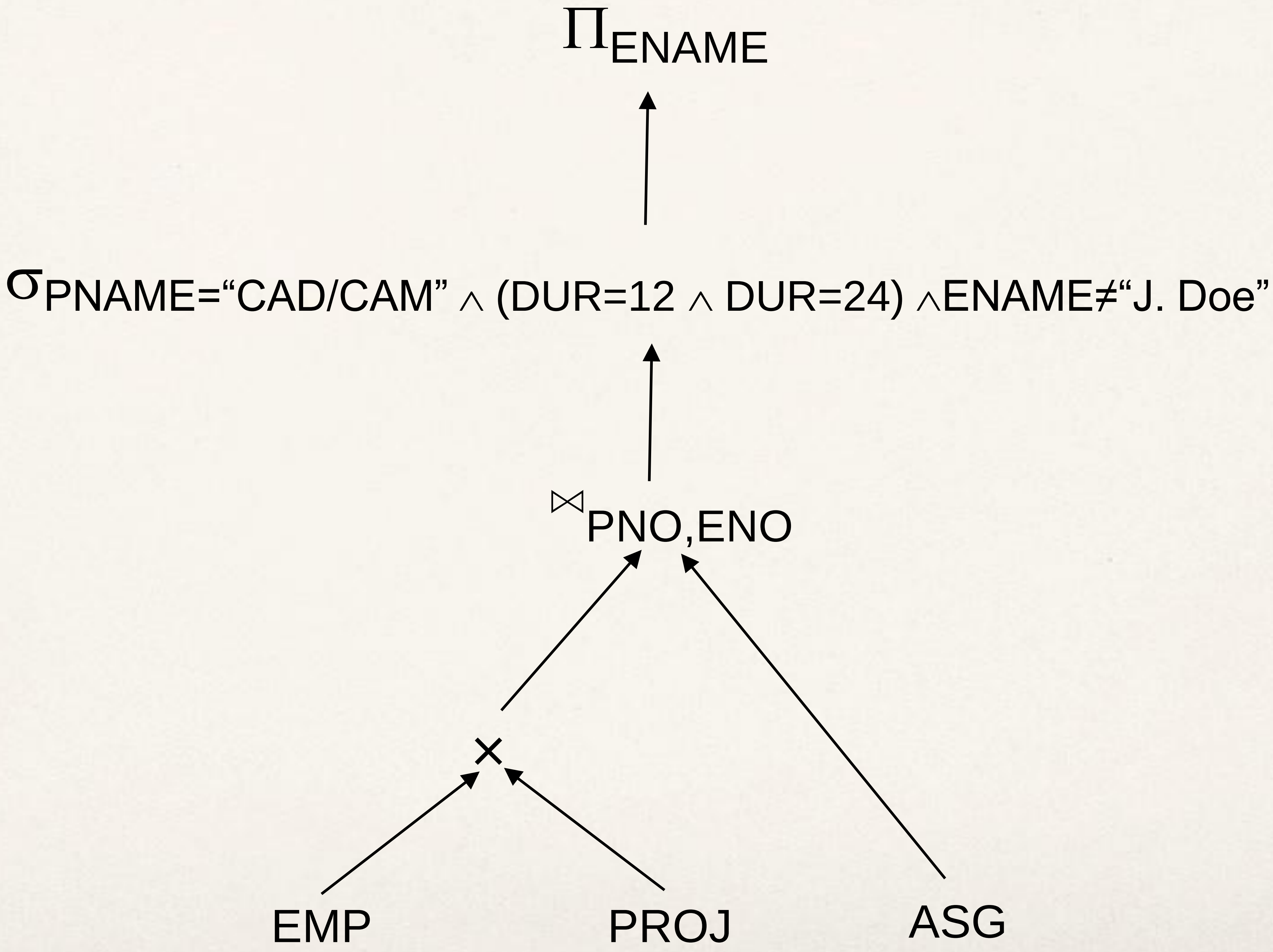
Recall the previous example:

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

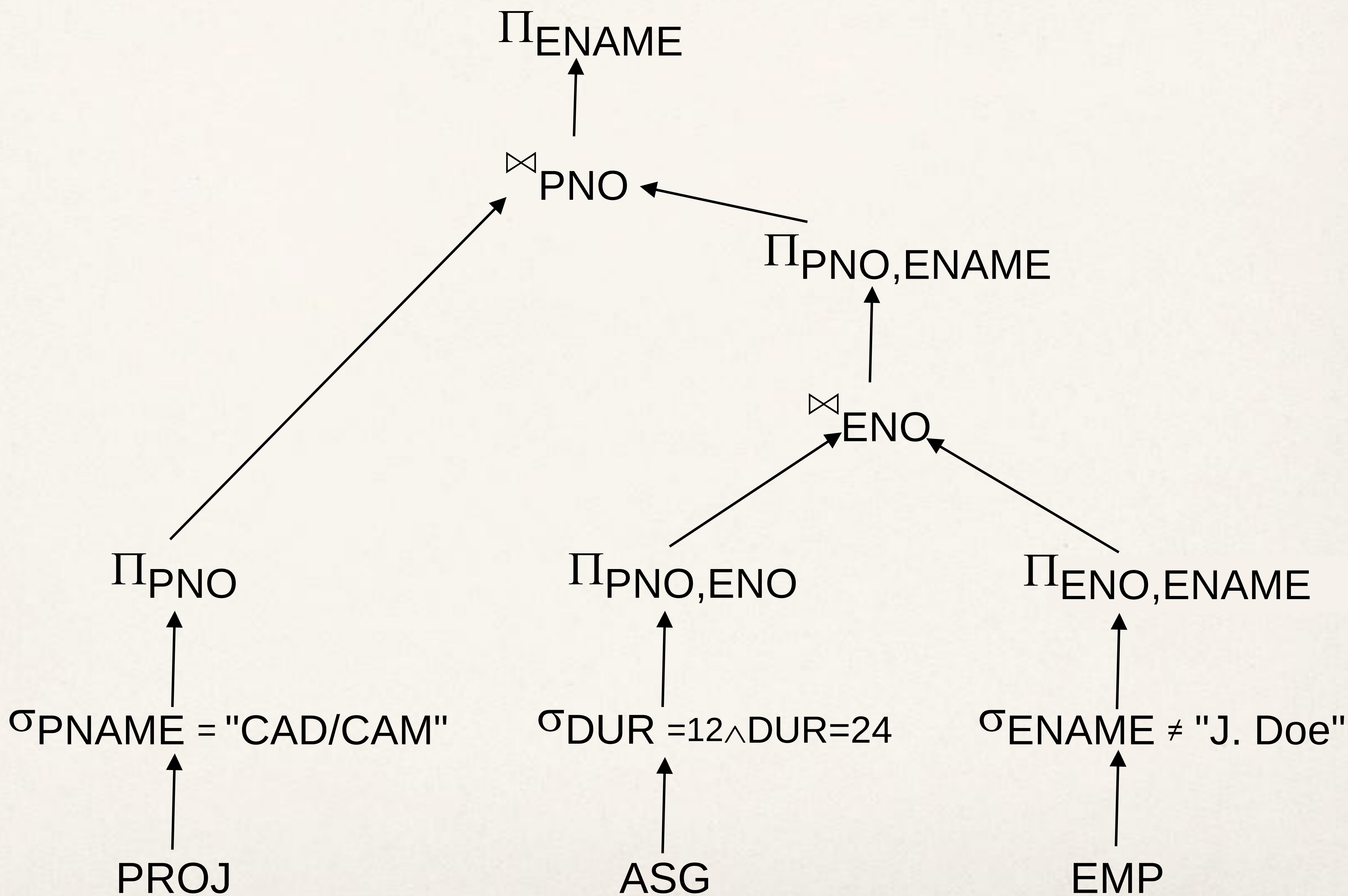
```
SELECT ENAME
FROM   PROJ, ASG, EMP
WHERE  ASG.ENO=EMP.ENO
AND    ASG.PNO=PROJ.PNO
AND    ENAME ≠ "J. Doe"
AND    PROJ.PNAME="CAD/CAM"
AND    (DUR=12 OR DUR=24)
```



Equivalent Query



Restructuring



Step 2 – Data Localization

Input: Algebraic query on distributed relations

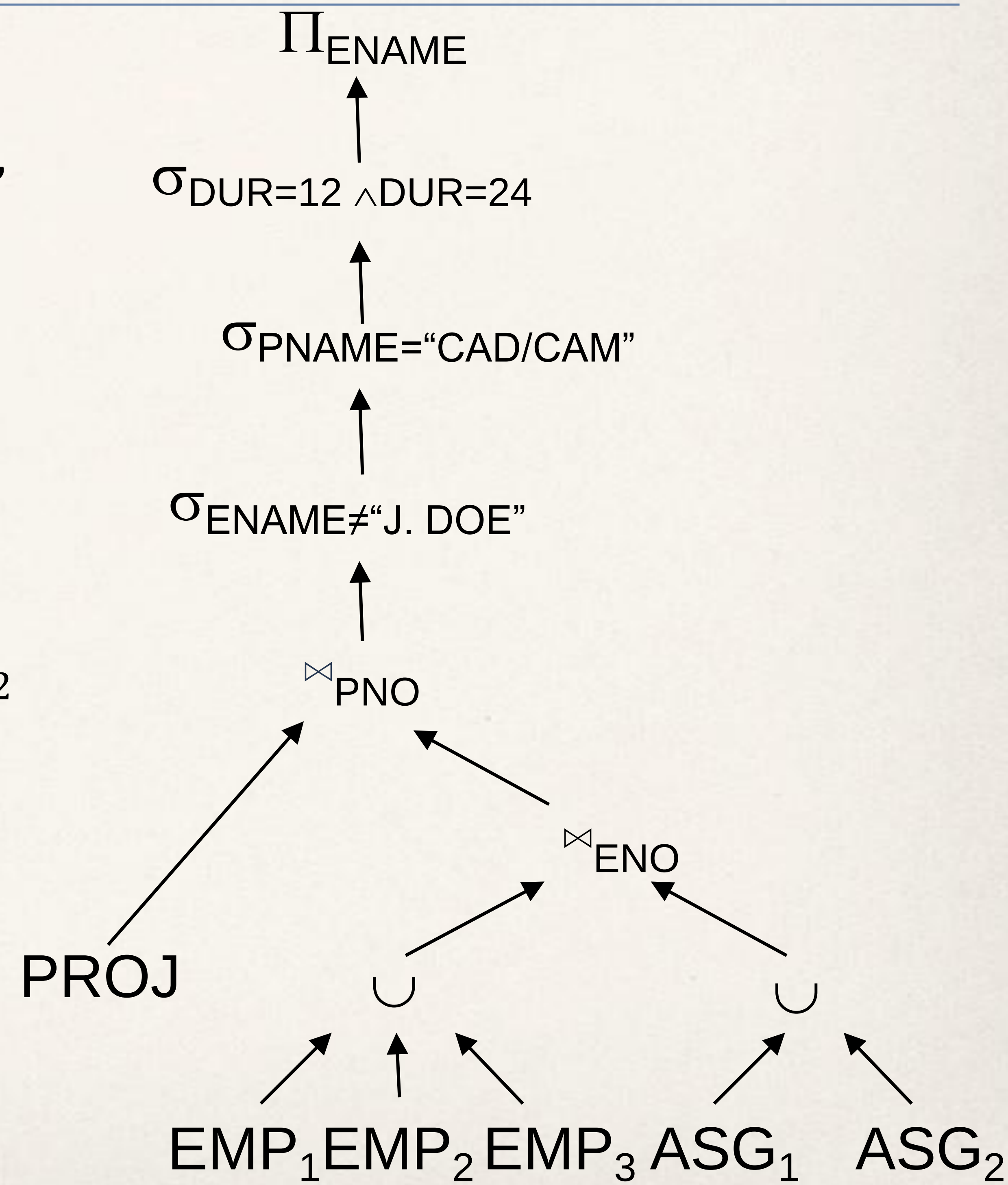
- Determine which fragments are involved
- Localization program
  - → substitute for each global query its materialization program
  - → optimize

Example

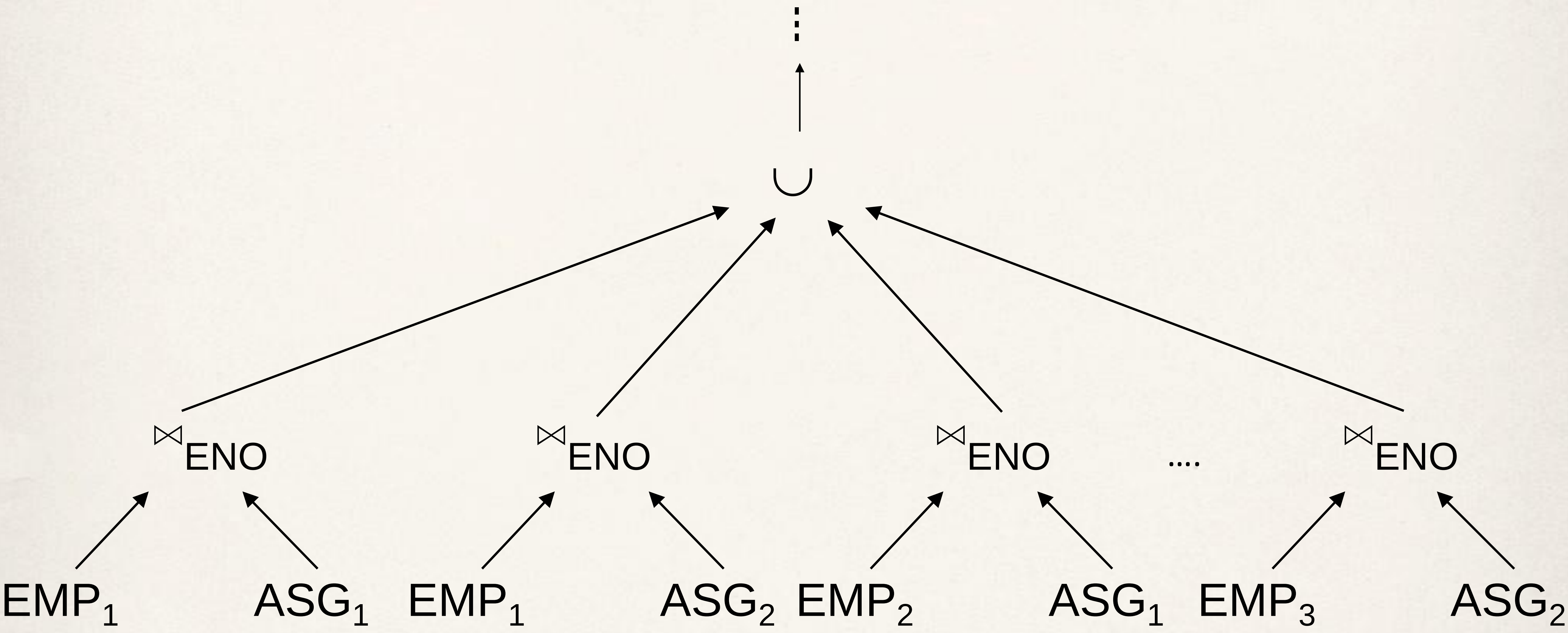
Assume

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

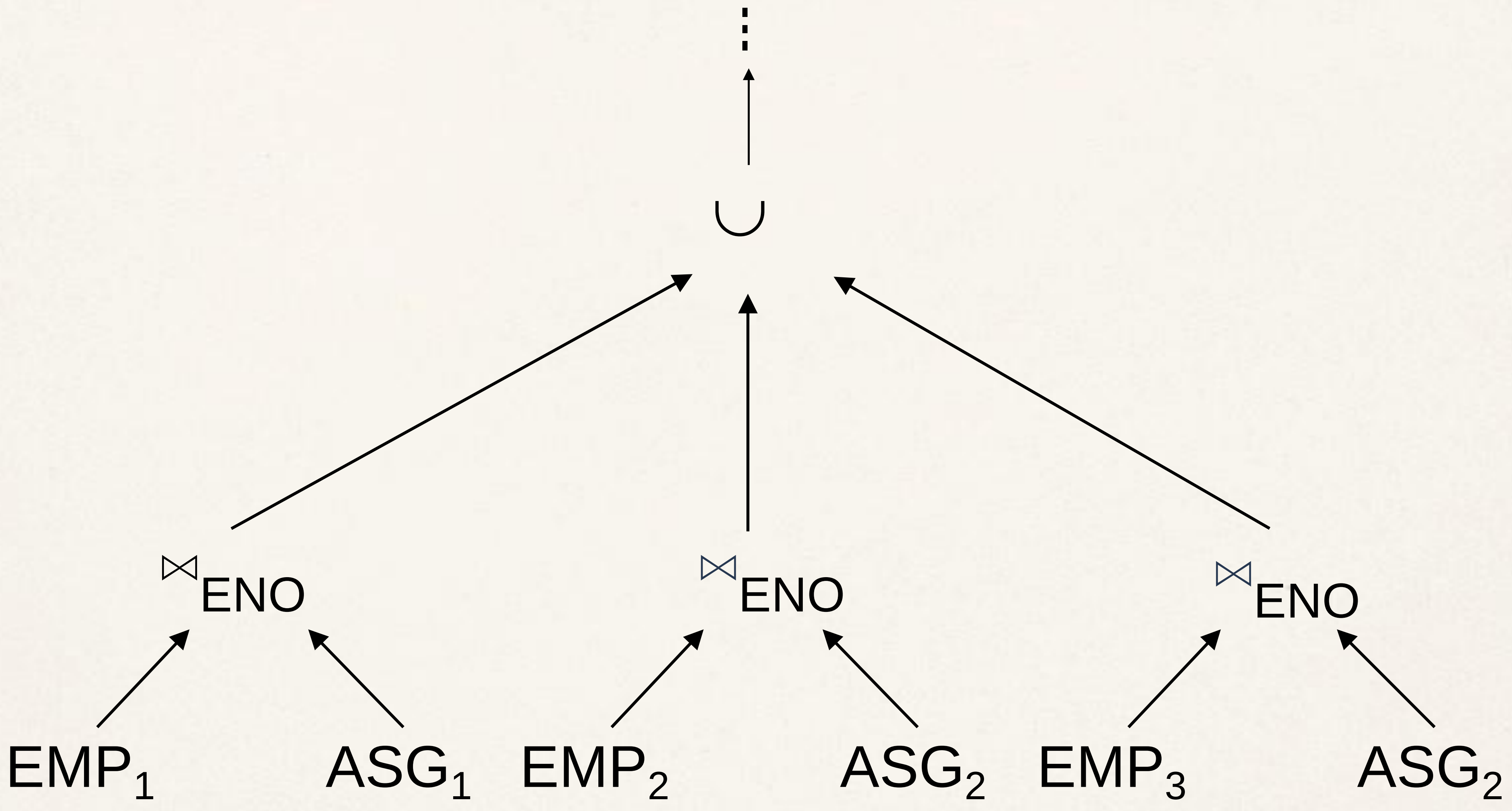
Replace EMP by $(EMP_1 \cup EMP_2 \cup EMP_3)$ and ASG by $(ASG_1 \cup ASG_2)$ in any query



Provides Parallellism



Eliminates Unnecessary Work

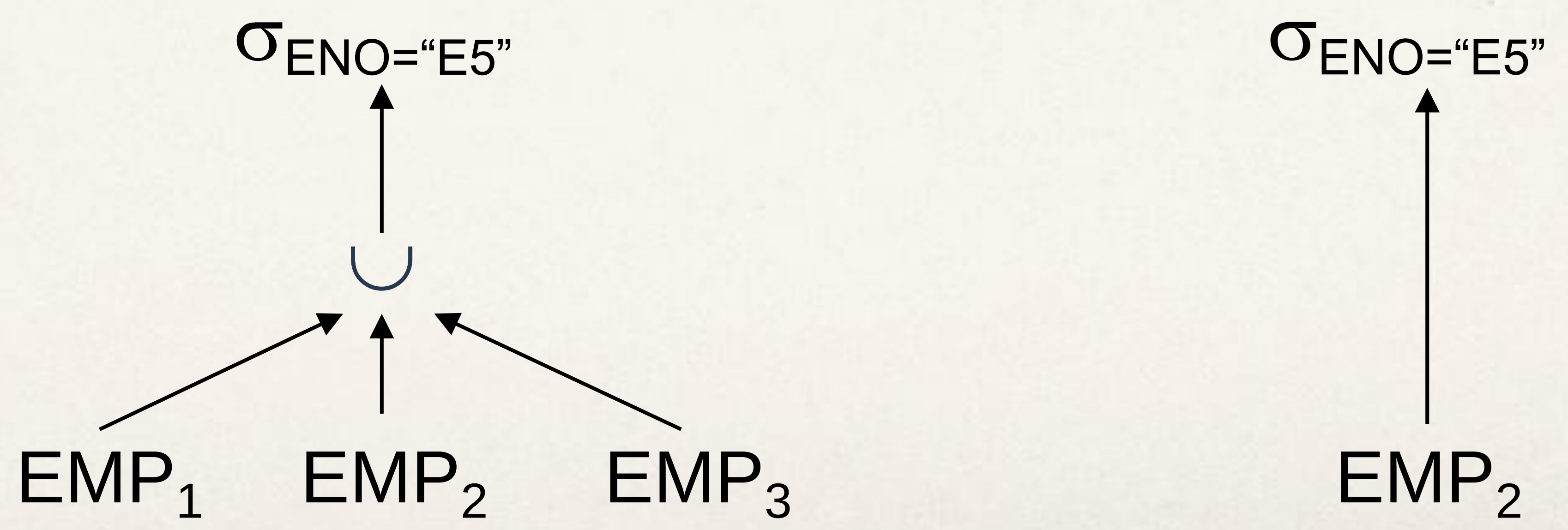


Reduction for PHF

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

    $\sigma_{p_i}(R_j)=\varnothing$ 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 $R_i = \sigma_{p_i}(R)$, $R_j = \sigma_{p_j}(R)$ and $S_j = \sigma_{p_j}(S)$

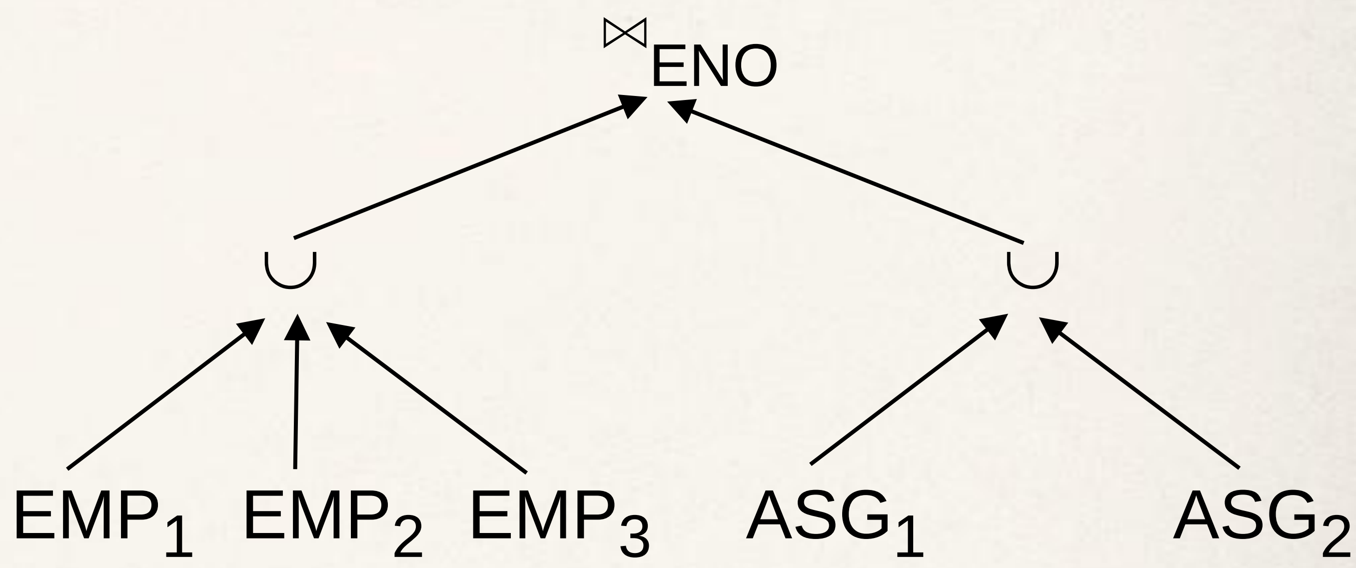
$$R_i \bowtie S_j = \varnothing \text{ if } \forall x \text{ in } R_i, \forall y \text{ in } R_j \text{: } \neg(p_i(x) \wedge p_j(y))$$

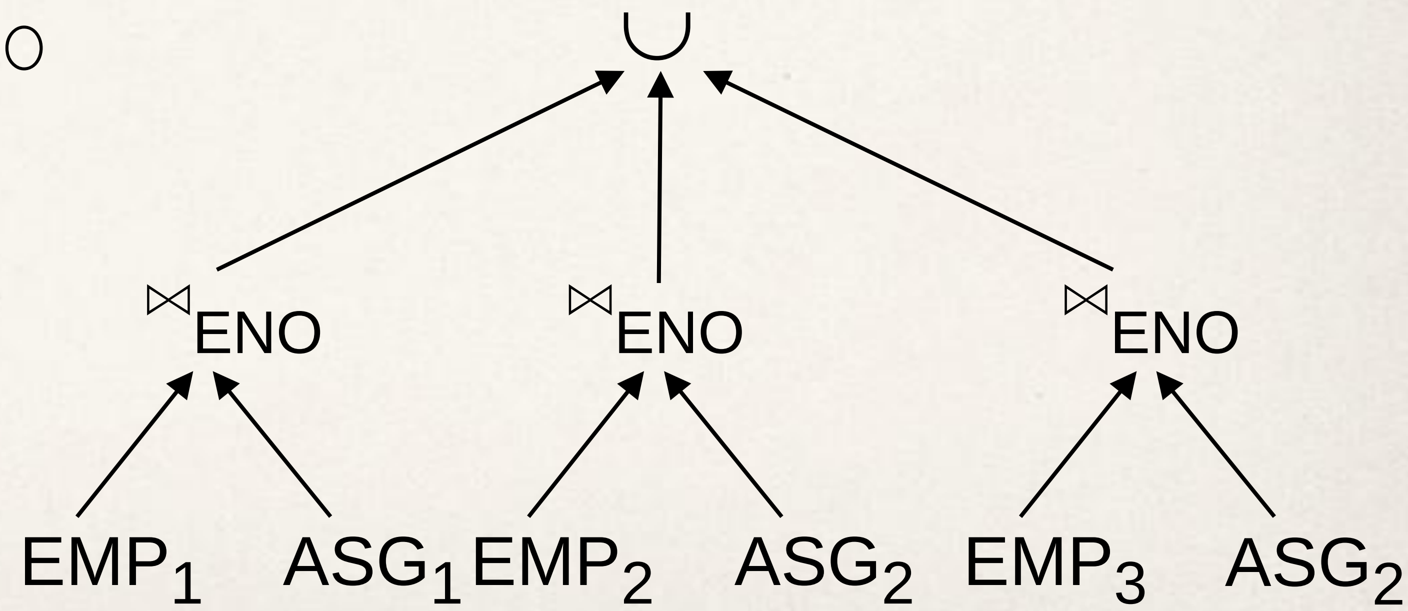
Reduction for PHF

- 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
```

- Distribute join over unions
- Apply the reduction rule





Reduction for VF

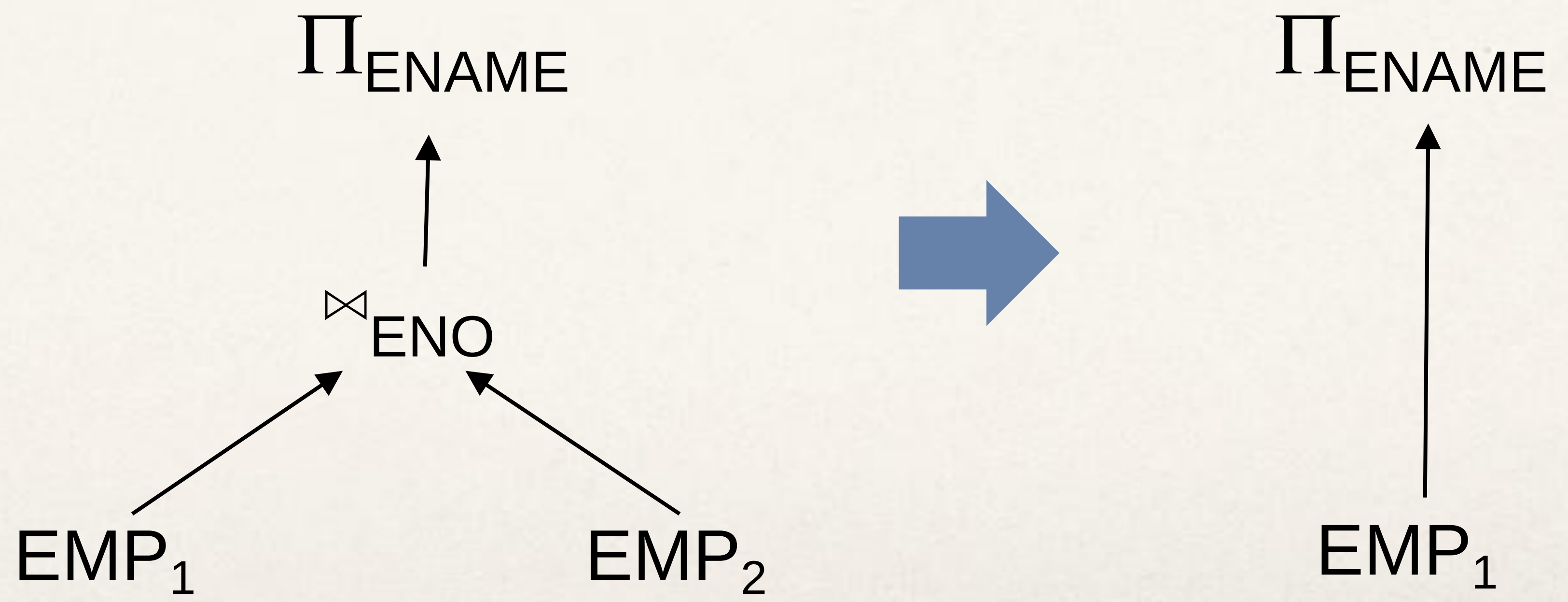
- Find useless (not empty) intermediate relations

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)

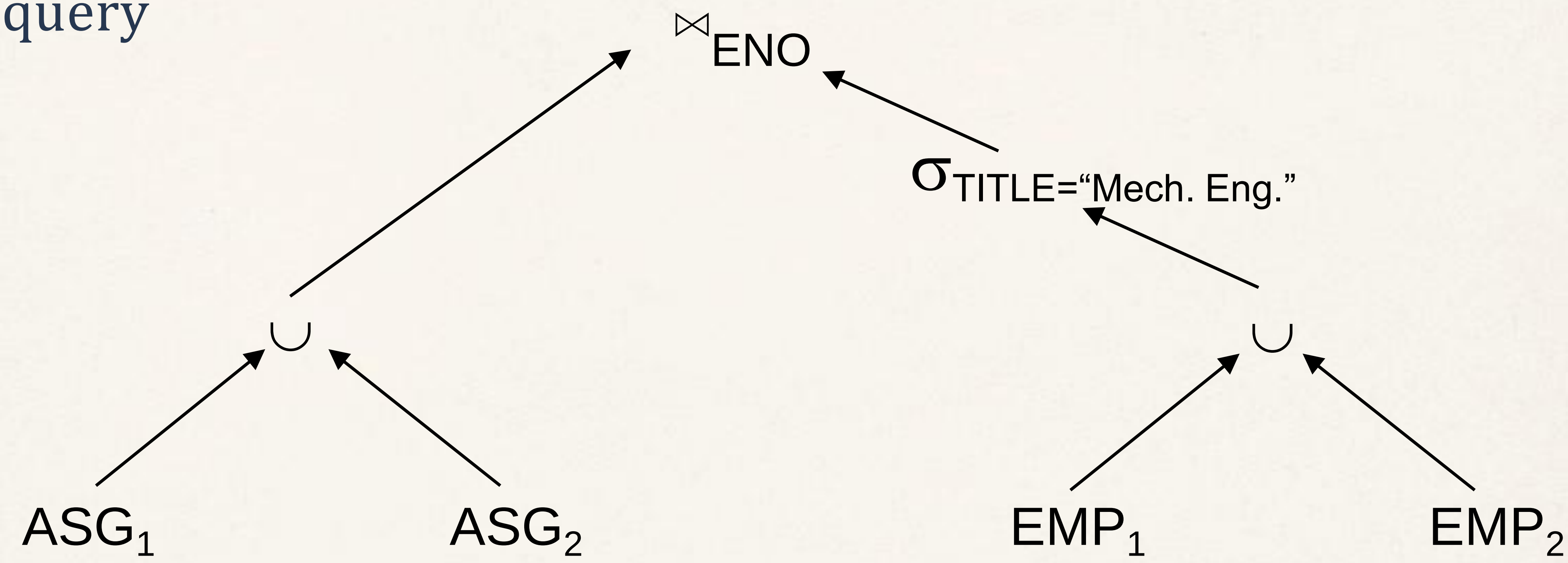
EMP_2: $\sigma_{TITLE\neq\text{“Programmer”}}$ (EMP)

- Query

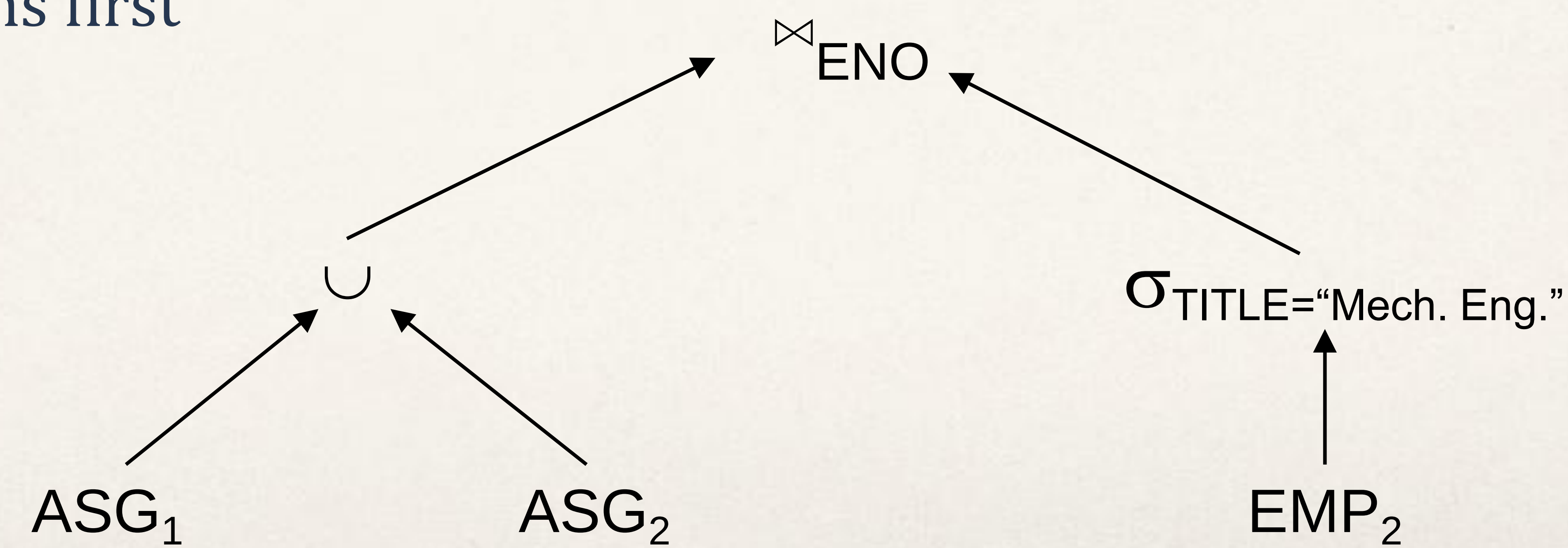
```
SELECT    *
FROM      EMP, ASG
WHERE     ASG.ENO  =   EMP.ENO
AND       EMP.TITLE = "Mech. Eng."
```

Reduction for DHF

Generic query

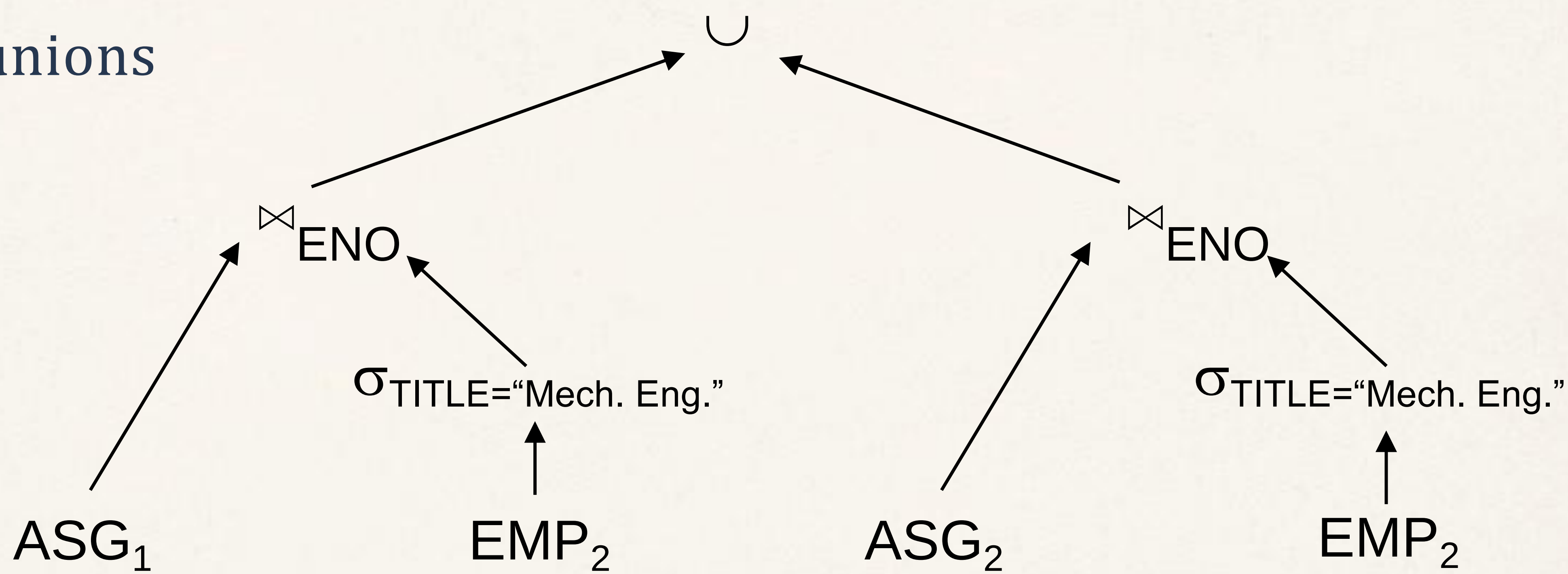


Selections first



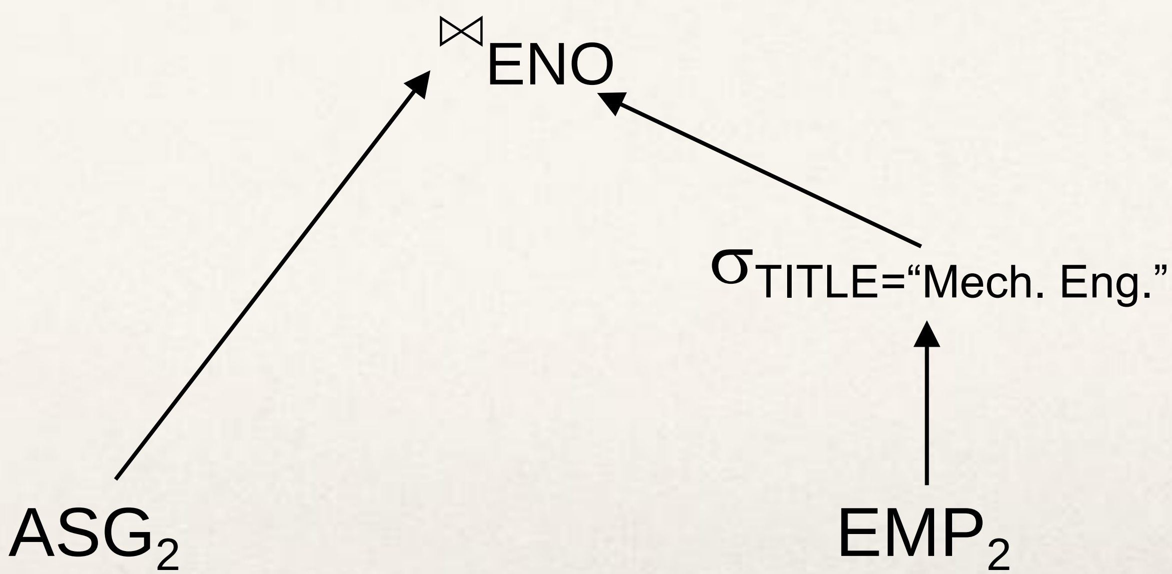
Reduction for DHF

Joins over unions



Elimination of the empty intermediate relations

(left sub-tree)



Reduction for Hybrid Fragmentation

- 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:

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

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

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

and the query

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