Distributed Database Systems

Query Decomposition and Data Localization

605.741

David Silberberg

Query Decomposition

- This is the same for central and distributed DBMSs
- Normalization
  - We are manipulating (normalizing) the constraints of the WHERE clause
  - Relational languages like SQL are quantifier-free
  - Conjunctive normal form

    $$(p1 \vee p2 \vee p3) \wedge (p4 \vee p5) \wedge (p6)$$

  - Disjunctive normal form

    $$(p1 \wedge p2 \wedge p3) \vee (p4 \wedge p5) \vee (p6)$$

    - Requires many unions, and thus, replicated selections
    - Conjunction is more efficient, because that is the way queries are generally written

Some Rules

- Commutative law
  - $a \wedge b \Leftrightarrow b \wedge a$
  - $a \vee b \Leftrightarrow b \vee a$
- Associative law
  - $(a \wedge b) \wedge c \Leftrightarrow a \wedge (b \wedge c)$
  - $(a \vee b) \vee c \Leftrightarrow a \vee (b \vee c)$
- Distributive law
  - $a \wedge (b \vee c) \Leftrightarrow (a \wedge b) \vee (a \wedge c)$
  - $a \vee (b \wedge c) \Leftrightarrow (a \vee b) \wedge (a \vee c)$
- Negation
  - $\neg(\neg a) \Leftrightarrow a$
- DeMorgan's law
  - $\neg(a \wedge b) \Leftrightarrow \neg a \vee \neg b$
  - $\neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b$

Example Decomposition

```
SELECT  ENAME
FROM    EMP, ASG
WHERE   EMP.ENO = ASG.ENO          AND
        ((RESP = 'Manager'         AND
          DUR = 12)                OR
         NOT (RESP = 'Programmer'  OR
                DUR <> 12))
```

goes to

```
SELECT  ENAME
FROM    EMP, ASG
WHERE   EMP.ENO = ASG.ENO          AND
        DUR = 12                   AND
        (RESP = 'Manager'          OR
         RESP <> 'Programmer')
```

Decomposition Analysis

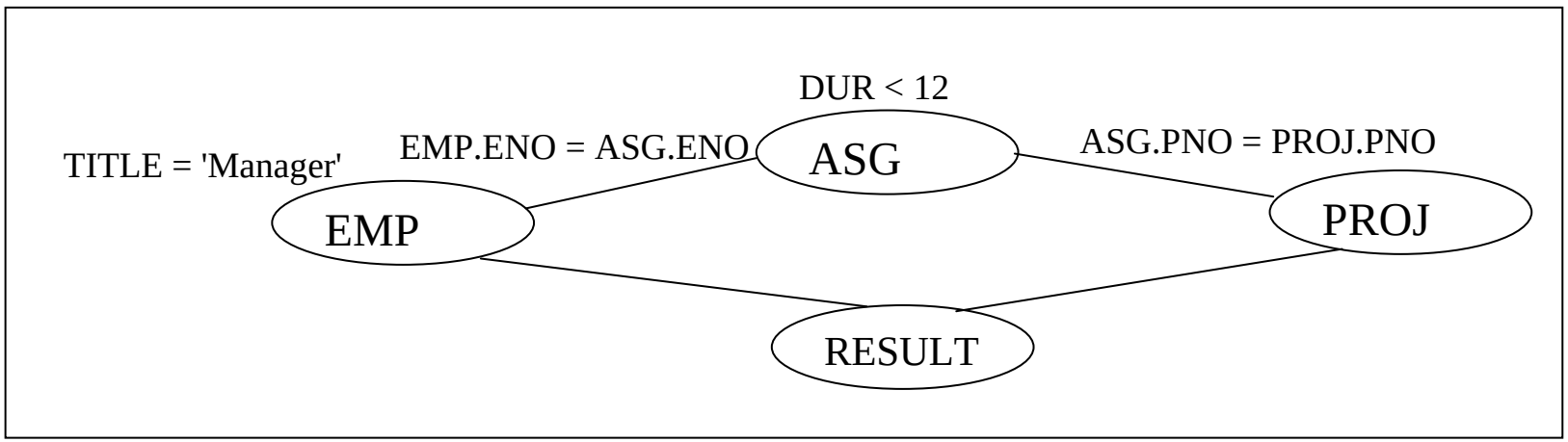
- Reject incorrect types & semantically incorrect queries
- Type incorrectness

```
SELECT    PNAME
FROM      PNO
WHERE     DUR = 'CAD/CAM'
```

- Semantic incorrectness
  - Reject queries that "don't make sense"
  - Components do not contribute to the results
  - This can be done in general for conjunctive queries
    - No negation or disjunction

Ullman's Graph Scheme

```
SELECT ENAME, PNAME
FROM   EMP, ASG, PROJ
WHERE  EMP.ENO = ASG.ENO      AND
       ASG.PNO = PROJ.PNO     AND
       DUR < 12               AND
       TITLE    = 'Manager'
```



Ullman's Scheme - Choices

- Semantically incorrect if graph is not connected
- Several choices
  - Reject
  - Cross-join (Cartesian product)
  - Infer missing join

Redundancy Elimination

- General laws for reduction - idempotency rules

  $p \wedge p \Leftrightarrow p$

  $p \vee p \Leftrightarrow p$

  $p \wedge true \Leftrightarrow p$

  $p \vee true \Leftrightarrow true$

  $p \wedge false \Leftrightarrow false$

  $p \vee false \Leftrightarrow p$

  $p \wedge \neg p \Leftrightarrow false$

  $p \vee \neg p \Leftrightarrow true$

  $p \wedge (p \vee q) \Leftrightarrow p$

  $p \vee (p \wedge q) \Leftrightarrow p$

Redundancy Elimination Example

```
SELECT    ENAME
FROM      EMP, ASG
WHERE     EMP.ENO = ASG.ENO      AND
          ((TITLE = 'Manager'    OR
            DUR = 12)            AND
           NOT (TITLE <> 'Manager' AND
                       DUR = 12)) AND
          (TITLE = 'Manager'     OR
           ENAME = 'Smith')
```

$((p \vee q) \wedge \neg(\neg p \wedge q)) \wedge (p \vee r)$
$((p \vee q) \wedge (p \vee \neg q)) \wedge (p \vee r)$
$(p \vee (q \wedge \neg q)) \wedge (p \vee r)$
$(p \vee false) \wedge (p \vee r)$
$p \wedge (p \vee r)$
p

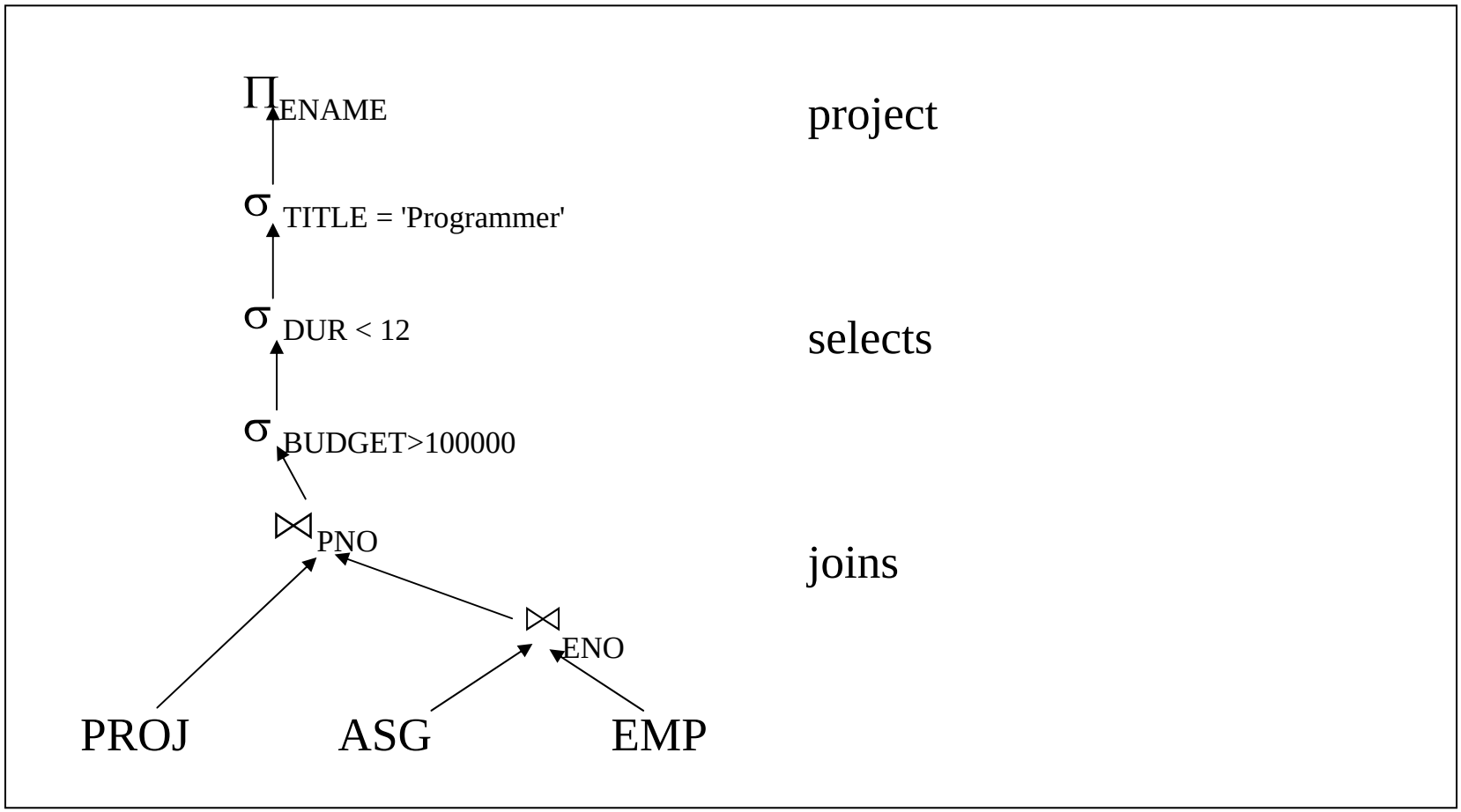
```
SELECT    ENAME
FROM      EMP, ASG
WHERE     EMP.ENO = ASG.ENO      AND
          TITLE = 'Manager'
```

Rewriting Queries

- Translate the query to relational algebra
- Create a query tree
- Use laws of the algebra (transformation rule) to push operations down the tree to effect efficiency
- Example

```
SELECT ENAME
FROM   EMP, ASG, PROJ
WHERE  EMP.ENO = ASG.ENO      AND
       ASG.PNO = PROJ.PNO     AND
       TITLE = 'Programmer'   AND
       DUR < 12               AND
       BUDGET > 100000
```

Create Tree in Order of Project, Select, and Join



Tree Transformation Rules

- Assume R = T = {A_1, A_2, A_3, …, A_n} ; S = {B_1, B_2, B_3, …, B_n}
- Commutative law (binary operators)
  - R X S = S X R
  - R ⋈ S = S ⋈ R
- Associative law (binary operators)
  - (R X S) X T = R X (S X T)
  - (R ⋈ S) ⋈ T = R ⋈ (S ⋈ T)

Transformation Rules (cont.)

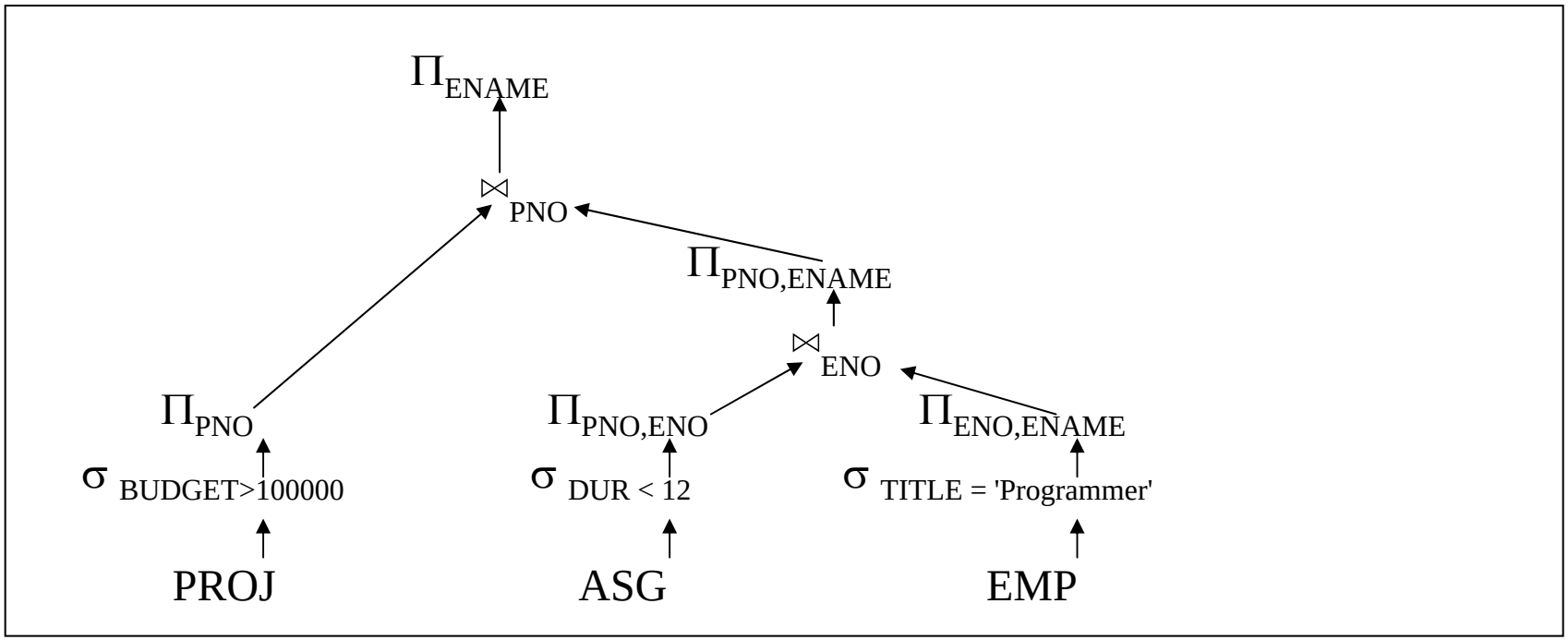
- Idempotency of unary operations
  - R is defined over attribute set A
  - $A' \subseteq A$, $A'' \subseteq A$, $A' \subseteq A''$
  - $\Pi_{A'} (\Pi_{A''} (R)) = \Pi_{A'} (R)$
  - $\sigma_{p1(A1)} (\sigma_{p2(A2)} (R)) = \sigma_{p1(A1) \wedge p2(A2)} (R)$
- Commuting selection with projection
  - $\Pi_{A1, A2, A3, \ldots, An} (\sigma_{p(Ap)} (R)) = \Pi_{A1, A2, A3, \ldots, An} (\sigma_{p(Ap)} \Pi_{A1, A2, A3, \ldots, An, Ap} (R))$
  - example
  - $\Pi_{A1, A2, A3, \ldots, An}$ is unnecessary if $A_p \in \{ A_1, A_2, A_3, \ldots, A_n \}$

Transformation Rules (cont.)

- Commuting selection with binary operators
  - $\sigma_{p(Ap)}(R \times S) = \sigma_{p(Ap)}(R) \times S$
  - $\sigma_{p(Ap)}(R \bowtie_{p(Aj,Bk)} S) = \sigma_{p(Ap)}(R) \bowtie_{p(Aj,Bk)} S$
  - $\sigma_{p(Ap)}(R \cup T) = \sigma_{p(Ap)}(R) \cup \sigma_{p(Ap)}(T)$
- Commuting projection with binary operators
  - $C = A' \cup B'$
  - $\Pi_C(R \times S) = \Pi_{A'}(R) \times \Pi_{B'}(S)$
  - $\Pi_C(R \bowtie_{p(Ai,Bj)} S) = \Pi_C(\Pi_{A',Ai}(R) \bowtie_{p(Ai,Bj)} \Pi_{B',Bj}(S))$
    - Note - Π_C is unnecessary if $A_i \in A'$ and $B_j \in B'$
  - $\Pi_C(R \cup S) = \Pi_C(R) \cup \Pi_C(S)$

Application to Query Tree

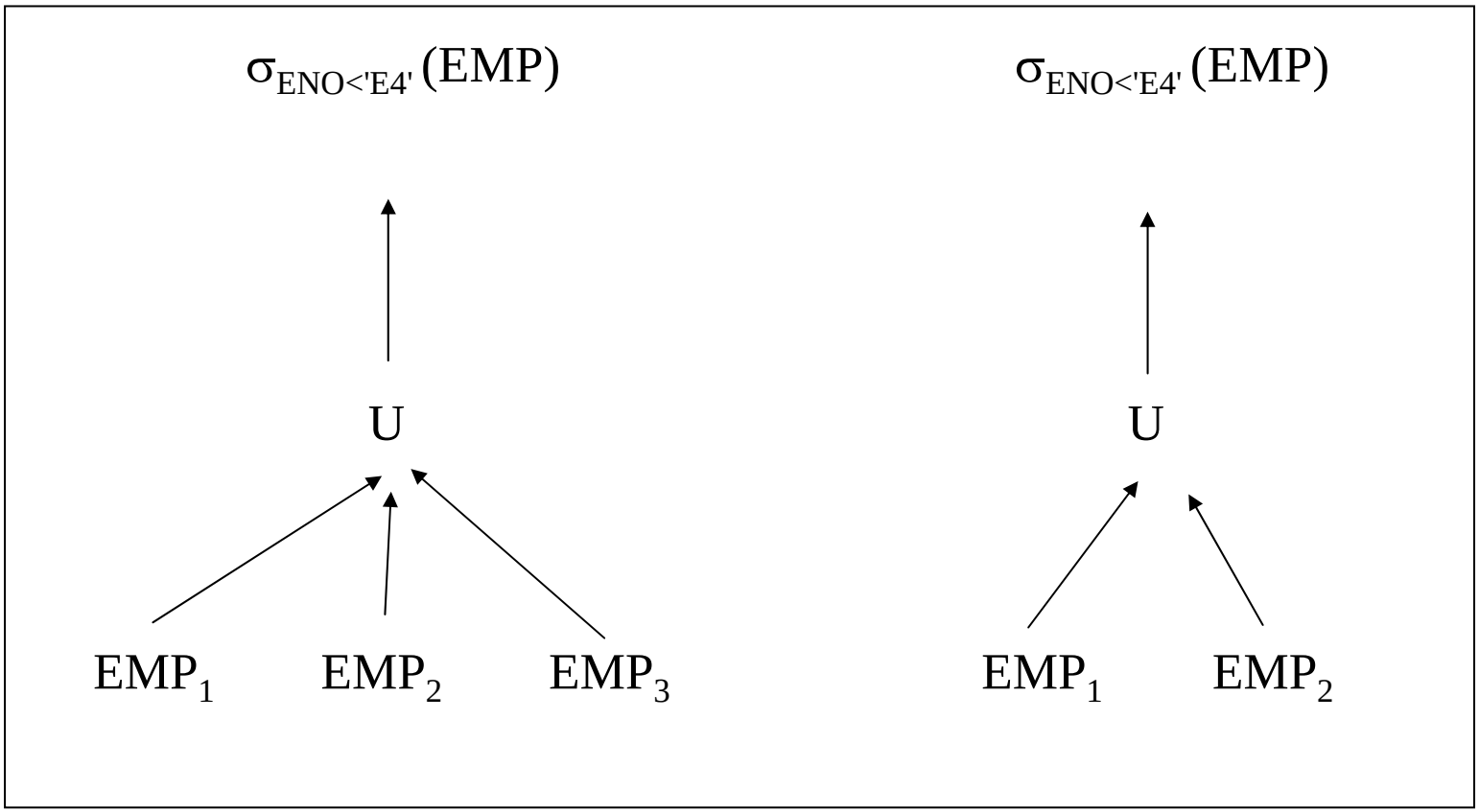
- Push operators down the tree as far as possible
  - Pushing down projections - Rule 4 & Rule 6
  - Pushing selects down - Rule 5



Horizontal Fragmentation Reduction - Select

- Assume that EMP = EMP_1 U EMP_2 U EMP_3
- $EMP_1 = \sigma_{ENO<='E3'}(EMP)$
- $EMP_2 = \sigma_{ENO>'E3' \wedge ENO<='E5'}(EMP)$
- $EMP_3 = \sigma_{ENO>'E5'}(EMP)$
- If query predicate is ENO < 'E4', then EMP_3 is excluded
- Formally, if R = {R_1, R_2, … R_n} and $R_j \in \sigma_{pj}(R)$,
  - Then $\sigma_{pj}(R_j) = \phi$ if $\forall x$ in R: $\neg(p_i(x) \wedge p_j(x))$

Select Reduction



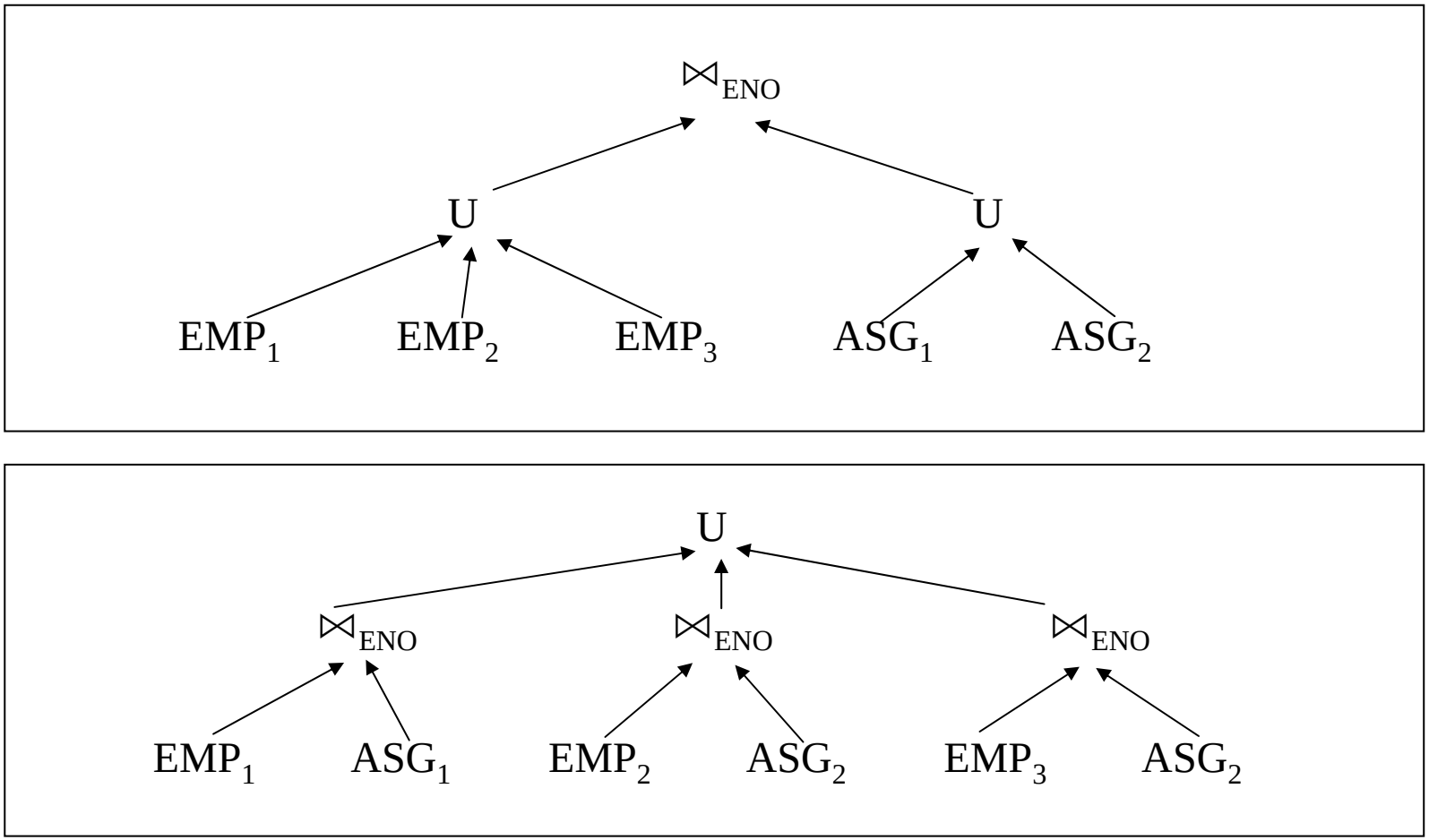
Join Reduction

- $(R_1 \cup R_2) \bowtie S = (R_1 \bowtie S) \cup (R_2 \bowtie S)$
- Sometimes this helps if you can eliminate some joins
- Not always effective, however
- Formally, $R_i \bowtie R_j = \phi$ if $\forall x$ in R_i, $\forall y$ in R_j: $\neg(p_i(x) \wedge p_j(y))$
- Example

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

  - $ASG_1 = \sigma_{ENO<='E3'}(ASG)$
  - $ASG_2 = \sigma_{ENO>'E3'}(ASG)$

Join Reduction Example



Vertical Fragmentation Reduction

- $EMP_1 = \Pi_{ENO,ENAME}$ (EMP)
- $EMP_2 = \Pi_{ENO,TITLE}$ (EMP)
- Reconstructing EMP = $EMP_1 \bowtie_{ENO} EMP_2$
- Formally, if K is key columns and $D \subset R$ are attributes to be projected, then $\Pi_{K,D}$ (R_j) is useless if $D \cap R_j = \phi$
- Example:

```
SELECT   ENAME
FROM     EMP
```

- EMP_2 is useless

Vertical Fragmentation Reduction Example