

---

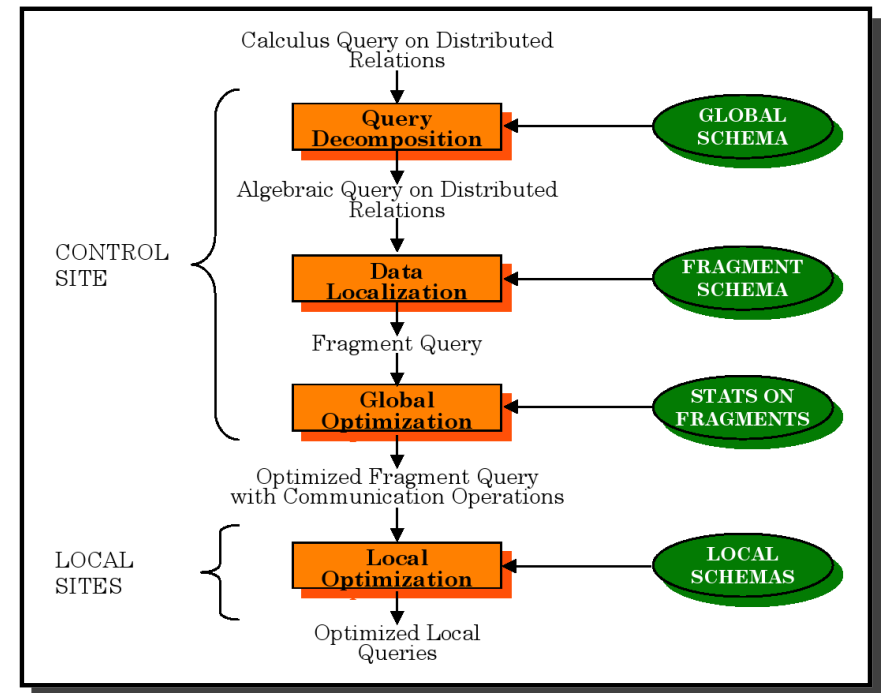
# Chapter 6: Query Decomposition and Data Localization

- Query Decomposition
- Data Localization

**Acknowledgements:** I am indebted to Arturas Mazeika for providing me his slides of this course.

# Query Decomposition

- **Query decomposition:** Mapping of calculus query (SQL) to algebra operations (select, project, join, rename)
- Both input and output queries refer to global relations, without knowledge of the distribution of data.
- The output query is semantically correct and good in the sense that redundant work is avoided.



- Query decomposition consists of 4 steps:
  1. **Normalization:** Transform query to a normalized form
  2. **Analysis:** Detect and reject "incorrect" queries; possible only for a subset of relational calculus
  3. **Elimination of redundancy:** Eliminate redundant predicates
  4. **Rewriting:** Transform query to RA and optimize query

# Query Decomposition – Normalization

---

- **Normalization:** Transform the query to a normalized form to facilitate further processing. Consists mainly of two steps.

## 1. **Lexical** and **syntactic** analysis

- Check validity (similar to compilers)
- Check for attributes and relations
- Type checking on the qualification

## 2. Put into **normal form**

- With SQL, the query qualification (WHERE clause) is the most difficult part as it might be an arbitrary complex predicate preceded by quantifiers ( $\exists, \forall$ )
- Conjunctive normal form

$$(p_{11} \vee p_{12} \vee \dots \vee p_{1n}) \wedge \dots \wedge (p_{m1} \vee p_{m2} \vee \dots \vee p_{mn})$$

- Disjunctive normal form

$$(p_{11} \wedge p_{12} \wedge \dots \wedge p_{1n}) \vee \dots \vee (p_{m1} \wedge p_{m2} \wedge \dots \wedge p_{mn})$$

- In the disjunctive normal form, the query can be processed as independent conjunctive subqueries linked by unions (corresponding to the disjunction)

# Query Decomposition – Normalization ...

---

- **Example:** Consider the following query: *Find the names of employees who have been working on project P1 for 12 or 24 months?*

- The query in SQL:

```
SELECT  ENAME
FROM    EMP, ASG
WHERE   EMP.ENO = ASG.ENO AND
          ASG.PNO = 'P1' AND
          DUR = 12 OR DUR = 24
```

- The qualification in conjunctive normal form:

$$EMP.ENO = ASG.ENO \wedge ASG.PNO = "P1" \wedge (DUR = 12 \vee DUR = 24)$$

- The qualification in disjunctive normal form:

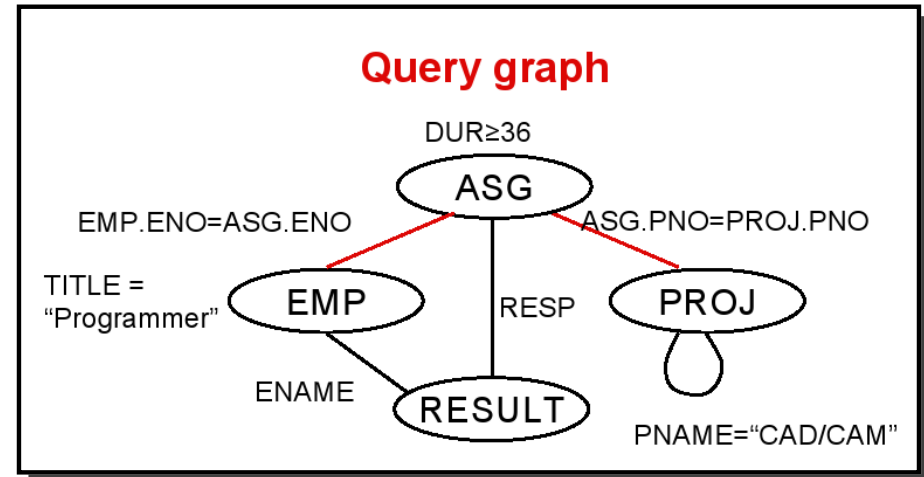
$$(EMP.ENO = ASG.ENO \wedge ASG.PNO = "P1" \wedge DUR = 12) \vee \\ (EMP.ENO = ASG.ENO \wedge ASG.PNO = "P1" \wedge DUR = 24)$$

- **Analysis:** Identify and reject type incorrect or semantically incorrect queries
- Type incorrect
  - Checks whether the attributes and relation names of a query are defined in the global schema
  - Checks whether the operations on attributes do not conflict with the types of the attributes, e.g., a comparison  $>$  operation with an attribute of type string
- Semantically incorrect
  - Checks whether the components contribute in any way to the generation of the result
  - Only a subset of relational calculus queries can be tested for correctness, i.e., those that do not contain disjunction and negation
  - Typical data structures used to detect the semantically incorrect queries are:
    - \* Connection graph (query graph)
    - \* Join graph

# Query Decomposition – Analysis ...

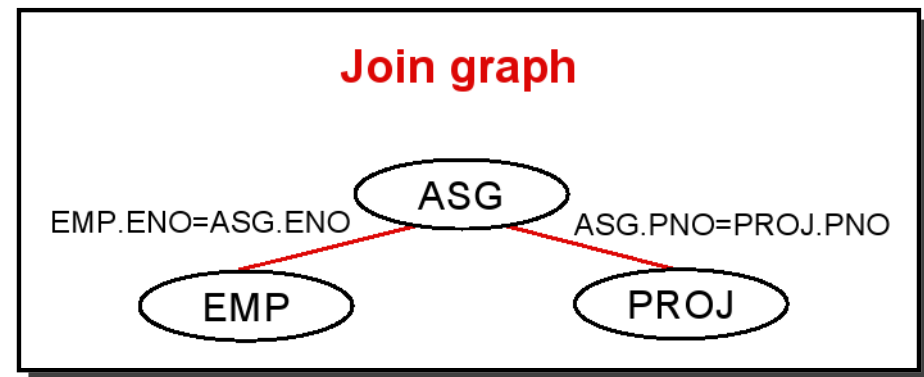
- **Example:** Consider a query:

```
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/connection graph
  - Nodes represent operand or result relation
  - Edge represents a join if both connected nodes represent an operand relation, otherwise it is a projection

- Join graph
  - a subgraph of the query graph that considers only the joins

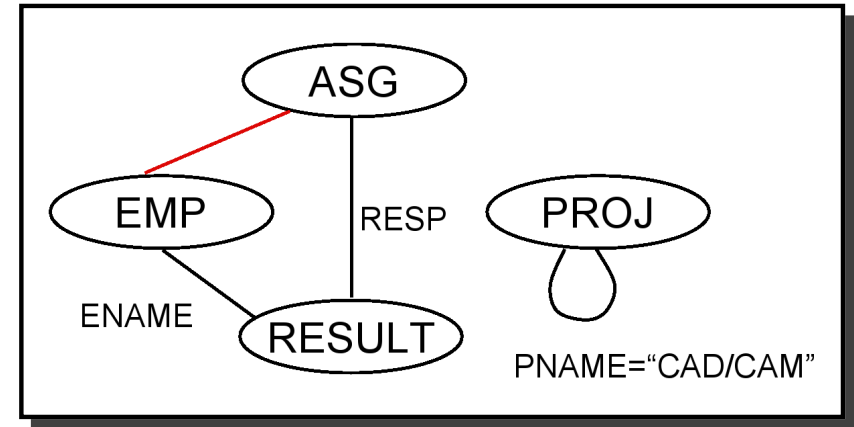


- Since the query graph **is connected**, the query is semantically correct

# Query Decomposition – Analysis ...

- **Example:** Consider the following query and its query graph:

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



- Since the graph **is not connected**, the query is semantically incorrect.
- 3 possible solutions:
  - Reject the query
  - Assume an implicit Cartesian Product between ASG and PROJ
  - Infer from the schema the missing join predicate  $ASG.PNO = PROJ.PNO$

# Query Decomposition – Elimination of Redundancy

---

- **Elimination of redundancy:** Simplify the query by eliminate redundancies, e.g., redundant predicates
  - Redundancies are often due to semantic integrity constraints expressed in the query language
  - e.g., queries on views are expanded into queries on relations that satisfies certain integrity and security constraints
- Transformation rules are used, e.g.,
  - $p \wedge p \iff p$
  - $p \vee p \iff p$
  - $p \wedge true \iff p$
  - $p \vee false \iff p$
  - $p \wedge false \iff false$
  - $p \vee true \iff true$
  - $p \wedge \neg p \iff false$
  - $p \vee \neg p \iff true$
  - $p_1 \wedge (p_1 \vee p_2) \iff p_1$
  - $p_1 \vee (p_1 \wedge p_2) \iff p_1$

- **Example:** Consider the following query:

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

- Let  $p_1$  be ENAME = "J. Doe",  $p_2$  be TITLE = "Programmer" and  $p_3$  be TITLE = "Elect. Eng."
- Then the qualification can be written as  $p_1 \vee (\neg p_2 \wedge (p_2 \vee p_3) \wedge \neg p_3)$  and then be transformed into  $p_1$
- Simplified query:

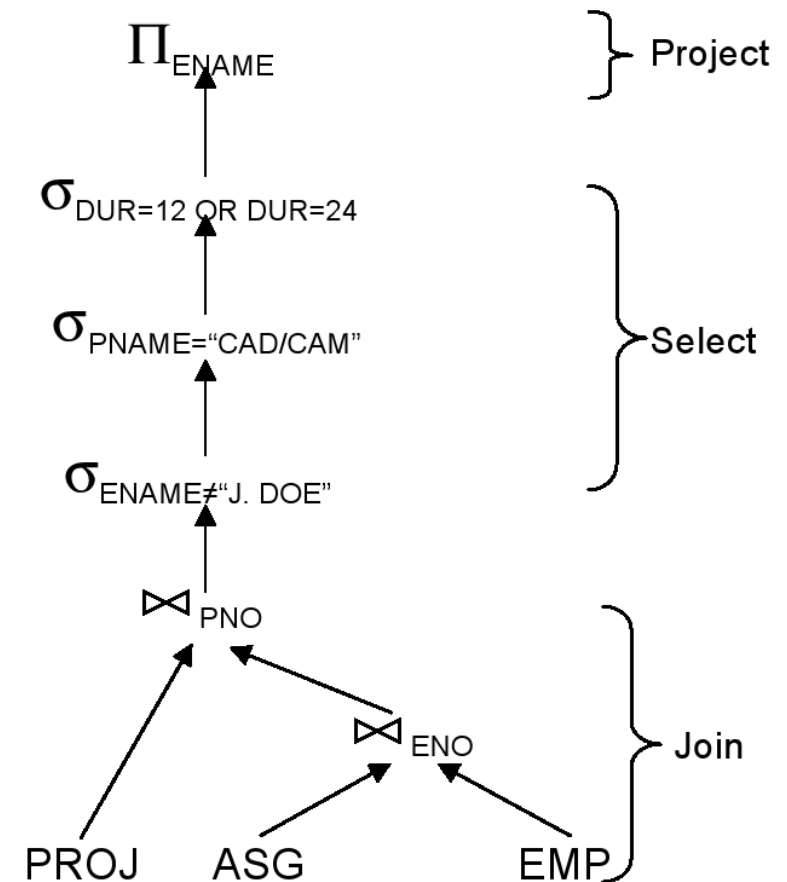
```
SELECT  TITLE
FROM    EMP
WHERE   EMP.ENAME = "J. Doe"
```

# Query Decomposition – Rewriting

- **Rewriting:** Convert relational calculus query to relational algebra query and find an **efficient** expression.
- **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)
```

- A **query tree** represents the RA-expression
  - Relations are leaves (FROM clause)
  - Result attributes are root (SELECT clause)
  - Intermediate leaves should give a result from the leaves to the root



- By applying **transformation rules**, many different trees/expressions may be found that are **equivalent** to the original tree/expression, but might be more efficient.
- In the following we assume relations  $R(A_1, \dots, A_n)$ ,  $S(B_1, \dots, B_n)$ , and  $T$  which is union-compatible to  $R$ .
- **Commutativity** of binary operations
  - $R \times S = S \times R$
  - $R \bowtie S = S \bowtie R$
  - $R \cup S = S \cup R$
- **Associativity** of binary operations
  - $(R \times S) \times T = R \times (S \times T)$
  - $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- **Idempotence** of unary operations
  - $\Pi_A(\Pi_A(R)) = \Pi_A(R)$
  - $\sigma_{p1(A1)}(\sigma_{p2(A2)}(R)) = \sigma_{p1(A1) \wedge p2(A2)}(R)$

- **Commuting selection** with binary operations

- $\sigma_{p(A)}(R \times S) \iff \sigma_{p(A)}(R) \times S$

- $\sigma_{p(A_1)}(R \bowtie_{p(A_2, B_2)} S) \iff \sigma_{p(A_1)}(R) \bowtie_{p(A_2, B_2)} S$

- $\sigma_{p(A)}(R \cup T) \iff \sigma_{p(A)}(R) \cup \sigma_{p(A)}(T)$

- \* ( $A$  belongs to  $R$  and  $T$ )

- **Commuting projection** with binary operations (assume  $C = A' \cup B'$ ,  $A' \subseteq A, B' \subseteq B$ )

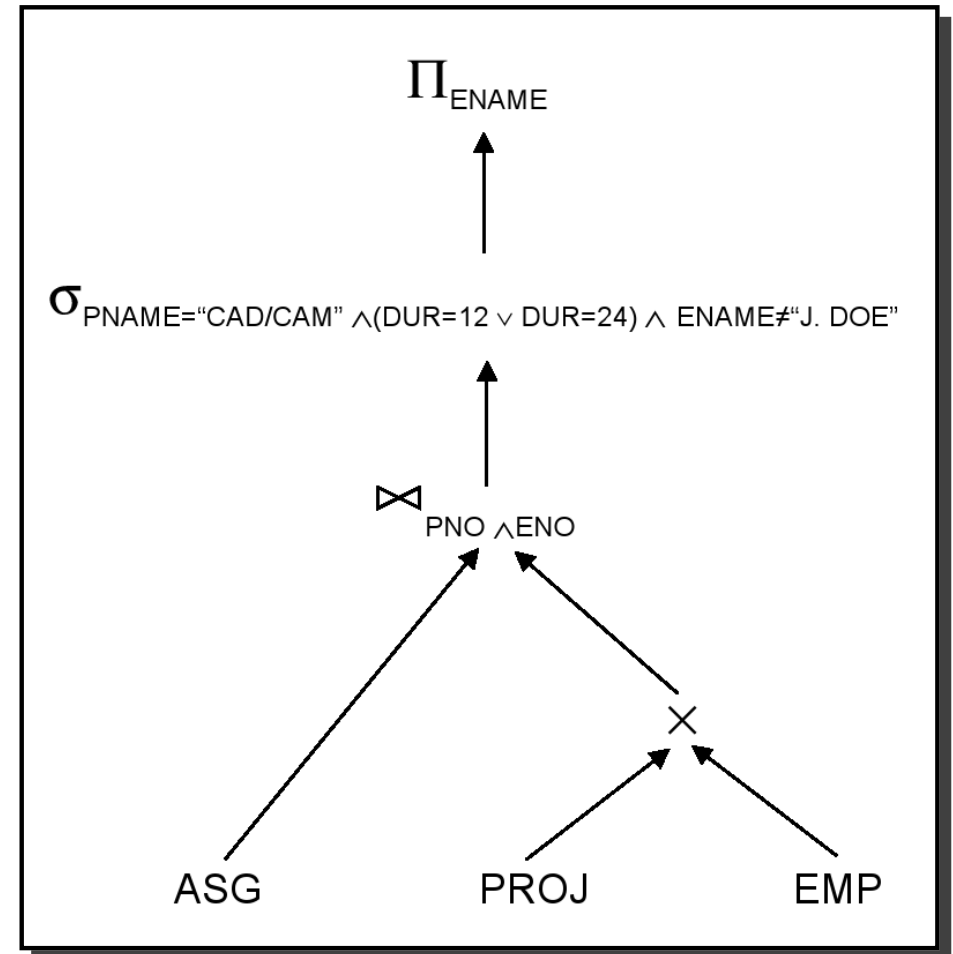
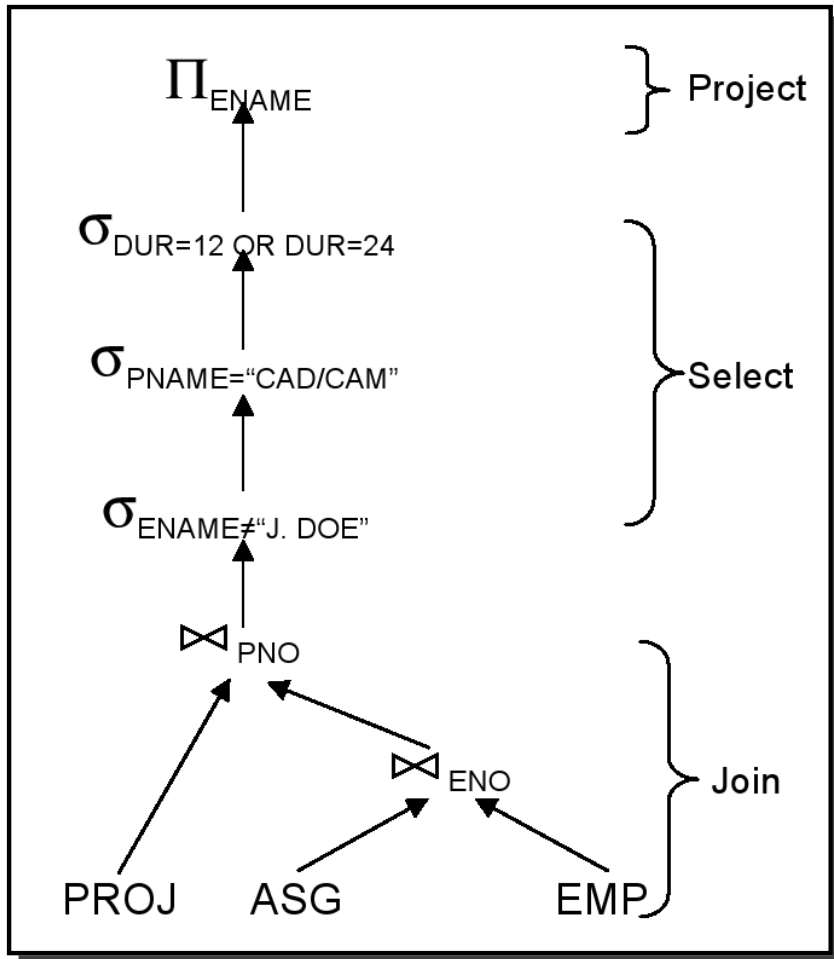
- $\Pi_C(R \times S) \iff \Pi_{A'}(R) \times \Pi_{B'}(S)$

- $\Pi_C(R \bowtie_{p(A', B')} S) \iff \Pi_{A'}(R) \bowtie_{p(A', B')} \Pi_{B'}(S)$

- $\Pi_C(R \cup S) \iff \Pi_C(R) \cup \Pi_C(S)$

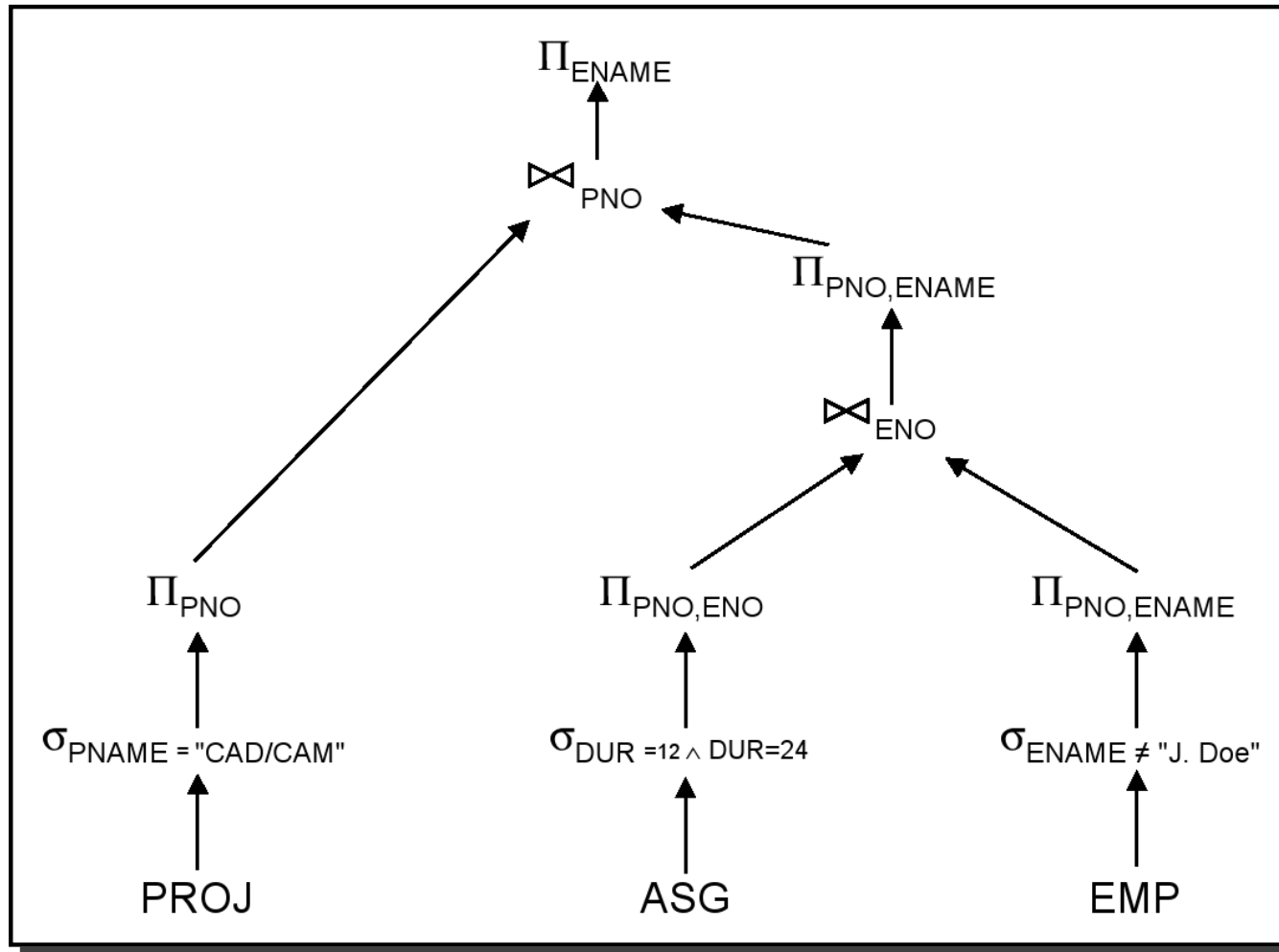
# Query Decomposition – Rewriting ...

- **Example:** Two equivalent query trees for the previous example
  - Recall the schemas: EMP(ENO, ENAME, TITLE)
  - PROJ(PNO, PNAME, BUDGET)
  - ASG(ENO, PNO, RESP, DUR)



# Query Decomposition – Rewriting ...

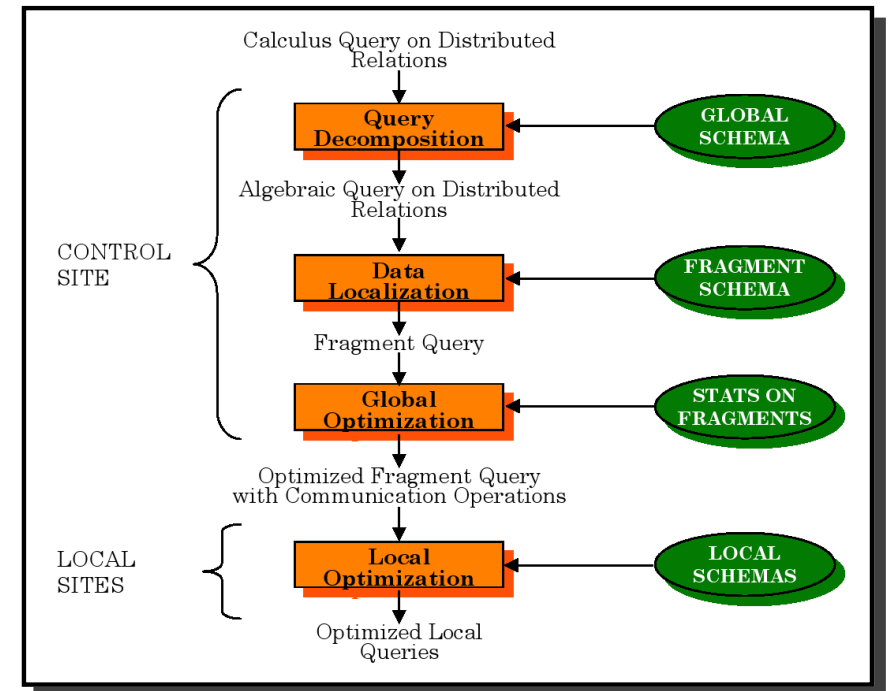
- **Example (contd.):** Another equivalent query tree, which allows a more efficient query evaluation, since the most selective operations are applied first.



# Data Localization

- **Data localization**

- **Input:** Algebraic query on global conceptual schema
- **Purpose:**
  - \* Apply data distribution information to the algebra operations and determine which fragments are involved
  - \* Substitute global query with queries on fragments
  - \* Optimize the global query



- **Example:**

- Assume EMP is horizontally fragmented into EMP1, EMP2, EMP3 as follows:

- \*  $EMP1 = \sigma_{ENO \leq "E3"}(EMP)$

- \*  $EMP2 = \sigma_{"E3" < ENO \leq "E6"}(EMP)$

- \*  $EMP3 = \sigma_{ENO > "E6"}(EMP)$

- ASG fragmented into ASG1 and ASG2 as follows:

- \*  $ASG1 = \sigma_{ENO \leq "E3"}(ASG)$

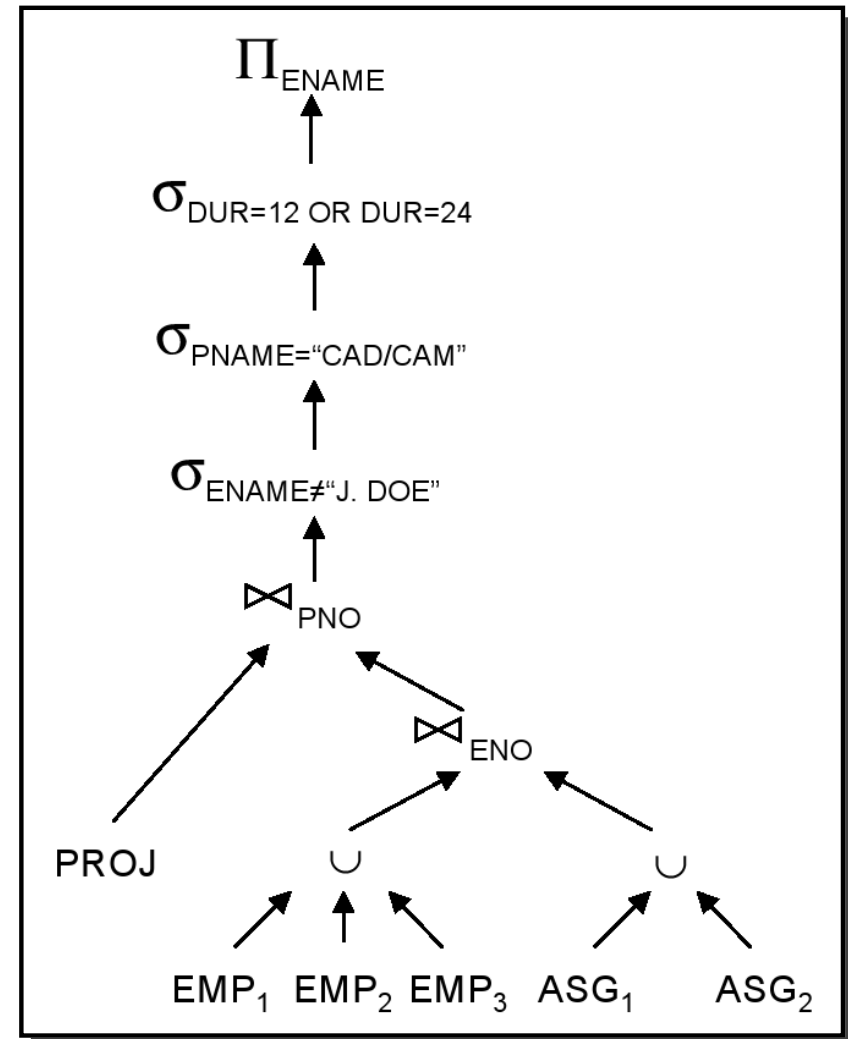
- \*  $ASG2 = \sigma_{ENO > "E3"}(ASG)$

- Simple approach: Replace in all queries

- EMP by  $(EMP1 \cup EMP2 \cup EMP3)$

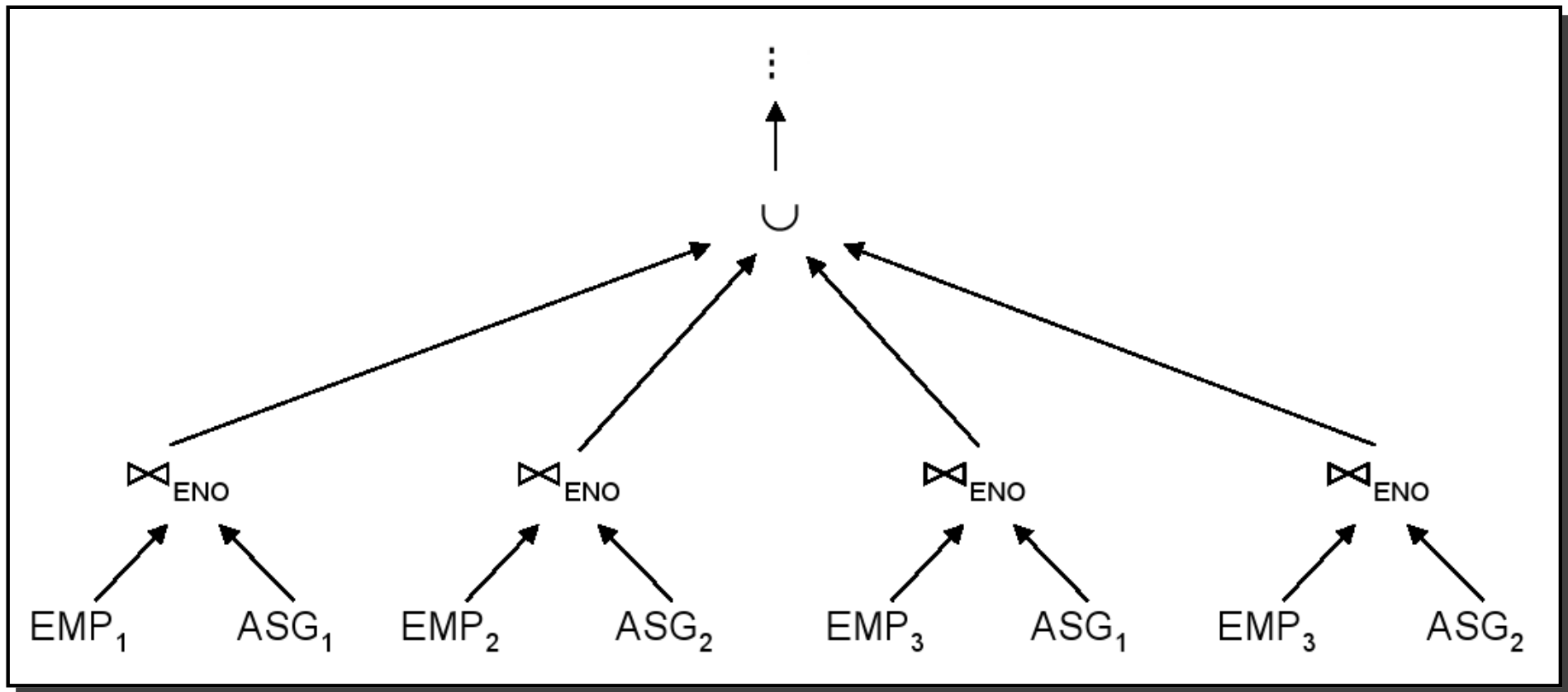
- ASG by  $(ASG1 \cup ASG2)$

- Result is also called **generic query**



- In general, the **generic query is inefficient** since important restructurings and simplifications can be done.

- **Example (contd.):** Parallelsim in the evaluation is often possible
  - Depending on the horizontal fragmentation, the fragments can be joined in parallel followed by the union of the intermediate results.

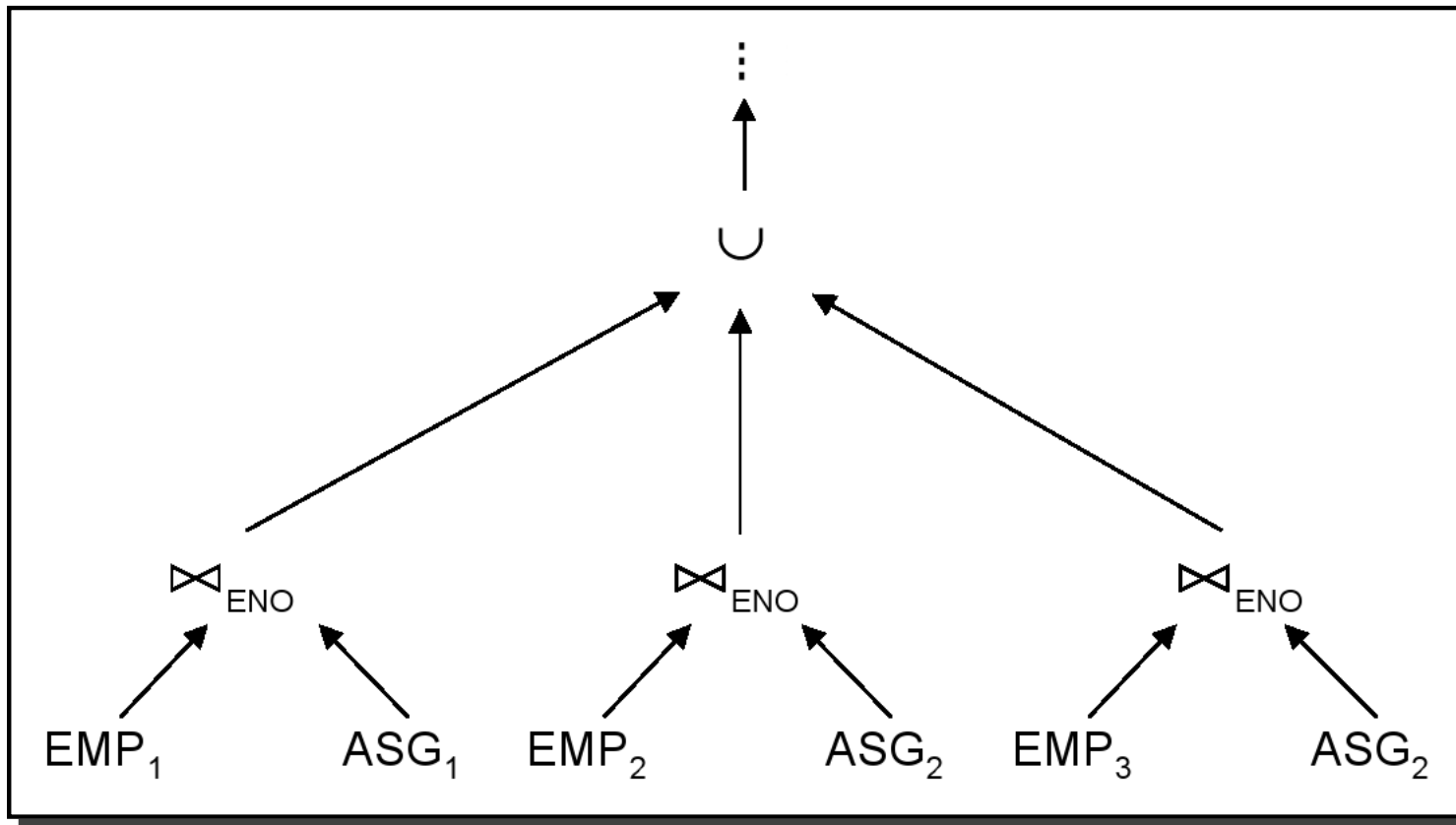


- **Example (contd.):** Unnecessary work can be eliminated

- e.g.,  $EMP_3 \bowtie ASG_1$  gives an empty result

- \*  $EMP_3 = \sigma_{ENO > "E6"}(EMP)$

- \*  $ASG_1 = \sigma_{ENO \leq "E3"}(ASG)$



- Various more advanced **reduction techniques** are possible to generate simpler and optimized queries.
- Reduction of horizontal fragmentation (HF)
  - Reduction with selection
  - Reduction with join
- Reduction of vertical fragmentation (VF)
  - Find empty relations

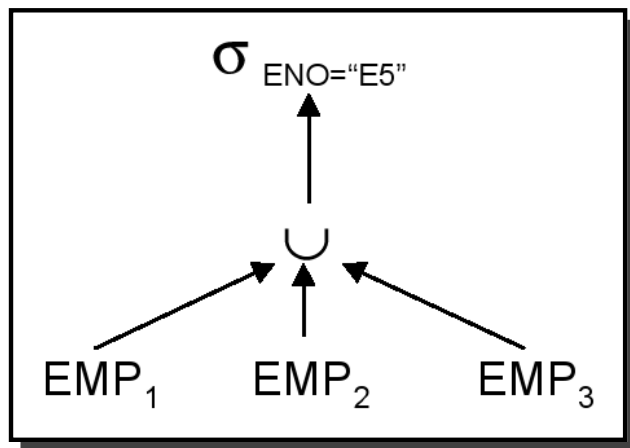
- **Reduction with selection for HF**

- Consider relation  $R$  with horizontal fragmentation  $F = \{R_1, R_2, \dots, R_k\}$ , where  $R_i = \sigma_{p_i}(R)$
- **Rule1:** Selections on fragments,  $\sigma_{p_j}(R_i)$ , that have a qualification contradicting the qualification of the fragmentation generate empty relations, i.e.,

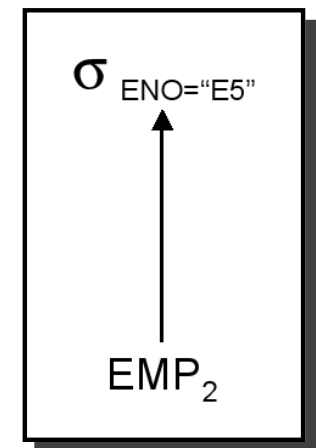
$$\sigma_{p_j}(R_i) = \emptyset \iff \forall x \in R(p_i(x) \wedge p_j(x) = false)$$

- Can be applied if fragmentation predicate is inconsistent with the query selection predicate.

- **Example:** Consider the query: **SELECT \* FROM EMP WHERE ENO="E5"**



After commuting the selection with the union operation, it is easy to detect that the selection predicate contradicts the predicates of  $EMP_1$  and  $EMP_3$ .



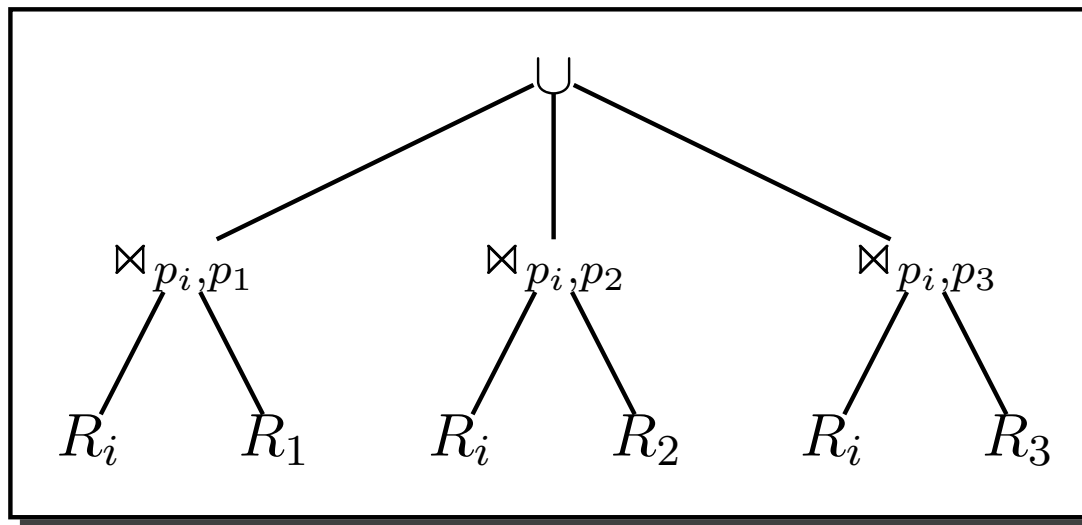
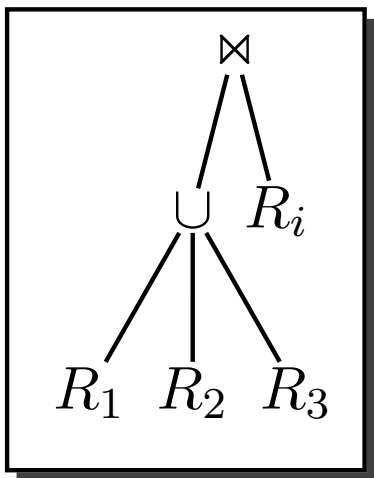
- **Reduction with join for HF**

- Joins on horizontally fragmented relations can be simplified when the joined relations are fragmented according to the join attributes.
- Distribute join over union

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

- **Rule 2:** Useless joins of fragments,  $R_i = \sigma_{p_i}(R)$  and  $R_j = \sigma_{p_j}(R)$ , can be determined when the qualifications of the joined fragments are contradicting, i.e.,

$$R_i \bowtie R_j = \emptyset \iff \forall x \in R_i, \forall y \in R_j (p_i(x) \wedge p_j(y) = \text{false})$$



# Data Localizations Issues – Reduction for HF ...

- **Example:** Consider the following query and fragmentation:

- Query: **SELECT \* FROM EMP, ASG WHERE EMP.ENO=ASG.ENO**

- Horizontal fragmentation:

- \*  $EMP_1 = \sigma_{ENO \leq "E3"}(EMP)$

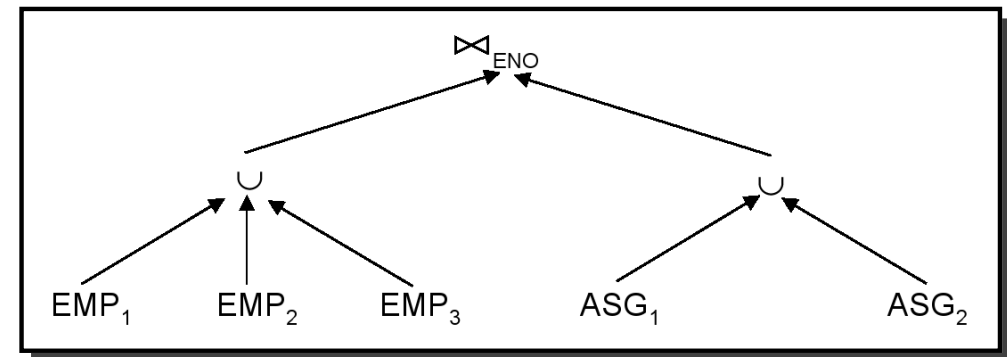
- \*  $EMP_2 = \sigma_{"E3" < ENO \leq "E6"}(EMP)$

- \*  $EMP_3 = \sigma_{ENO > "E6"}(EMP)$

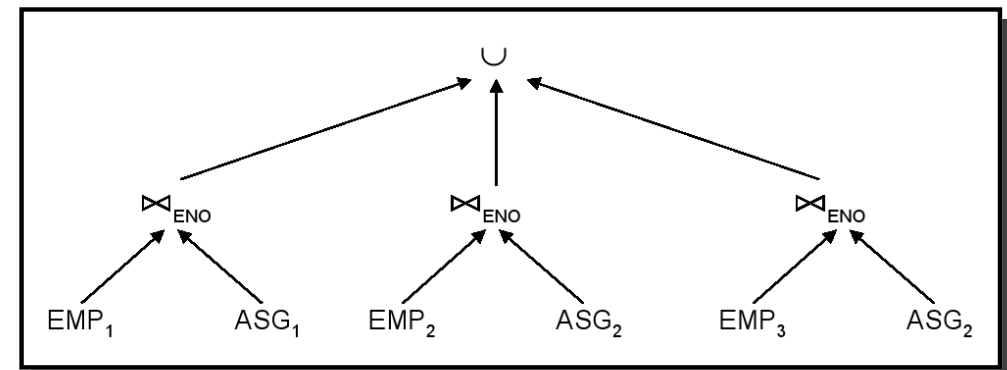
- \*  $ASG_1 = \sigma_{ENO \leq "E3"}(ASG)$

- \*  $ASG_2 = \sigma_{ENO > "E3"}(ASG)$

- Generic query



- The query reduced by distributing joins over unions and applying rule 2 can be implemented as a union of three partial joins that can be done in parallel.



- **Reduction with join for derived HF**

- The horizontal fragmentation of one relation is **derived** from the horizontal fragmentation of another relation by using semijoins.
- If the fragmentation is not on the same predicate as the join (as in the previous example), derived horizontal fragmentation can be applied in order to make efficient join processing possible.
- **Example:** Assume the following query and fragmentation of the EMP relation:
  - Query: **SELECT \* FROM EMP, ASG WHERE EMP.ENO=ASG.ENO**
  - Fragmentation (**not** on the join attribute):
    - \*  $EMP_1 = \sigma_{TITLE="Prgrammer"}(EMP)$
    - \*  $EMP_2 = \sigma_{TITLE \neq "Prgrammer"}(EMP)$
  - To achieve efficient joins ASG can be fragmented as follows:
    - \*  $ASG_1 = ASG \bowtie_{ENO} EMP_1$
    - \*  $ASG_2 = ASG \bowtie_{ENO} EMP_2$
  - The fragmentation of ASG is derived from the fragmentation of EMP
  - Queries on derived fragments can be reduced, e.g.,  $ASG_1 \bowtie EMP_2 = \emptyset$

- **Reduction for Vertical Fragmentation**

- Recall, VF distributes a relation based on projection, and the reconstruction operator is the join.
- Similar to HF, it is possible to identify useless intermediate relations, i.e., fragments that do not contribute to the result.
- Assume a relation  $R(A)$  with  $A = \{A_1, \dots, A_n\}$ , which is vertically fragmented as  $R_i = \pi_{A'_i}(R)$ , where  $A'_i \subseteq A$ .
- **Rule 3:**  $\pi_{D,K}(R_i)$  is useless if the set of projection attributes  $D$  is not in  $A'_i$  and  $K$  is the key attribute.
- Note that the result is not empty, but it is useless, as it contains only the key attribute.

- **Example:** Consider the following query and vertical fragmentation:

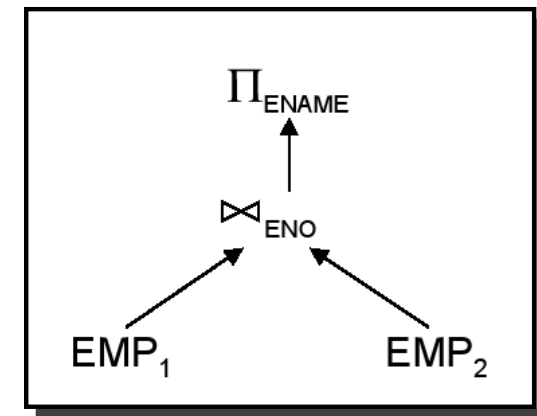
- Query: **SELECT ENAME FROM EMP**

- Fragmentation:

- \*  $EMP_1 = \Pi_{ENO,ENAME}(EMP)$

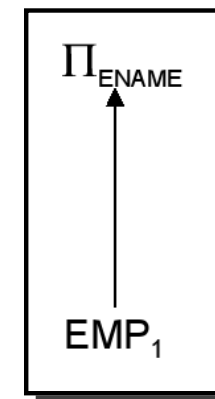
- \*  $EMP_2 = \Pi_{ENO,TITLE}(EMP)$

- Generic query



- Reduced query

- By commuting the projection with the join (i.e., projecting on ENO, ENAME), we can see that the projection on  $EMP_2$  is useless because ENAME is not in  $EMP_2$ .



# Conclusion

---

- Query decomposition and data localization maps calculus query into algebra operations and applies data distribution information to the algebra operations.
- Query decomposition consists of normalization, analysis, elimination of redundancy, and rewriting.
- Data localization reduces horizontal fragmentation with join and selection, and vertical fragmentation with joins, and aims to find empty relations.