An SQL query: 

```sql
SELECT ENAME 
FROM EMPLOYEE, WORKSON, PROJECT 
WHERE PNAME = 'database' AND 
PNUM = PNO AND 
ENO = ENUM AND 
BDATE > '1965'
```

This query tree (procedure) will compute the correct result. However, the performance will be very poor.  

⇒ needs optimization!
**Optimization Strategies**

**GOAL**: Reducing the sizes of the intermediate results as quickly as possible.

**STRATEGY**:

1. Move SELECTs and PROJECTs as far down the query tree as possible.
2. Among SELECTs, reordering the tree to perform the one with lowest selectivity factor first.
3. Among JOINs, reordering the tree to perform the one with lowest join selectivity first.
Example: Apply SELECTs First

Canonical Query Tree

SELECT Clause

WHERE Clause

FROM Clause

After Optimization
Example: Replace “σ − ×” by “▽”

**Before Optimization**

πENAME

σPNUM = PNO

×

σENUM = ENO

σPNAME = ‘database’

σBDATE > ‘1965’

PROJECT

WORKSON

EMPLOYEE

**After Optimization**

πENAME

△PNUM = PNO

△ENUM = ENO

△PNAME = ‘database’

σBDATE > ‘1965’

WORKSON

PROJECT

EMPLOYEE
Example: Move PROJECTs Down

Before Optimization

\[ \pi_{\text{ENAME}} \]
\[ \sigma_{\text{BDATE} > '1965'} \]
\[ \pi_{\text{ENAME}} \]
\[ \pi_{\text{ENAME}, \text{PNO}} \]
\[ \sigma_{\text{PNAME} = 'database'} \]
\[ \pi_{\text{PNUM}} \]

After Optimization

\[ \pi_{\text{ENAME}} \]
\[ \sigma_{\text{BDATE} > '1965'} \]
\[ \pi_{\text{ENAME}, \text{PNO}} \]
\[ \pi_{\text{ENAME}, \text{ENUM}} \]
\[ \pi_{\text{ENO}, \text{PNO}} \]
\[ \pi_{\text{PNUM}} \]

\[ \text{ENAME} \]
\[ \text{PNUM} = \text{PNO} \]
\[ \text{ENUM} = \text{ENO} \]
\[ \text{WORKSON} \]
\[ \text{PROJECT} \]

\[ \text{EMPLOYEE} \]
Localization of Distributed Data

**GOAL:** Translate an algebraic query on global relations into an algebraic query expressed on physical fragments.

**STRATEGY:**

1. Generate the query tree by replacing the leaves of the tree with subtrees corresponding to the reconstruction programs.
2. Apply the reduction techniques to simplify the generic query tree.

---

### Reduction Technique for Primary Horizontal Fragmentation (1)

**Fragmentation Schema**

\[
E_1 = \sigma_{ENO=E5}(E) \\
E_2 = \sigma_{ENO=E5\cup E6}(E) \\
E_3 = \sigma_{ENO=E6}(E)
\]

**Query**

```
SELECT * 
FROM E 
WHERE ENO="E5"
```
Reduction Technique for Primary Horizontal Fragmentation (2)

Commuting SELECT with UNIONs

Remove SELECTs that produce empty relations.

Fragmentation Schema

\[ E_1 = \sigma_{\text{ENO} \leq E5}^{E5}(E) \]
\[ E_2 = \sigma_{\text{ENO} \leq E5 < E6}^{E5}(E) \]
\[ E_3 = \sigma_{\text{ENO} > E6}^{E5}(E) \]

Reduction Technique for Vertical Fragmentation

Query

SELECT ENAME FROM EMP

Original Query Tree:

\[ \Pi_{\text{ENAME}} \]
\[ \Pi_{\text{ENAME}} \]

(a) Generic query
(b) Reduced query

Fragmentation Schema

\[ E_1 = \Pi_{\text{ENO}, \text{ENAME}}(EMP) \]
\[ E_2 = \Pi_{\text{ENO}, \text{ETITLE}}(EMP) \]
Reduction for Derived Fragmentation

Fragmentation Schema

\[ EMP_1 = \sigma_{\text{title} = \text{Mech. Eng}}(EMP) \]
\[ EMP_2 = \sigma_{\text{title} = \text{Mech. Eng}}(EMP) \]
\[ ASG_1 = ASG \Join_{ENO} EMP_1 \]
\[ ASG_2 = ASG \Join_{ENO} EMP_2 \]

\[ \text{SELECT } * \]
\[ \text{FROM EMP, ASG} \]
\[ \text{WHERE ASG.ENO = EMP.ENO} \]
\[ \text{AND TITLE = "Mech. Eng"} \]

Reduction Techniques for Hybrid Fragmentation

1. Remove empty relations generated by contradicting selections on horizontal fragments.
2. Remove useless relations generated by projections on vertical fragments.
3. Distribute joins over unions in order to isolate and remove useless joins.

\[ EMP_1 = \sigma_{\text{ENO}=\text{EN}}(\Pi_{\text{ENO},\text{ENAME}}(EMP)) \]
\[ EMP_2 = \sigma_{\text{ENO}=\text{EN}}(\Pi_{\text{ENO},\text{ENAME}}(EMP)) \]
\[ EMP_3 = \Pi_{\text{ENO},\text{TITLE}}(EMP) \]

\[ \text{SELECT ENAME} \]
\[ \text{FROM EMP} \]
\[ \text{WHERE ENO = "E5";} \]
\[ EMP_1 = \sigma_{ENO='E4'}(\Pi_{ENO,ENAME}(EMP)) \]
\[ EMP_2 = \sigma_{ENO='E4'}(\Pi_{ENO,ENAME}(EMP)) \]
\[ EMP_3 = \Pi_{ENO,TITLE}(EMP) \]