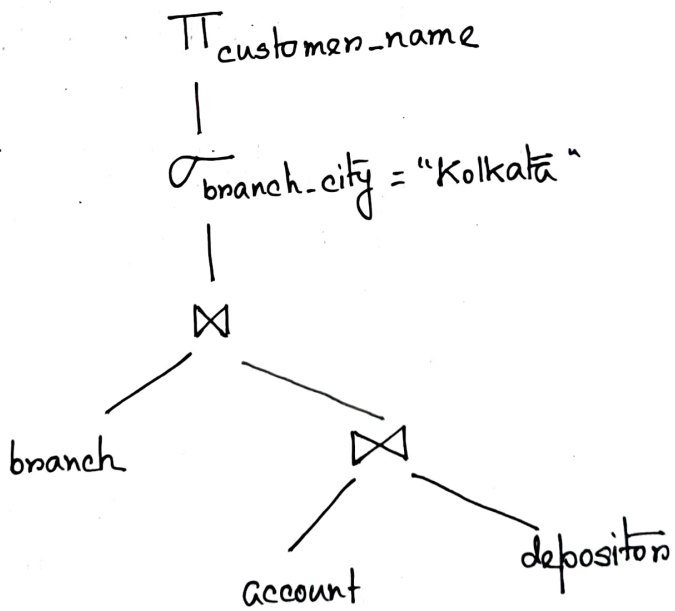
Query Optimization

Query Tree : A query tree is a data structure that corresponds to a relational algebra expression. It represents the input relations of the query as leaf nodes of the tree, and represents the relational algebra operations as internal nodes.

Consider, Branch { branch_name, branch_city, assets }
Account { account_name, branch_name, balance }
Depositor { customer_name, account_name }

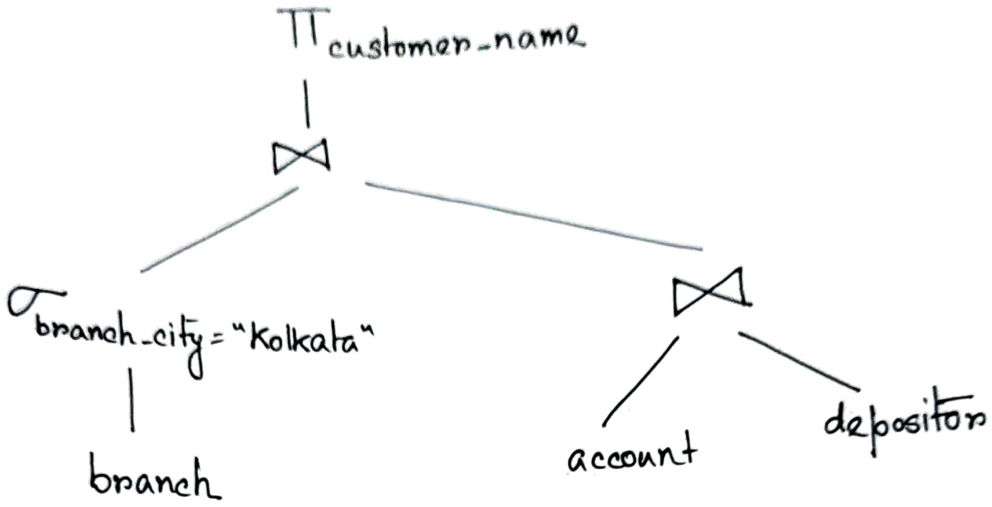
Find the names of all customers who have an account at any branch located in Kolkata.

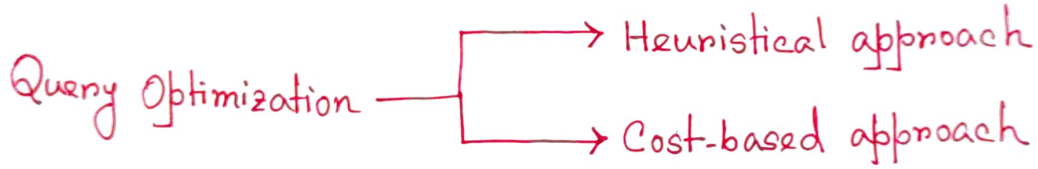
$$\Pi_{customer\_name}\left(\sigma_{branch\_city = \text{"Kolkata"}}\left(branch \bowtie (account \bowtie depositor)\right)\right)$$



Initial expression/query tree

$\Pi_{customer\_name}((\sigma_{branch\_city = "Kolkata"}(branch)) \bowtie (account \bowtie depositor))$





Heuristical approach :- In heuristical approach of query optimization which uses transformation rules to convert one relational algebra expression into an equivalent form that becomes efficient.

Cost-based approach : It uses formulate that estimates and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate.

Equivalence Rules

1. Conjunctive selection operation can be deconstructed into individual selections

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Selection operations are commutative

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Only the final operations in a sequence of projection operations are needed

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E)) = \Pi_{L_1}(E)$$

4. Selections can be combined with Cartesian products and theta joins

a) $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$

b) $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

5. Theta join operations are commutative

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

6. a) Natural join operations are associative

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

b) Theta joins are associative

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3$$

7. Selection operation distributes over theta-join operation

a) $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta_2} E_2$

b) $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta_3} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta_3} (\sigma_{\theta_2}(E_2))$

8. Projection operation distributes over theta-join operation

a) $\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$

b) $\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\Pi_{L_2 \cup L_4}(E_2)))$

9. Set operations intersection and union are commutative

$$E_1 \cap E_2 = E_2 \cap E_1$$

$$E_1 \cup E_2 = E_2 \cup E_1$$

* Set difference is not commutative.

10. Set union and intersection are associative

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. Selection operation distributes over union, intersection and set-difference operation

$$\sigma_p(E_1 - E_2) = \sigma_p(E_1) - \sigma_p(E_2)$$

12. Projection operation distributes over union operation

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Heuristical approach :

Consider, Branch { branch_name, branch_city, assets }
Account { account_number, branch_name, balance }
Depositor { customer_name, account_number }

* Find the name of the customers who have an account at any branch located in Kolkata and who have a balance greater than 1000

$$\Pi_{customer\_name}(\sigma_{branch\_city = "Kolkata" \wedge balance > 1000}(branch \bowtie (account \bowtie depositor)))$$

apply rule 6.a (associativity of natural join)

$$\Pi_{customer\_name}((\sigma_{branch\_city = "Kolkata" \wedge balance > 1000}((branch \bowtie account) \bowtie depositor)))$$

apply rule 7.a

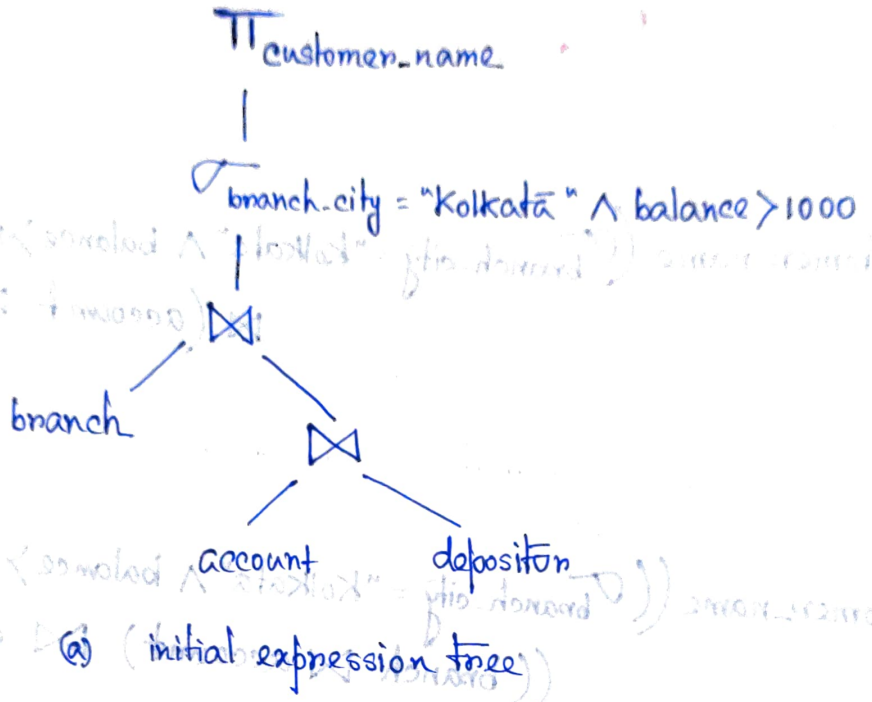
$$\Pi_{customer\_name}((\sigma_{branch\_city = "Kolkata" \wedge balance > 1000}(branch \bowtie account)) \bowtie depositor)$$

apply rule 1.

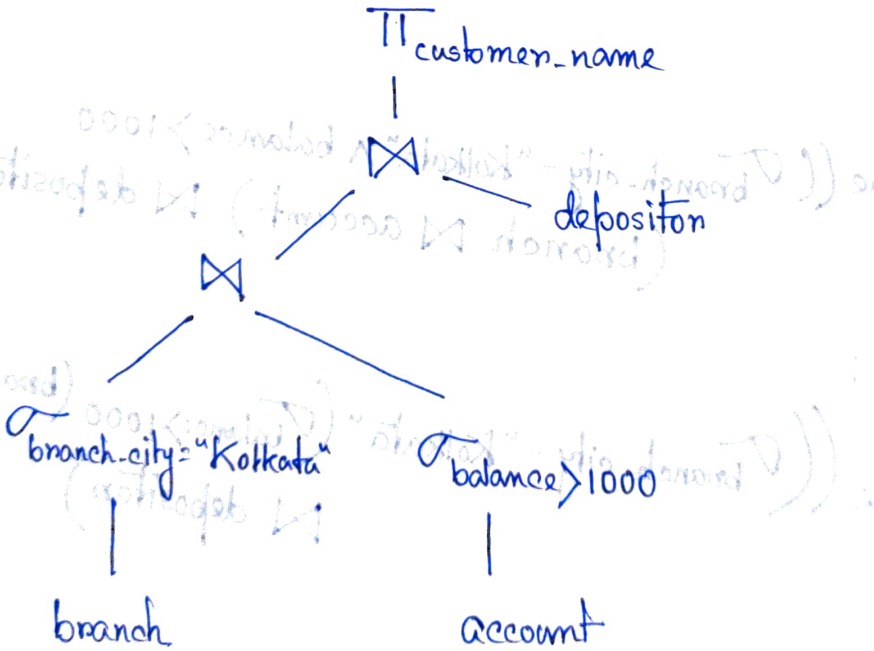
$$\Pi_{customer\_name}(((\sigma_{branch\_city = "Kolkata"}(\sigma_{balance > 1000}(branch \bowtie account))) \bowtie depositor)$$

apply rule 7.b

$$\Pi_{customer\_name}((\sigma_{branch\_city = "Kolkata"}(branch) \bowtie \sigma_{balance>1000}(account \bowtie depositor))$$



(a) initial expression tree



(b) Tree after multiple transformation

mple:

Employee { Fname, Lname, SSN, Bdate, Address, Salary, SuperSSN, DNo }

Department { Dname, Dnumber, MGRSSN, MGRStartDate }

Dept_location { Dnumber, Dlocation }

Project { Pname, Pnumber, Plocation, Dnum }

Works_ON { ESSN, Pno, Hours }

Dependent { ESSN, Dependent_name, Bdate, relationship }

* Find the last names of employees born after 1957 who work on a project named 'Thermal'.

(a)

$$\pi_{Lname}$$

$$\sigma_{Pname = \text{'Thermal'} \wedge Pnumber = Pno \wedge ESSN = SSN \wedge Bdate > \text{'1957-12-3}}$$

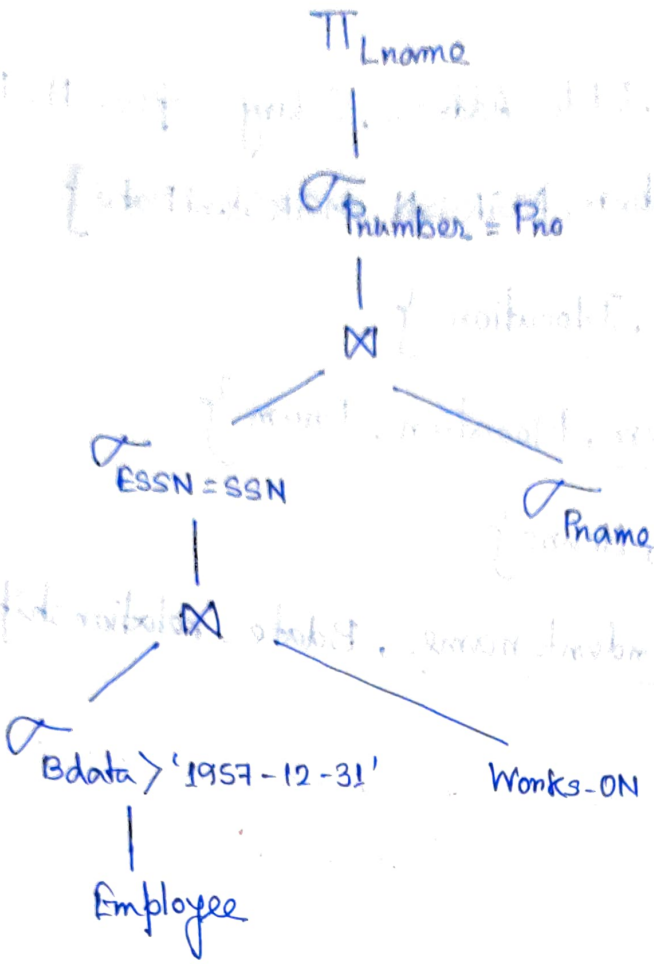
$$\bowtie$$

Project

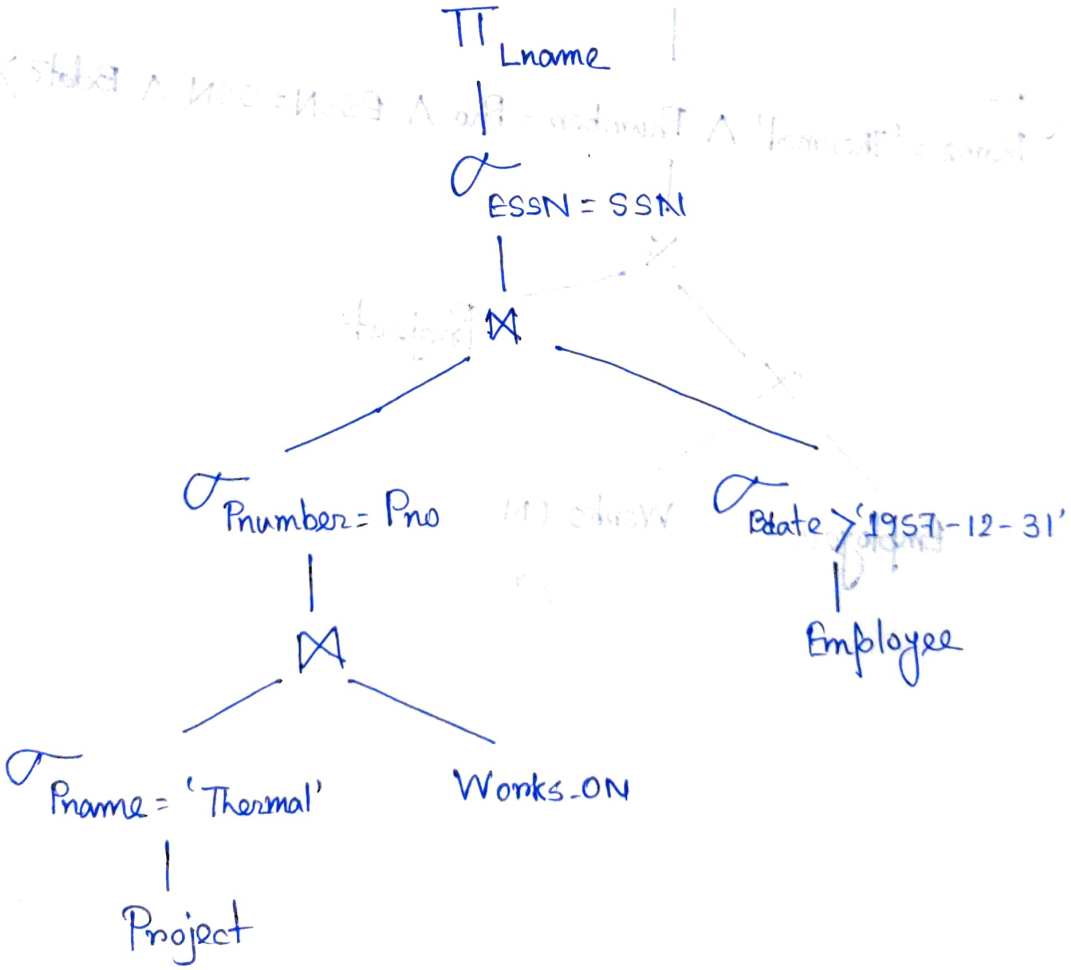
$$\bowtie$$

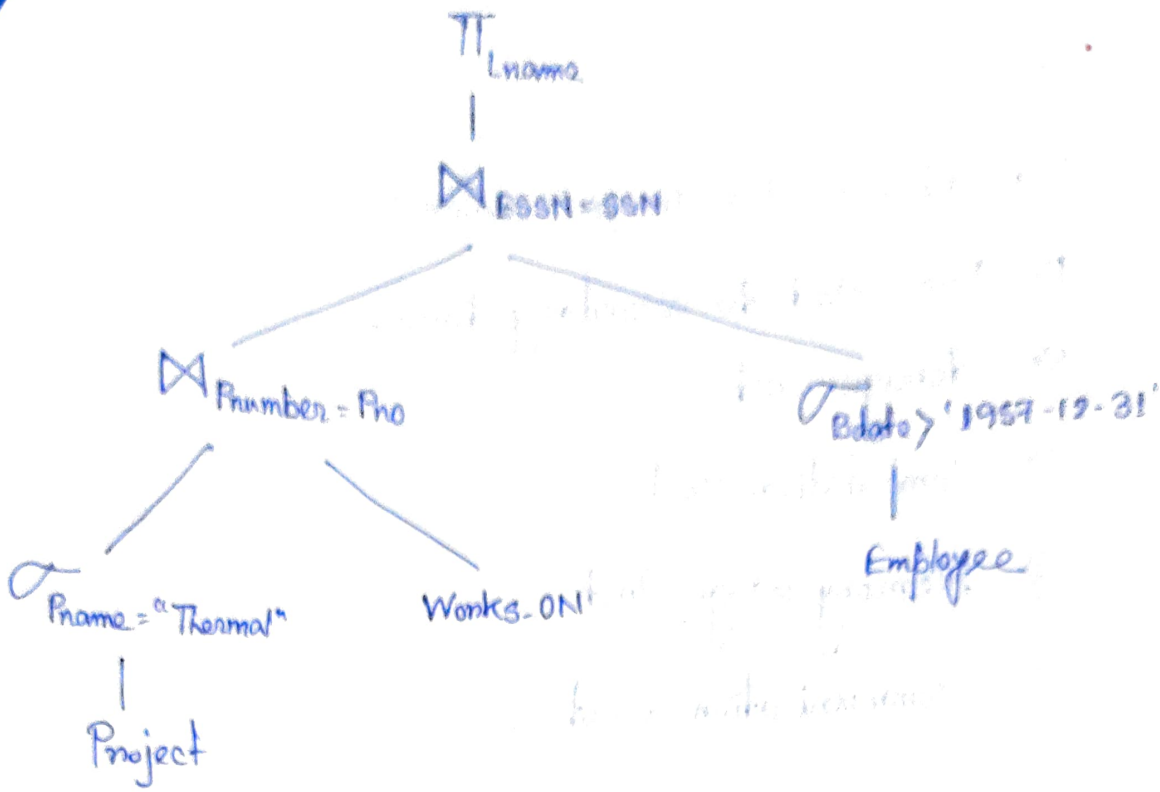
Employee Works_ON

(b)

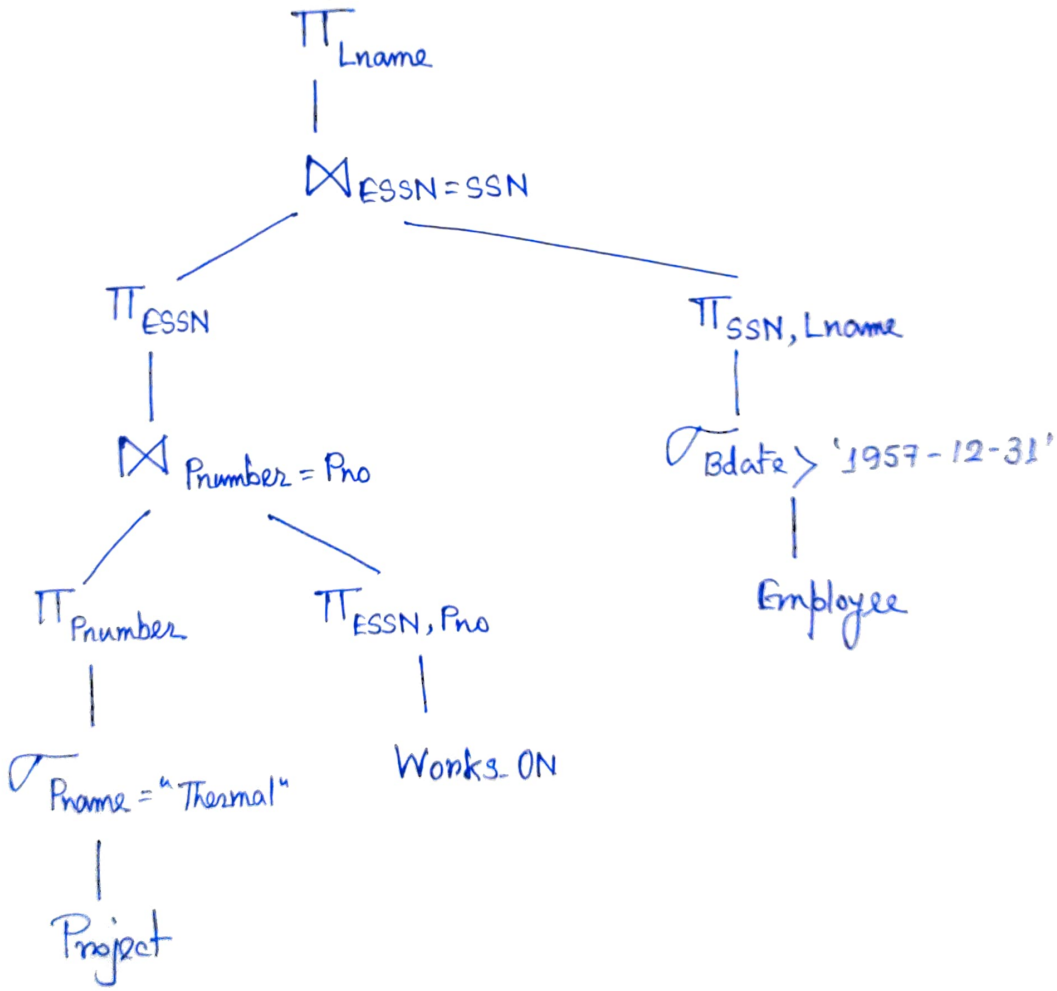

π_{Lname}
$\sigma_{Pnumber = Pno}$
$\bowtie$
$\sigma_{ESSN = SSN}$
$\sigma_{Pname = "Thermal"}$
$\bowtie$
Project
$\sigma_{Bdata > '1957-12-31'}$
Works_ON
Employee

(c)


π_{Lname}
$\sigma_{ESSN = SSN}$
$\bowtie$
$\sigma_{Pnumber = Pno}$
$\sigma_{Bdate > '1957-12-31'}$
$\bowtie$
Employee
$\sigma_{Pname = 'Thermal'}$
Works_ON
Project



(e)



*** SELECT - PROJECT - JOIN

Cost-based Approach

Cost component for Query Execution :-

1) Access Cost to secondary storage
2) Storage Cost
3) Computation Cost
4) Memory usage Cost
5) Communication Cost .