# Chapter 7
# Query Optimization

## Introduction

Query Processing refers to range of activities involved in extracting data from a database. The basic steps involved in processing of a query are

1. Parsing and translation
2. Optimization
3. Evaluation

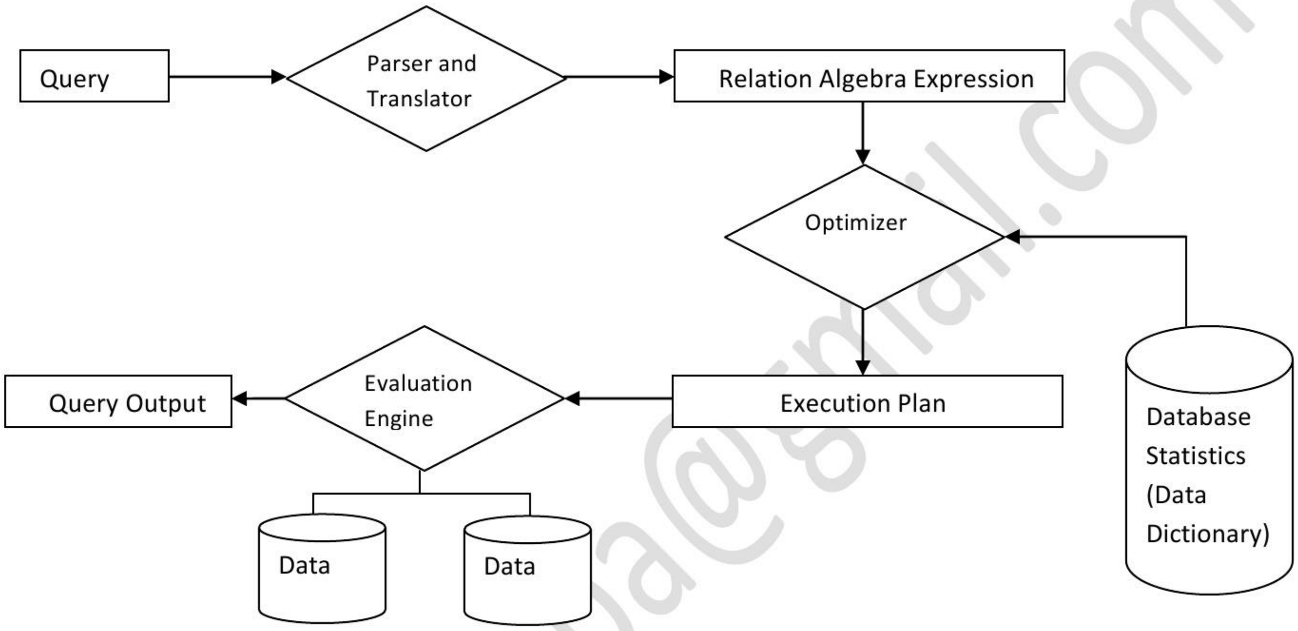


Fig: Steps in Query Processing

**1. Parsing and Translation**

The first step in any query processing system is to translate a given query into its internal form. This translation process is similar to the work performed by the parser of a compiler. In generating the internal from of the query, the parser check the syntax of the user's query, verifies that the relation is formulated according to the syntax rules of the query language. Then this is translated into relational algebra.

**2. Optimization**

A relational algebra expression may have many equivalent expressions.

`E.g.` $\sigma_{balance<2500}(\prod_{balance}(\textbf{account}))$ is equivalent to $\prod_{balance}(\sigma_{balance<2500}(\textbf{account}))$

We can execute each relation algebra operation by one of several different execute algorithms. The process of choosing a suitable one with lowest cost is known as query optimization. Cost is estimated using the statistical information from database catalog. The different statistical information is number of tuples in each relation, size of tuples etc. So among all equivalent expressions, choose the one with the cheapest possible evaluation plan (one of the possible way of executing a query).

**3. Execution**

The query execution engine takes a query evaluation plan, executes that plan and returns the answer to the query.

## Query Cost Estimation

Each query is translated into a number of semantically equivalent plans. So there are several alternatives, now the question is which one is the most efficient evaluation plan to be selected for execution. To get the answer, the cost for all alternatives must be estimated and the plan with lowest cost is selected. Since a database resides on disk, often the cost of reading and writing to disk dominates the cost of processing a query.

We can choose a strategy based on reliable information, database systems may store statistics (metadata) for each relation R. These statistics includes number of tuples in a relation, size of tuples in a relation etc. Cost is generally measured as total elapsed time for answering a query. Many factors contribute to time cost. Some of them are disk accesses, CPU, network communication etc.

## Equivalence / Transformation Rules

Two algebraic expressions are said to be equivalent if they produce same result. By using the equivalence rule which is concerned with basic relational algebra operator, we can formulate any equivalent expressions for a single query. If R, S and T are relations and C1, C2……Cn are conditions then equivalent rules are

**1. Commutativity of binary operators**

$R \cup S \equiv S \cup R$ $R \cap S \equiv S \cap R$ $R \bowtie S \equiv S \bowtie R$ $R \times S \equiv S \times R$

**2. Associativity of binary operator**

$(R \cup S) \cup T \equiv R \cup (S \cup T)$ $(R \cap S) \cap T \equiv R \cap (S \cap T)$ $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$ $R \times (S \times T) \equiv (R \times S) \times T$

**3. Commutating projection with binary operator**

$\prod_C(R \times S) \equiv \prod_A(R) \times \prod_B(S)$ where C=A U B such that attribute A is in relation R and attribute B is in relation S.

And similar for join operator also.

**4. Commutating selection with binary operator**

a. $\sigma_C(R \times S) \equiv \sigma_C(R) \times S$ , if the attribute involved in condition is from relation R

b. $\sigma_C(R \times S) \equiv R \times \sigma_C(S)$ , if the attribute involved in condition is from relation S

c. $\sigma_C(R \times S) \equiv \sigma_A(R) \times \sigma_B(S)$ , where $C=A \wedge B$ such that condition A has attribute from R and condition B has attribute from S. notes: *applying more restrictive selection first*

And similar for join operator also.

**5. Commutating selection and projection**

$\prod_X(\sigma_C(R)) \equiv \sigma_C(\prod_X(R))$ $\sigma_C(\prod_X(R)) \equiv \prod_X(\sigma_C(R))$

**6. Idempotence of unary operator**

**a. Combine Cascade Selection**

$\sigma_{C1}(\sigma_{C2}(R)) \equiv \sigma_{C1 \wedge \sigma C2}(R)$

**b. Combine Cascade Projection**

$\prod_X(\prod_Y(R)) \equiv \prod_X(R)$ if X is subset of Y.

**Example: Suppose we have the relational algebra expression as below.**

$\prod_{customer\text{-}name}(\sigma_{branch\text{-}city = 'ktm' \wedge balance > 1000}(\text{branch} \bowtie \text{account} \bowtie \text{depositer}))$

Using rule no 4a we can have equivalent expression as below

$\prod_{customer\text{-}name}((\sigma_{branch\text{-}city = 'ktm' \wedge balance > 1000}(\text{branch} \bowtie \text{account}) \bowtie \text{depositer}))$

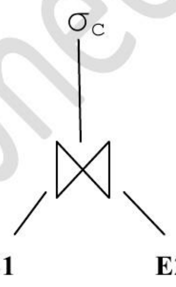
Using rule no 4c, we can have another equivalent expression as below

$\prod_{customer\text{-}name}((\sigma_{branch\text{-}city = 'ktm'}(\text{Branch}) \bowtie \sigma_{balance > 1000}(\text{account}) \bowtie \text{depositer}))$

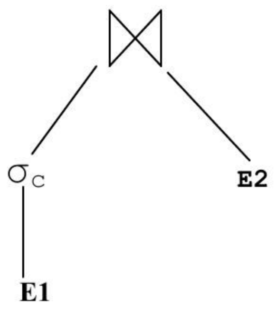
## Operator Tree

The relational algebra query can be represented graphically for simplicity by an operator tree. An operator tree is a tree in which leaf node is a relation stored in the database and a non-leaf node is a intermediate relation produce by a relational algebra operator. The sequence of operations is directed from leaves to the root, which represents the answer to the query.

*Note, Let E1 and E2 be two relation then using rule no 4*
*(if c has attribute only from E1)*



≡

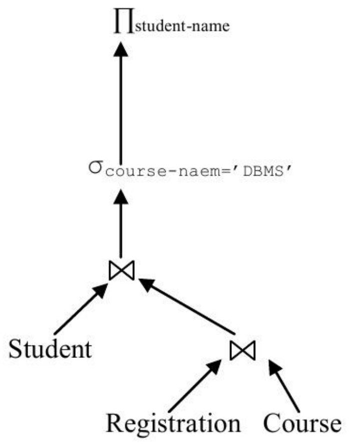


**Example:**

Suppose we have a relational algebra expression as below.

1. $\prod_{student\text{-}name}(\sigma_{course\text{-}naem='DBMS'}(\text{Student} \bowtie \text{Registration} \bowtie \text{Course}))$

The initial operator tree for the above relational expression is as below.



## Query Optimization

It is the process of selecting the most efficient query execution plan among the many strategies possible for processing a query. The query optimizer is very important component of a database system because the efficiency of the system depends on the performance of the optimizer. The selected plan minimizes the cost function.

*Query optimization refers to the process of producing a query execution plan which represents an execution strategy for the query. The selected plan minimizes an object cost function.*

Steps of optimization

1. Create an initial operator (expression) tree.
2. Move select operation down the tree for the easiest possible execution.
3. Applying more restrictive select operation first.
4. Replace Cartesian product by join.
5. Creating new projection whenever needed.
6. Adjusting rest of the tree accordingly.

Example1:

The following query retrieves the customer name from branch city pokhara whose balance is greater then 1000

**Select customer-name from Branch, Account, Depositor where city='Pkr' and balance>1000**

To process the above query, there are number of evaluation plan in which the above query can be processed.

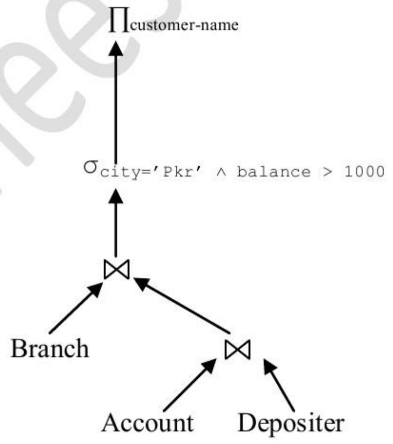
1. join relation Branch and Account, join the result with Depositor and then do the restriction.
2. join the relation Branch and Account, do the restrictions and then join the result with Depositer.
3. do the restriction, join the relations Branch and Account, join the result with Depositor.

The query optimizer estimates cost for each of the plan and choose the best way to process the query.

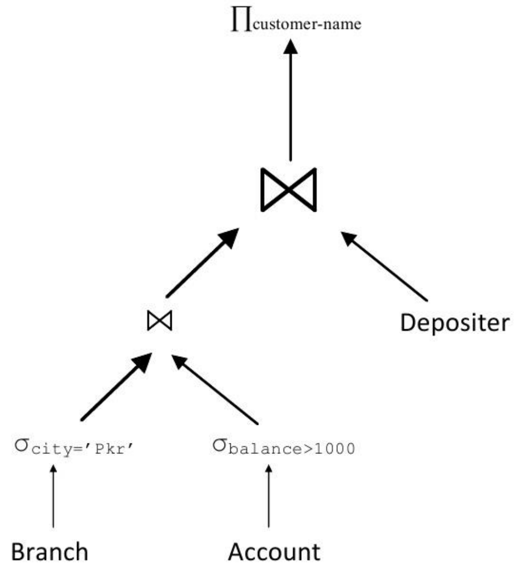
Let us consider the following algebraic expression

$\prod_{customer\text{-}name}(\sigma_{city='Pkr' \wedge balance>1000}$ (Branch X Account X Depositor))

The initial operator tree is



The final Tree after multiple Transformations is



Exmple2:

Suppose we are given the following table definitions with the certain records in each table.

PROJ (<u>PNO</u>, PNAME, BUDGET)
EMP(<u>ENO,</u> ENAME, TITLE)
ASG(<u>ENO</u>, <u>PNO</u>, DUR)

Write the sql statement and RA expression: "Find the names of employees other than Ram Thapa who worked on CAD/CAM project for either 1 or 2 years". Construct **initial operator tree** and final **efficient operator tree** after applying transformation rules.

SQL:

**select ENAME from EMP, ASG, PROJ where EMP.ENO=ASG.ENO and ASG.PNO=PROJ.PNO and ENAME!= 'Ram Thapa' and PNAME='CAD/CAM' and (DUR = 1 or DUR =2)**

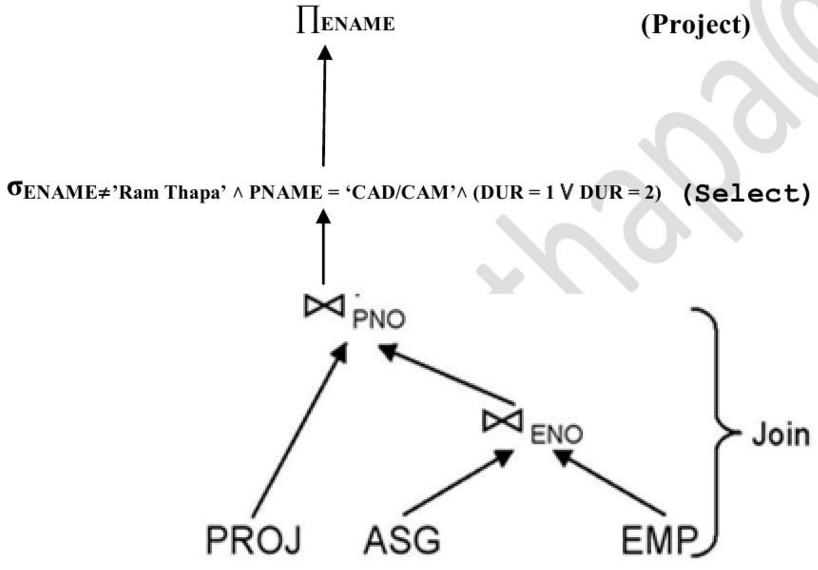
also the above SQL can be written as below

**select ENAME from EMP join ASG on EMP.ENO=ASG.ENO join PROJ on ASG.PNO=PROJ.PNO where ENAME!= 'Ram Thapa' and PNAME='CAD/CAM' and (DUR = 1 or DUR =2)**

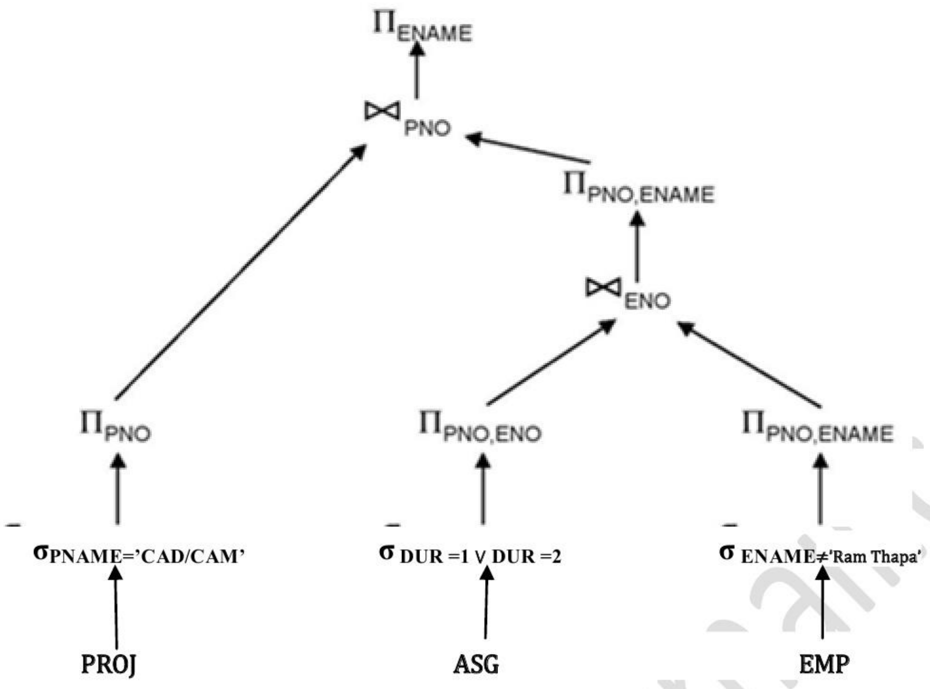
**RA:**

$$\prod_{ENAME}(\sigma_{ENAME\neq\text{'Ram Thapa'} \wedge PNAM=\text{'CAD/CAM'} \wedge (DUR=1 \vee DUR=2)}(PROJ \bowtie (EMP \bowtie ASG)))$$

**Initial Operator tree**



**Final operator tree (*a more efficient query evaluation tree, since more selective operations are performed first*)**



**Assignment**

**Show how to derive the following equivalences by a sequence of transformation using the equivalence rules.**

1. $\sigma_{\Theta 1 \wedge \Theta 2 \wedge \Theta 3}(E) = \sigma_{\Theta 1}(\sigma_{\Theta 2}(\sigma_{\Theta 3}(E))$
2. $\sigma_{\Theta 1 \wedge \Theta 2}(E1 \bowtie_{\Theta 3} E2) = \sigma_{\Theta 1(E1 \bowtie_{\Theta 3}(\sigma_{\Theta 2(E2)})))}$, where $\Theta 2$ **involves only attributes of E2.**