Chapter 6 Query Processing

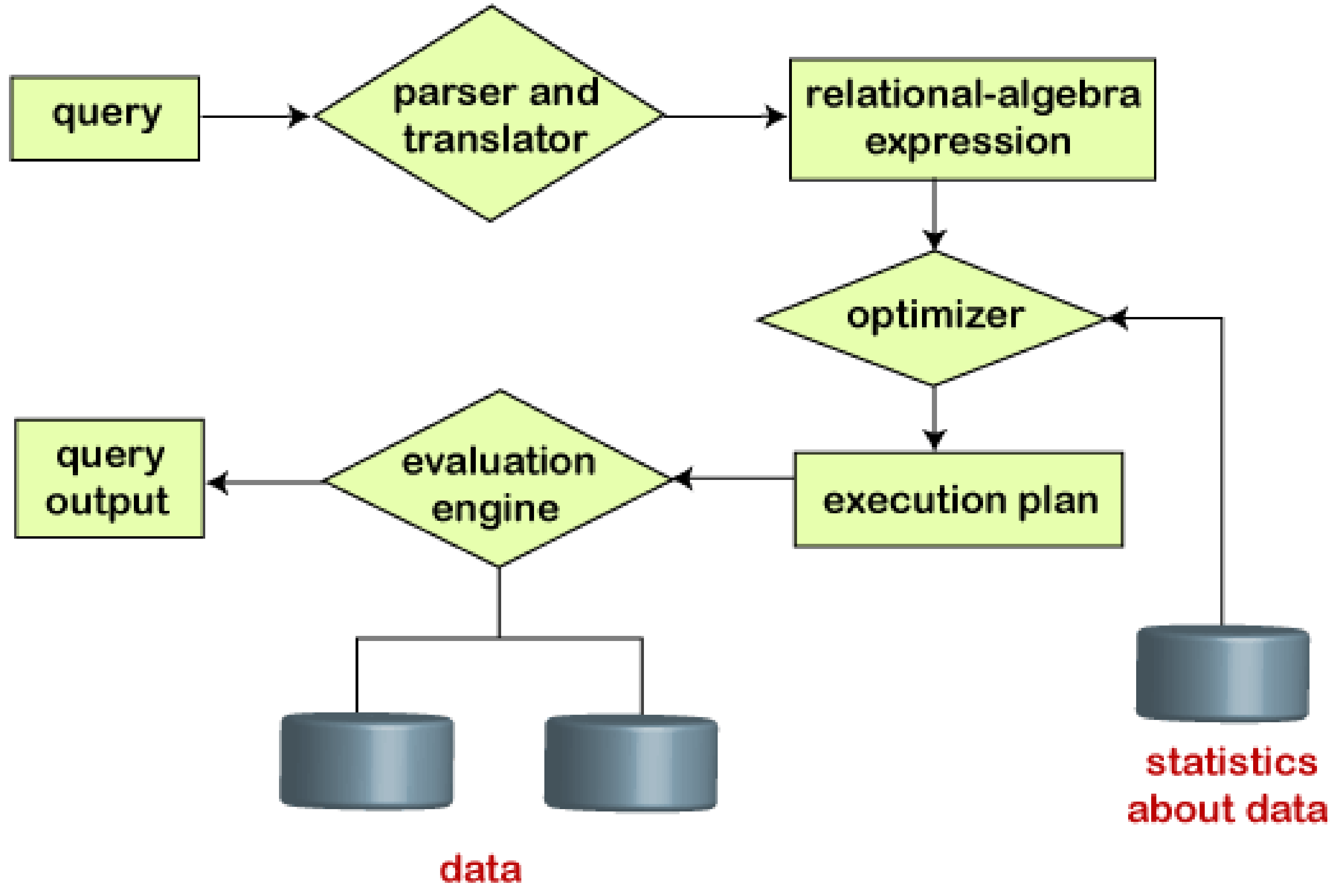
6.1 Query Processing

- In SQL, queries are expressed in high level declarative form. So the query has to be processed and optimized so that query of internal form gets a suitable execution strategy for processing.
- This optimization helps getting the result in a lesser time.
- Query processing (or Query Interpretation) is a set of activities to obtain the desired information from a database system in a predictable and reliable fashion

6.1 Query Processing

- Query Processing includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.
- In query processing, it takes various steps for fetching the data from the database. The steps involved are:
  - **Parsing and translation**
  - **Optimization**
  - **Evaluation**

6.1 Query Processing



Steps in query processing

6.1 Query Processing

1. Parsing and Translation

- ❖ Query parsing is the first step in query processing.
- ❖ When a user executes any query, for generating the internal form of the query, the parser in the system checks the syntax of the query, verifies the name of the relation in the database, the tuple, and finally the required attribute value.
- ❖ The parser creates a tree of the query, known as 'parse-tree.' Further, translate it into the form of relational algebra.

6.1 Query Processing

1. Parsing and Translation

- ❖ Translate the query into its internal form. This is then translated into relational algebra.
- ❖ Parser checks syntax, verifies relation. Then it converts it into the parse tree.
- ❖ So, a parse tree represents the query in a format that is easy to understand for DBMS.
- ❖ A parse tree is used in other steps of query processing in DBMS.

6.1 Query Processing

1. Parsing and Translation

- Suppose a user executes a query. In SQL, a user wants to fetch the records of the employees whose salary is greater than or equal to 10000. For doing this, the following query is undertaken:

  select emp_name,salary from Employee where salary>=10000;

6.1 Query Processing

1. Parsing and Translation

- Thus, to make the system understand the user query, it needs to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:
  - σsalary>=10000 (πsalary,empname (Employee))
  - Πsalary,empname (σsalary>=10000 (Employee))
- After translating the given query, we can execute each relational algebra operation by using different algorithms. So, in this way, a query processing begins its working.

6.1 Query Processing

2. Optimization

- ❖ After doing query parsing, the DBMS starts finding the most efficient way to execute the given query.
- ❖ The optimization process follows some factors for the query. These factors are indexing, joins, and other optimization mechanisms.
- ❖ These help in determining the most efficient query execution plan. So, query optimization tells the DBMS what the best execution plan is for it.
- ❖ The main goal of this step is to retrieve the required data with minimal cost in terms of resources and time.

6.1 Query Processing

2. Optimization

- ❖ The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does need not to write their query efficiently.
- ❖ Usually, a database system generates an efficient query evaluation plan, which minimizes its cost. This type of task performed by the database system and is known as **Query Optimization.**
- ❖ For optimizing a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs, and so on.

6.1 Query Processing

3. Evaluation

- For this, with addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation.
- Thus, after translating the user query, the system executes a query evaluation plan.
- The query evaluation engine takes a query evaluation plan, executes that plan and returns the answer to that query.

6.1 Query Processing

3. Evaluation

Query Evaluation Plan

- In order to fully evaluate a query, the system needs to construct a query evaluation plan.
- The annotations in the evaluation plan may refer to the algorithms to be used for the particular index or the specific operations.
- Such relational algebra with annotations is referred to as Evaluation Primitives. The evaluation primitives carry the instructions needed for the evaluation of the operation.

6.1 Query Processing

3. Evaluation

Query Evaluation Plan

- ❖ Thus, a query evaluation plan defines a sequence of primitive operations used for evaluating a query. The query evaluation plan is also referred to as the query execution plan.
- ❖ A query execution engine is responsible for generating the output of the given query. It takes the query execution plan, executes it, and finally makes the output for the user query.

6.2 Equivalence of Expression

- Two relation algebraic expression are said to be equivalent if they generate same result.
- Equivalent expression are very useful during query optimization to construct equivalent expression where different equivalent rules are used.

6.2 Equivalence of Expression

Equivalence Rules:

- ❖ The equivalence rule says that expressions of two forms are the same or equivalent because both expressions produce the same outputs on any legal database instance.
- ❖ It means that we can possibly replace the expression of the first form with that of the second form and replace the expression of the second form with an expression of the first form.
- ❖ Thus, the optimizer of the query-evaluation plan uses such an equivalence rule or method for transforming expressions into the logically equivalent one.

6.2 Equivalence of Expression

Equivalence Rules:

- The optimizer uses various equivalence rules on relational-algebra expressions for transforming the relational expressions.
- For describing each rule, we will use the following symbols:
  - **θ, θ1, θ2 … : Used for denoting the predicates.**
  - **L1, L2, L3 … : Used for denoting the list of attributes.**
  - **E, E1, E2 …. : Represents the relational-algebra expressions.**
- Let's discuss a number of equivalence rules:

6.2 Equivalence of Expression

Equivalence Rules:

1. Conjunctive selection operations can be written as a sequence of individual selections. This is called a sigma-cascade.

$$\sigma_{\theta 1 \wedge \theta 2} (E) = \sigma_{\theta 1} (\sigma_{\theta 2} (E))$$

2. Selection is commutative.

$$\sigma_{\theta 1} (\sigma_{\theta 2} (E)) = \sigma_{\theta 2} (\sigma_{\theta 1} (E))$$

6.2 Equivalence of Expression

Equivalence Rules:

3. All following projections can be omitted, only the first projection is required. This is called a pi-cascade.

$\Pi L1 (\Pi L2 (. . .(\Pi Ln (E)). . .)) = \Pi L1 (E)$

4. Selections on Cartesian Products can be re-written as Theta Joins.

i. $\sigma_\theta (E_1 \times E_2) = E_1 \,_\theta \bowtie E_2$

ii. $\sigma_{\theta 1} (E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

6.2 Equivalence of Expression

Equivalence Rules:

5. Theta Joins are commutative.

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

6. Join operations are associative.

i. Natural Join

$$(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$$

ii. Theta Join

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

In the theta associativity, θ2 involves the attributes from E2 and E3 only. There may be chances of empty conditions, and thereby it concludes that Cartesian Product is also associative.

6.2 Equivalence of Expression

Equivalence Rules:

6. Selection operation can be distributed.

When all attributes in the selection condition θ0 include only attributes of one of the expressions which are being joined.

$$\sigma_{\theta 0} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0} (E_1)) \bowtie_{\theta} E_2$$

When the selection condition θ1 involves the attributes of E1 only, and θ2 includes the attributes of E2 only.

$$\sigma_{\theta 1 \wedge \theta 2} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 1} (E_1)) \bowtie_{\theta} ((\sigma_{\theta 2} (E_2))$$

6.2 Equivalence of Expression

Equivalence Rules:

8. Projection distributes over the Theta Join.

Assume that the join condition θ includes only in L1 ᴜ L2 attributes of E1 and E2 Then, we get the following expression:

$$\Pi_{L1 \cup L2} (E_1 \bowtie_{\theta} E_2) = (\Pi_{L1} (E_1)) \bowtie_{\theta} (\Pi_{L2} (E_2))$$

Assume a join as E1 ⋈ E2. Both expressions E1 and E2 have sets of attributes as L1 and L2. Assume two attributes L3 and L4 where L3 be attributes of the expression E1, involved in the θ join condition but not in L1 ᴜ L2 Similarly, an L4 be attributes of the expression E2 involved only in the θ join condition and not in L1 ᴜ L2 attributes. Thus, we get the following expression:

$$\Pi_{L1 \cup L2} (E_1 \bowtie_{\theta} E_2) = \Pi_{L1 \cup L2} ((\Pi_{L1 \cup L3} (E_1)) \bowtie_{\theta} ((\Pi_{L2 \cup L4} (E_2)))$$

6.2 Equivalence of Expression

Equivalence Rules:

9. Union and Intersection are commutative.

i. $E_1 \cup E_2 = E_2 \cup E_1$

ii. $E_1 \cap E_2 = E_2 \cap E_1$

However, set difference operations are not commutative.

10. Union and Intersection are associative.

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

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

6.2 Equivalence of Expression

Equivalence Rules:

11. Selection operation distributes over the union, intersection, and difference operations.

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

We can similarly distribute the selection operation on υ and ∩ by replacing with -

12. Projection operation distributes over the union operation.

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

Apart from these discussed equivalence rules, there are various other equivalence rules also.

6.2 Equivalence of Expression

** Let’s do an example.*

❖ Assume the following relations:

```
instructor(Id,name,dept_name,salary)
teaches(Id,course_id,sec_id,semester,year)
course(course_id,title,dept_name,credits)
```

6.2 Equivalence of Expression

Query 1.

❖ find the name of all instructor in the music department , along with the titles of the courses that they teach.

$\pi_{\text{name,title}} (\sigma_{\text{dept\_name = “music”}} (\text{instructor} \bowtie (\text{teaches} \bowtie \pi_{\text{title,course\_id}} (\text{course}))))$

Optimized query:

$\pi_{\text{name,title}} ((\sigma_{\text{dept\_name = “music”}} (\text{instructor}) \bowtie (\text{teaches} \bowtie \pi_{\text{title,course\_id}} (\text{course}))))$

6.2 Equivalence of Expression

Query 2.

- find the name of all instructor in the music department who have taught a course in 2009, along with the titles of the courses that they taught.

$\pi_{\text{name,title}}\ (\sigma_{\text{dept\_name = "music" and year=2009}}\ (\text{instructor} \bowtie (\text{teaches} \bowtie \pi_{\text{title,course\_id}}\ (\text{course}))))$

Optimized query:

$\pi_{\text{name,title}}\ ((\sigma_{\text{dept\_name = "music"}}(\text{instructor})) \bowtie (\sigma_{\text{year=2009}}\ (\text{teaches}\ )) \bowtie \pi_{\text{title,course\_id}}\ (\text{course}))$

6.3 Query Cost Estimation

- For calculating the net estimated cost of any plan, the cost of each operation within a plan should be determined and combined to get the net estimated cost of the query evaluation plan.
- The cost estimation of a query evaluation plan is calculated in terms of various resources that include:
  - Number of disk accesses
  - Execution time taken by the CPU to execute a query
  - Communication costs in distributed or parallel database systems.

6.3 Query Cost Estimation

- To estimate the cost of a query evaluation plan, we use **the number of blocks transferred from the disk**, and **the number of disks seeks**.
- Suppose the disk has **an average block access time of ts seconds** and **takes an average of tT seconds to transfer x data blocks**.
- The block access time is the sum of disk seeks time and rotational latency (the time required to position a specific sector under the read–write head)

6.3 Query Cost Estimation

- ❖ tT – time to transfer one block
- ❖ tS – time for one to seek

❖ Cost for b block transfers plus S seeks

b * tT + S * tS

❖ If tT=0.1 ms, tS =4 ms, the block size is 4 KB, and its transfer rate is 40 MB per second. With this, we can easily calculate the estimated cost of the given query evaluation plan.

6.3 Query Cost Estimation

- The query optimizer chooses a query plan based on cost estimates. The query optimizer uses information from a number of sources to develop potential plans and determine their relative costs. These include:
  - Number of table rows
  - Column statistics, including: number of distinct values (cardinality), minimum/maximum values, distribution of values, and disk space usage
  - Access path that is likely to require fewest I/O operations, and lowest CPU, memory, and network usage
  - Available eligible projections
  - Join options: join types (merge versus hash joins), join order
  - Query predicates

6.4 Query Optimization

- A single query can be executed through different algorithms or re-written in different forms and structures.
- Hence, the question of query optimization comes into the picture – Which of these forms or pathways is the most optimal?
- The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.
- **Importance:** The goal of query optimization is to reduce the system resources required to fulfill a query, and ultimately provide the user with the correct result set faster.

6.4 Query Optimization

- First, it provides the user with faster results, which makes the application seem faster to the user.
- Secondly, it allows the system to service more queries in the same amount of time, because each request takes less time than non-optimized queries.
- Thirdly, query optimization ultimately reduces the amount of wear on the hardware (e.g. disk drives), and allows the server to run more efficiently (e.g. lower power consumption, less memory usage).

6.4 Query Optimization

- There are broadly two ways a query can be optimized:
  - **Analyze and transform equivalent relational expressions:** Try to minimize the tuple and column counts of the intermediate and final query processes.To optimize a query, we must convert the query into its equivalent form as long as an equivalence rule is satisfied.
  - **Using different algorithms for each operation:** These underlying algorithms determine how tuples are accessed from the data structures they are stored in, indexing, hashing, data retrieval and hence influence the number of disk and block accesses.

6.4 Query Optimization

Relational Algebra Query Tree

- ❖ A query tree is a tree structure which refers to a relational algebra expression.
- ❖ It denotes the input relations as leaf nodes of the tree, and denotes the relational algebra operations as internal nodes.
- ❖ An execution of the query tree contains executing an internal node operation while its operands are available and after that replacing that internal node by the relation which results from executing the operation.
- ❖ The execution terminates while the root node is executed and generates the result relation for the query.

6.4 Query Optimization

Steps for Query Optimization through Query Tree

- ❖ Decompose each selection which immediately follow the binary operation.
- ❖ Apply selection and projection before join operation that means move down the select operation into query tree.
- ❖ Replace the selection and Cartesian product with a join operation.
- ❖ Move down the project operation to minimize the number of attributes.

6.4 Query Optimization

** Let’s do an example.*

- Assume the following relations:

```
Emp(Fname,Mname,Lname,SSN,Bdate,..)
Proj(Pname,Pnumber,Plocation..)
Work_on(Essn,Pno)
```

6.4 Query Optimization

** Let’s do an example.*

Query:

SELECT LNAME FROM Emp, Works_On, Proj

Where,

PName = ‘DBMS’ AND PNumber = PNo AND ESSN = SSN AND BDate >= ’12-09-1997’

Relational Algebra:

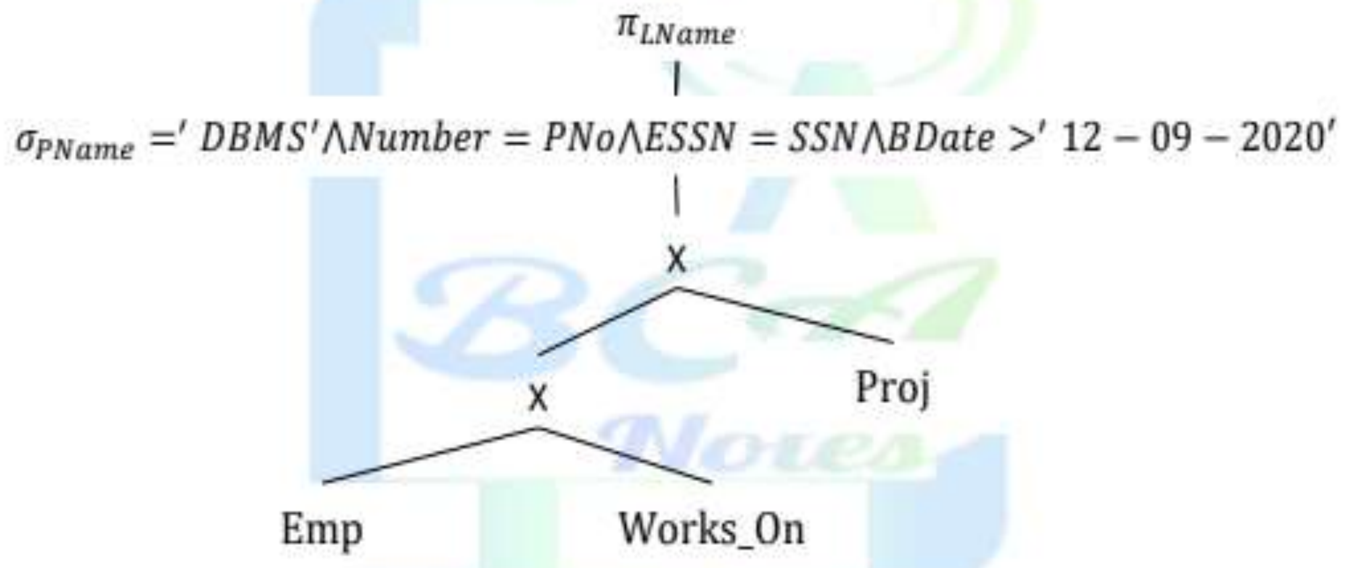
$$\pi_{\text{LName}}\left(\sigma_{\text{Name}} =' \text{DBMS}' \wedge \text{Name} = \text{PNo} \wedge \text{ESSN} = \text{SSN} \wedge \text{BData} >' 12 - 09 - 1997' (\text{Emp} \bowtie \text{Works\_On} \bowtie \text{Proj})\right)$$

6.4 Query Optimization

** Let’s do an example.*

Query Tree:

1. **The Initial Query Tree Will Be As:**

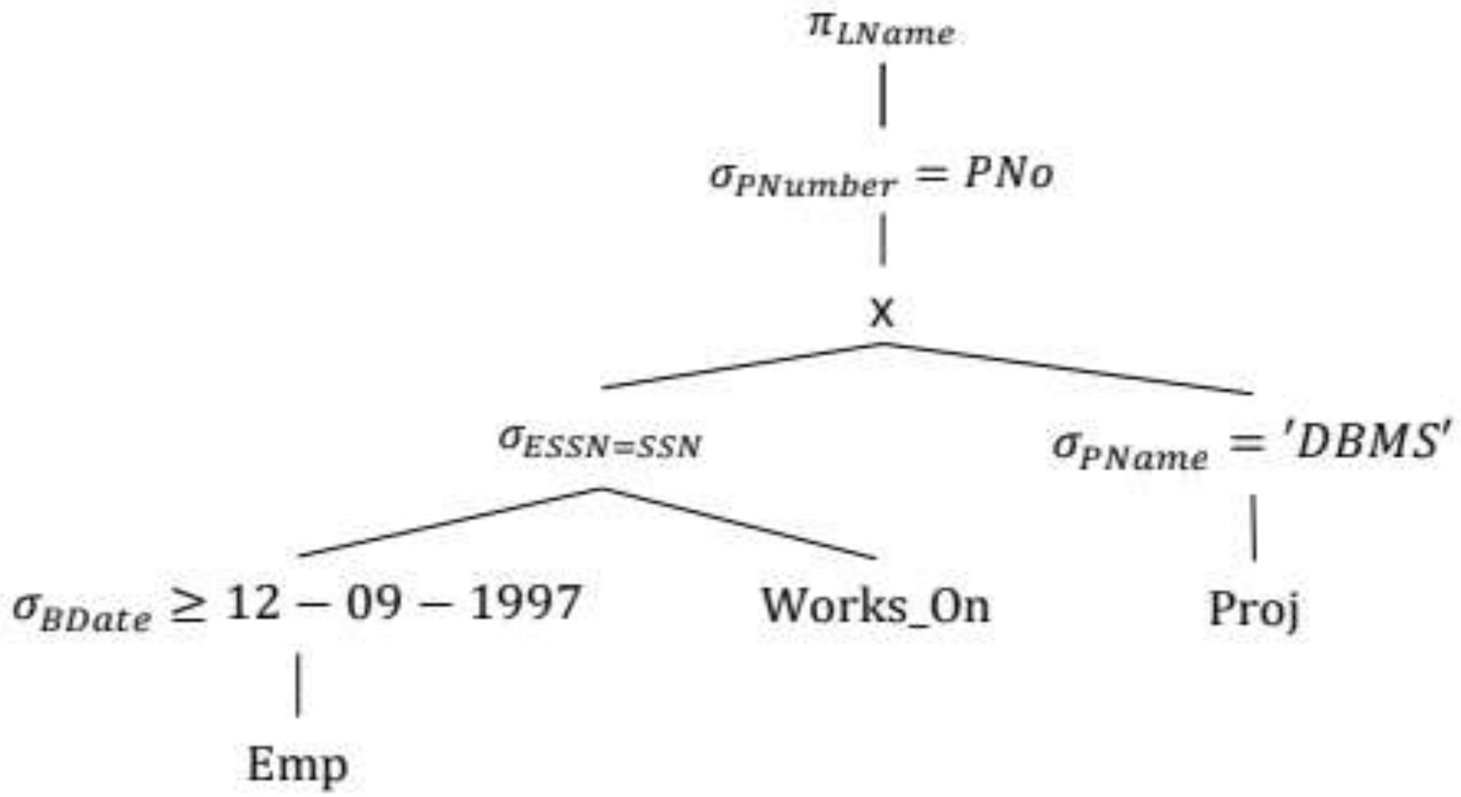


π_{LName}
σ_{PName} =' DBMS'∧Number = PNo∧ESSN = SSN∧BDate >' 12 – 09 – 2020'
X
X
Proj
Emp
Works_On

6.4 Query Optimization

** Let's do an example.*

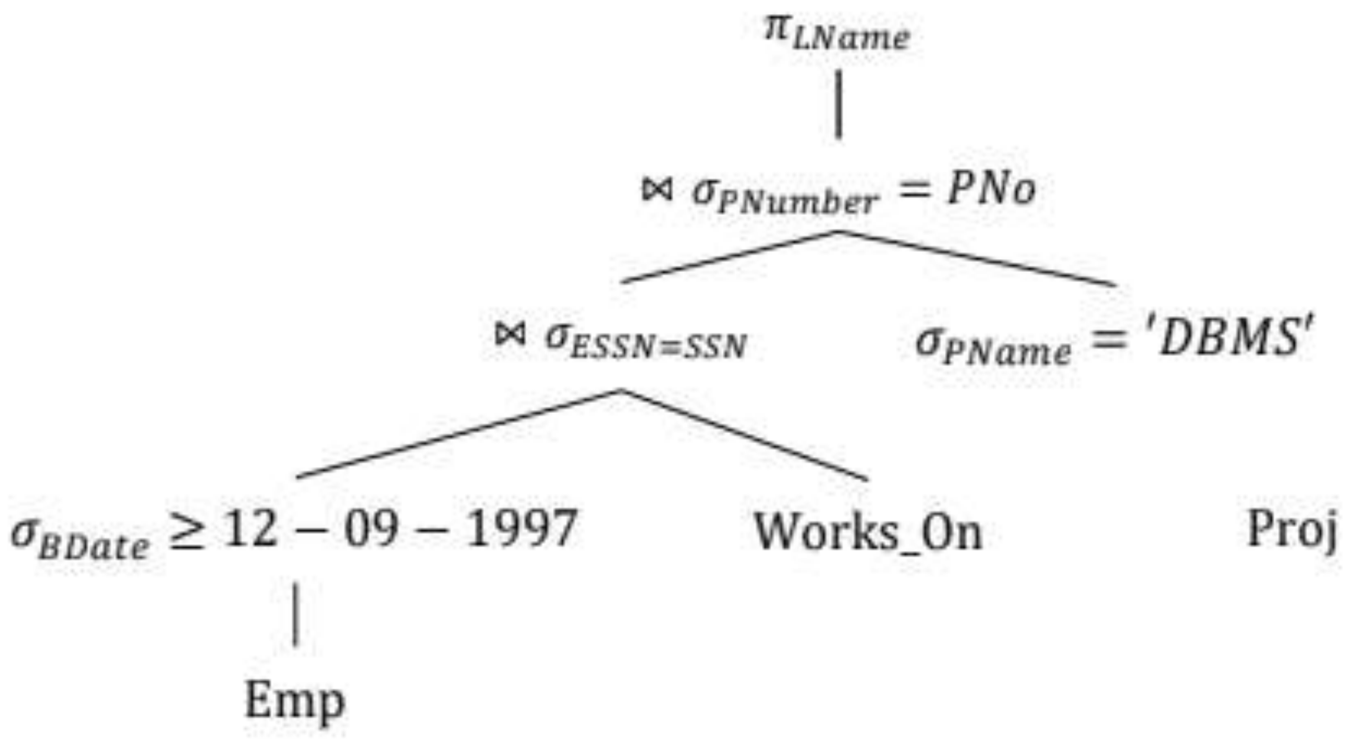
2. Move Down The Select Operation



6.4 Query Optimization

** Let's do an example.*

3. Replace The Cartesian Product And Selection With Join.


π_{LName}
$\bowtie \sigma_{PNumber} = PNo$
$\bowtie \sigma_{ESSN=SSN}$
$\sigma_{PName} = 'DBMS'$
$\sigma_{BDate} \geq 12-09-1997$
Works_On
Proj
Emp

6.4 Query Optimization

* *Let’s do an example.*

4. Moving Down The Project Operation.

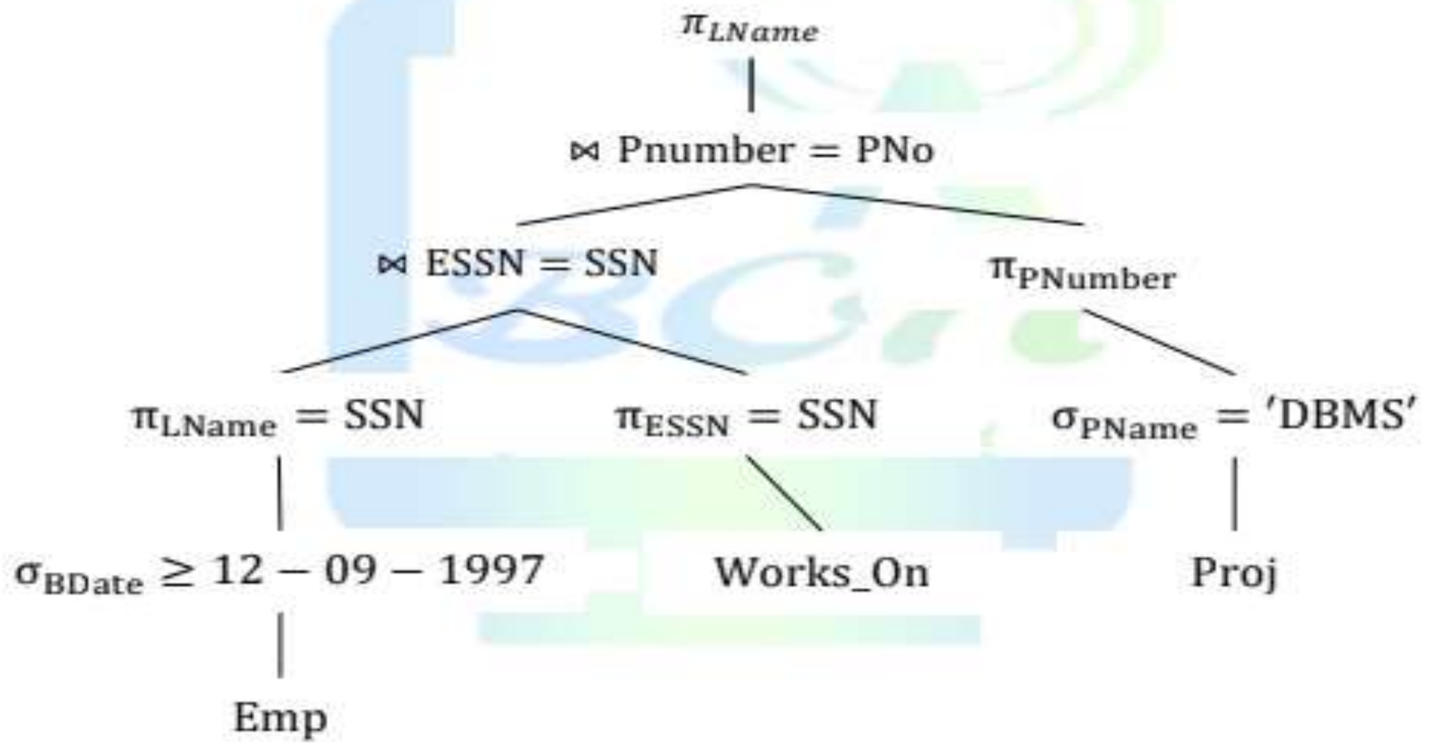


Fig: Final Optimized Query Tree

This is the end of the lecture!
I hope you enjoyed it.
Thank You