

Query Processor + optimizer

DYNAMO

6.03.2025 - 14.04.2025

Preetham Battula 22CS10015

Sumith Chandra 22CS10067

Kovvuru Kasyap 22CS10039

Jayanth Kumar 22CS30018

Pratapagiri Sathvik 22CS10039

(Git Link 📖 : <https://github.com/Snakemanp/DBMS-term.git>)

Problem Statement

The objective is to design a system capable of processing a given SQL-like query and generating its corresponding **computation tree**, which outlines the sequence of operations required for query evaluation. This involves performing **lexical** and **syntactic analysis** using tools such as **Flex** and **Bison** to parse the input query.

The system should generate a **query execution plan**, similar to those produced by database engines like **PostgreSQL**, and estimate the **cost of execution** based on factors such as the number of tuples and operation types.

The system must apply **basic query optimization techniques** inspired by those discussed in **Silberschatz's database systems** textbook to enhance performance and reduce execution cost.

Goals

1. **Perform Lexical and Syntax Analysis:** Use tools like **Flex** and **Bison** to tokenize and parse input queries, transforming them into an intermediate structured form
2. **Generate Computation Graphs:** Construct and display the corresponding **relational algebra trees** (computation graphs) that represent the logical execution of the query
3. **Apply Query Optimization Techniques:** To generate logically equivalent but more efficient query trees, apply standard **query optimization strategies**, such as **selection and projection pushdown** and **join reordering**
4. **Estimate Query Costs and Select Optimal Plan:** Compute an estimated **execution cost** for each optimized query tree, and choose the one with the **lowest cost** as the optimal execution plan

Methodology (Approach)

To implement a working SQL query processor and optimizer, we selected a simplified subset of the SQL grammar that supports the core data manipulation operations: **SELECT**, **INSERT**, **DELETE**, and **UPDATE**. These operations were chosen to ensure that our system can model the fundamental patterns of relational data access and manipulation. The grammar was constructed concerning PostgreSQL's documented grammar rules and then adapted to a more manageable subset for parsing and optimization purposes.

Lexical and Syntax Analysis

We began by designing a lexical analyzer using **Flex**, which identifies and classifies the key tokens needed for our subset of SQL. These tokens include SQL keywords (e.g., **SELECT**, **FROM**, **WHERE**), identifiers (e.g., table and column names), operators (**=**, **>**, **<**, etc.), and literals (strings, numbers).

The syntax analysis was implemented using **Bison (Yacc)**. The **parser.y** file defines the rules for parsing SQL statements according to our subset. Each rule in the grammar corresponds to a specific SQL construct (e.g., a **SELECT** clause or a **WHERE** condition). Semantic actions in Bison were used to construct **relational algebra trees** as abstract representations of SQL statements. These trees are structured as binary trees, where each node represents a relational operator (e.g., selection, projection, join, etc.), and each internal node can have at most two children. Leaf nodes correspond to base relations (tables).

For example:


- A **SELECT** with a **WHERE** clause is parsed into a σ (**selection**) operator above a π (**projection**) node, which then links to the base table.
- A **JOIN** is represented as a node with two children, each representing a participating relation.

These trees capture the logical flow of query execution and are stored in memory for subsequent optimization and evaluation.

Query Tree Optimization

The core of our system is the **query optimizer**, which operates on the relational algebra tree produced by the parser. A set of transformation functions recursively examines the original tree to identify and apply common optimization rules, such as:

- **Selection Pushdown:** Moves selection operators as close to the base tables as possible to reduce intermediate result sizes early in the execution.
- **Projection Pushdown:** Similar to selection pushdown, but for projection operations—removes unused columns at earlier stages.
- **Join Reordering:** Applies associativity and commutativity rules to change the order of joins for potentially lower-cost plans.



Each transformation generates a **new equivalent tree**, and these are stored in a container (e.g., an array). To prevent exponential growth in the number of trees, we use a **pruning strategy**: we cap the number of generated trees at 100. Once this limit is reached, further equivalent plans are discarded.

Cost Estimation and Plan Selection

To choose the most efficient plan, each tree is evaluated using a **cost estimation function**, which considers factors such as:

- Estimated number of rows from intermediate operations
- Cost of performing selections, projections, and joins
- Tree height or operator depth (as a proxy for complexity)

These cost functions are simplistic and heuristic-driven but provide a sufficient basis for comparing query plans. After all trees are evaluated, the one with the **lowest estimated cost** is selected as the optimal plan for execution.

Metadata

- This schema is read from `documentation.txt` at startup.
- The program parses the file and **stores the metadata in local structures** (like maps or structs).
- The metadata is used for:
 - **Verifying column names and table references**
 - **Choosing optimal join orders** based on known relationships
 - **Projection pushdown** based on used attributes
 - **Validation of SQL query correctness**

1. Table: `instructor` (ID , name , dept_name , salary)

2. Table: `teaches` (ID , course_id , sec_id , semester , year)

3. Table: `course` (course_id , title , dept_name , credits)

Results

Projection pushdown demonstration -

Sample sql query -

```
SELECT instructor.ID, course.course_id
FROM instructor
JOIN course ON instructor.dept_name = course.dept_name
WHERE instructor.salary > 50000;
```

Original and pre-optimized trees

```
Original Relational Algebra Tree for sql statement 7:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 50000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    Relation: instructor
    Relation: course
Cost of actual tree: 9295
```

```
Tree after Pre-Transformations:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 50000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    Relation: instructor
    Relation: course
```

Multiple trees after various optimizations -

```
Transformed Tree 1:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
  π-Project ()
    Attribute: instructor.ID
    Attribute: instructor.dept_name
  σ-Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 50000
  π-Project ()
    Attribute: instructor.salary
    Relation: instructor
  π-Project ()
    Attribute: course.course_id
    Attribute: course.dept_name
    Relation: course
Cost of transformed tree 1: 2174
```

```
Transformed Tree 2:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 50000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    Relation: instructor
    Relation: course
Cost of transformed tree 2: 9295
```

```

Transformed Tree 3:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    σ-Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 50000
      Relation: instructor
      Relation: course
Cost of transformed tree 3: 4363

```

```

Transformed Tree 4:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    π-Project ()
      Attribute: instructor.ID
      Attribute: instructor.dept_name
    σ-Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 50000
      Relation: instructor
    π-Project ()
      Attribute: course.course_id
      Attribute: course.dept_name
      Relation: course
Cost of transformed tree 4: 2306

```

Final Optimized tree with least cost -

```

Optimized Tree:
π-Project ()
  Attribute: course.course_id
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.dept_name
        Attribute: course.dept_name
    π-Project ()
      Attribute: instructor.ID
      Attribute: instructor.dept_name
    σ-Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 50000
      π-Project ()
        Attribute: instructor.salary
        Relation: instructor
    π-Project ()
      Attribute: course.course_id
      Attribute: course.dept_name
      Relation: course
Cost of Optimized tree 1: 2174

```


Selection pushdown demonstration -

Sample sql query -

```
SELECT *
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
WHERE instructor.dept_name = 'Physics' AND teaches.semester = 'Fall';
```

Original and pre-optimized trees -

```
Original Relational Algebra Tree for sql statement 8:
π-Project ()
  Attribute: *
  σ-Select[WHERE]
    And: AND
      Comparison: =
        Attribute: instructor.dept_name
        Literal: 'Physics'
      Comparison: =
        Attribute: teaches.semester
        Literal: 'Fall'
    ⋈-join
      Condition:
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
      Relation: instructor
      Relation: teaches
Cost of actual tree: 8122
```

```
Tree after Pre-Transformations:
σ-Select[WHERE]
  And: AND
    Comparison: =
      Attribute: instructor.dept_name
      Literal: 'Physics'
    Comparison: =
      Attribute: teaches.semester
      Literal: 'Fall'
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
    Relation: instructor
    Relation: teaches
```

Multiple trees after various optimizations -

```
Transformed Tree 1:
⋈-join
  Condition:
    Comparison: =
      Attribute: instructor.ID
      Attribute: teaches.ID
  σ-Select[]
    Comparison: =
      Attribute: instructor.dept_name
      Literal: 'Physics'
    Relation: instructor
  σ-Select[]
    Comparison: =
      Attribute: teaches.semester
      Literal: 'Fall'
    Relation: teaches
Cost of transformed tree 1: 556
```

```
Transformed Tree 2:
σ-Select[WHERE]
  And: AND
    Comparison: =
      Attribute: instructor.dept_name
      Literal: 'Physics'
    Comparison: =
      Attribute: teaches.semester
      Literal: 'Fall'
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
    Relation: instructor
    Relation: teaches
Cost of transformed tree 2: 8122
```

Final Optimized tree with least cost -

```
Optimized Tree:
⋈-join
  Condition:
    Comparison: =
      Attribute: instructor.ID
      Attribute: teaches.ID
  σ-Select[]
    Comparison: =
      Attribute: instructor.dept_name
      Literal: 'Physics'
    Relation: instructor
  σ-Select[]
    Comparison: =
      Attribute: teaches.semester
      Literal: 'Fall'
    Relation: teaches

Cost of Optimized tree 1: 556
```

Join Reordering demonstration -

Sample sql query -

```
SELECT instructor.ID
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
JOIN course ON teaches.course_id = course.course_id
WHERE instructor.salary > 60000;
```

Original and pre-optimized trees -

```
Original Relational Algebra Tree for sql statement 9:
π-Project ()
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
      Relation: instructor
      Relation: teaches
      Relation: course

Cost of actual tree: 26426
```

```
Tree after Pre-Transformations:
π-Project ()
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
      Relation: instructor
      Relation: teaches
      Relation: course
```


Multiple trees after various optimizations -

```
Transformed Tree 1:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
    π-Project ()
      Attribute: instructor.ID
      Attribute: teaches.course_id
    ⋈-join
      Condition:
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
      π-Project ()
        Attribute: instructor.ID
        σ-Select[]
          Comparison: >
            Attribute: instructor.salary
            Literal: 60000
      π-Project ()
        Attribute: instructor.salary
        Relation: instructor
    π-Project ()
      Attribute: teaches.course_id
      Attribute: teaches.ID
      Relation: teaches
    π-Project ()
      Attribute: course.course_id
      Relation: course

Cost of transformed tree 1: 6453419
```

```
Transformed Tree 2:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
    σ-Select[]
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
    π-Project ()
      Attribute: teaches.course_id
      Attribute: course.course_id
      Attribute: teaches.ID
    ⋈-join
      π-Project ()
        Attribute: course.course_id
        Relation: course
      π-Project ()
        Attribute: teaches.course_id
        Attribute: teaches.ID
        Relation: teaches
    π-Project ()
      Attribute: instructor.ID
      σ-Select[]
        Comparison: >
          Attribute: instructor.salary
          Literal: 60000
      π-Project ()
        Attribute: instructor.salary
        Relation: instructor

Cost of transformed tree 2: 56566326
```

```
Transformed Tree 3:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
    π-Project ()
      Attribute: instructor.ID
      Attribute: course.course_id
    ⋈-join
      π-Project ()
        Attribute: instructor.ID
        σ-Select[]
          Comparison: >
            Attribute: instructor.salary
            Literal: 60000
      π-Project ()
        Attribute: instructor.salary
        Relation: instructor
    π-Project ()
      Attribute: course.course_id
      Relation: course
    π-Project ()
      Attribute: teaches.course_id
      Attribute: teaches.ID
      Relation: teaches

Cost of transformed tree 3: 36553279
```

```
Transformed Tree 4:
π-Project ()
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
  ⋈-join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
    ⋈-join
      Condition:
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
        Relation: instructor
        Relation: teaches
      Relation: course

Cost of transformed tree 4: 26426
```

```

Transformed Tree 5:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
     $\sigma$ -Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 60000
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
      Relation: instructor
      Relation: teaches
  Relation: course

```

Cost of transformed tree 5: 17197

```

Transformed Tree 6:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: teaches.course_id
   $\sigma$ -Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
      Relation: instructor
      Relation: teaches
   $\pi$ -Project ()
    Attribute: course.course_id
    Relation: course

```

Cost of transformed tree 6: 10089

```

Transformed Tree 7:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: teaches.course_id
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
   $\sigma$ -Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor
    Relation: teaches
   $\pi$ -Project ()
    Attribute: course.course_id
    Relation: course

```

Cost of transformed tree 7: 6049

```

Transformed Tree 8:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: teaches.course_id
   $\bowtie$ -join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
   $\pi$ -Project ()
    Attribute: instructor.ID
   $\sigma$ -Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor
   $\pi$ -Project ()
    Attribute: teaches.course_id
    Attribute: teaches.ID
    Relation: teaches
   $\pi$ -Project ()
    Attribute: course.course_id
    Relation: course

```

Cost of transformed tree 8: 6453551

```

Transformed Tree 9:
π-Project ()
  Attribute: instructor.ID
  σ-Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
  ⋈-join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
  ⋈-join
    Relation: course
    Relation: teaches
    Relation: instructor

```

Cost of transformed tree 9: 82002641349

```

Transformed Tree 10:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
  ⋈-join
    Relation: course
    Relation: teaches
  σ-Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor

```

Cost of transformed tree 10: 329070015

```

Transformed Tree 11:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
  π-Project ()
    Attribute: teaches.course_id
    Attribute: course.course_id
    Attribute: teaches.ID
  ⋈-join
    Relation: course
    Relation: teaches
  π-Project ()
    Attribute: instructor.ID
  σ-Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor

```

Cost of transformed tree 11: 107958854

```

Transformed Tree 12:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
  σ-Select[]
    Comparison: =
      Attribute: teaches.course_id
      Attribute: course.course_id
  π-Project ()
    Attribute: teaches.course_id
    Attribute: course.course_id
    Attribute: teaches.ID
  ⋈-join
    Relation: course
    Relation: teaches
  π-Project ()
    Attribute: instructor.ID
  σ-Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor

```

Cost of transformed tree 12: 56583215

```

Transformed Tree 13:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: instructor.ID
        Attribute: teaches.ID
  σ-Select[]
    Comparison: =
      Attribute: teaches.course_id
      Attribute: course.course_id
  π-Project ()
    Attribute: teaches.course_id
    Attribute: course.course_id
    Attribute: teaches.ID
  ⋈-join
    π-Project ()
      Attribute: course.course_id
      Relation: course
    π-Project ()
      Attribute: teaches.course_id
      Attribute: teaches.ID
      Relation: teaches
  π-Project ()
    Attribute: instructor.ID
  σ-Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
    Relation: instructor

```

Cost of transformed tree 13: 56566458

```

Transformed Tree 14:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\sigma$ -Select[WHERE]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
   $\bowtie$ -join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
   $\bowtie$ -join
    Relation: instructor
    Relation: course
    Relation: teaches

```

Cost of transformed tree 14: 81972175073

```

Transformed Tree 15:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
   $\sigma$ -Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
   $\bowtie$ -join
    Relation: instructor
    Relation: course
    Relation: teaches

```

Cost of transformed tree 15: 159758480

```

Transformed Tree 16:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: course.course_id
   $\sigma$ -Select[]
    Comparison: >
      Attribute: instructor.salary
      Literal: 60000
   $\bowtie$ -join
    Relation: instructor
    Relation: course
   $\pi$ -Project ()
    Attribute: teaches.course_id
    Attribute: teaches.ID
    Relation: teaches

```

Cost of transformed tree 16: 43560426

```

Transformed Tree 17:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: course.course_id
   $\bowtie$ -join
     $\sigma$ -Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 60000
      Relation: instructor
      Relation: course
   $\pi$ -Project ()
    Attribute: teaches.course_id
    Attribute: teaches.ID
    Relation: teaches

```

Cost of transformed tree 17: 36556494

```

Transformed Tree 18:
 $\pi$ -Project ()
  Attribute: instructor.ID
   $\bowtie$ -join
    Condition:
      And: AND
        Comparison: =
          Attribute: teaches.course_id
          Attribute: course.course_id
        Comparison: =
          Attribute: instructor.ID
          Attribute: teaches.ID
   $\pi$ -Project ()
    Attribute: instructor.ID
    Attribute: course.course_id
   $\bowtie$ -join
     $\pi$ -Project ()
      Attribute: instructor.ID
     $\sigma$ -Select[]
      Comparison: >
        Attribute: instructor.salary
        Literal: 60000
      Relation: instructor
   $\pi$ -Project ()
    Attribute: course.course_id
    Relation: course
   $\pi$ -Project ()
    Attribute: teaches.course_id
    Attribute: teaches.ID
    Relation: teaches

```

Cost of transformed tree 18: 36553411

Final Optimized tree with least cost

```

Optimized Tree:
π-Project ()
  Attribute: instructor.ID
  ⋈-join
    Condition:
      Comparison: =
        Attribute: teaches.course_id
        Attribute: course.course_id
    π-Project ()
      Attribute: instructor.ID
      Attribute: teaches.course_id
      ⋈-join
        Condition:
          Comparison: =
            Attribute: instructor.ID
            Attribute: teaches.ID
        σ-Select[]
          Comparison: >
            Attribute: instructor.salary
            Literal: 60000
          Relation: instructor
          Relation: teaches
    π-Project ()
      Attribute: course.course_id
      Relation: course

Cost of Optimized tree 7: 6049

```

Assumptions

1. Limited Clause Support

The optimizer **does not support** the following SQL clauses:

- UNION
- INTERSECT
- GROUP BY
- HAVING

The parser works correctly and parses them but the optimizer does not work and gives false results (scope checker won't work for them).

2. Default cost of Projection to be Zero (As sql SELECT don't do any duplicate elimination)

The optimizer assumes the projection cost to be zero. If we want to consider the duplicate elimination case then we can use `SELECT DISTINCT`.

3. Index free Intermediate Results

The intermediate results formed from the tables do not have indices. The optimizer assumes the indices are not forwarded from the table to upward levels and calculator costs.

4. Cost of write operations

The optimizer assumes the cost of writing the block backs (In queries like `INSERT`, `UPDATE`, `DELETE`) as twice the no of block transfer time

5. Bias towards selection and projection

When The resultant equivalent trees formed over shoots the Max resultant trees the optimiser has bias towards projection and selection pushdowns and assumes the cost is reduced on these operations and continues optimizing without checking the validity.

(The Max Intermediate Trees can be changed by changing the defined value `MAX_RES_TREES` defined in `sql_parser.h` Default set to 25.)

Conclusion and Future Scope

In this project, we successfully developed a simplified **SQL Query Processor and Optimizer** capable of parsing, analyzing, and optimizing basic SQL queries involving `SELECT`, `INSERT`, `DELETE`, and `UPDATE` operations. By leveraging **Flex** for lexical analysis and **Bison** for syntax parsing, we translated SQL queries into **relational algebra trees**, enabling us to visualize and manipulate the logical structure of each query.

Our system applies fundamental **query optimization techniques**—including selection and projection pushdown, as well as join reordering—to generate multiple equivalent query trees. To manage computational complexity, we implemented a pruning mechanism to limit the number of generated trees. Each tree is evaluated using a cost estimation model based on relational algebra operation heuristics, and the plan with the lowest estimated cost is selected as the optimal query plan.

Additionally, the use of **external metadata** ensures accurate query validation and improves optimization quality by allowing schema-aware transformations.

Future Scope

While the current system achieves the foundational goals of query parsing and optimization, there are several avenues for future enhancement:

- **Support for Advanced SQL Features:** Extend the grammar to support more complex SQL clauses and aggregate functions.
- **Improved Cost Estimation Model:** Replace the heuristic-based cost model with a more accurate model that uses real table statistics (e.g., histograms, indexes, data distributions).
- **Dynamic Query Plan Generation:** Integrate runtime statistics or feedback-based optimization to adjust query plans dynamically during execution.
- **Integration with Real Databases:** Interface the optimizer with a real database system to execute and benchmark generated plans.
- **Graphical User Interface (GUI):** Develop a simple UI to visualize relational algebra trees and track how optimizations alter the query plan structure.
- **Query Caching and Reuse:** Implement a query cache to store and reuse previously optimized query plans for similar query patterns.

By expanding in these directions, this project can evolve into a robust educational or research-oriented SQL query processing engine.

References

1. The grammar used is based on [PostgreSQL Documentation](#).
2. Optimizations and cost optimizations
 - a. [Silberschatz's database systems](#)
 - b. [CS-30202 DBMS spring2k25](#)
3. Lexical Analysis and Parsing [CS39003-Compilers theory and course](#)



Thank You