CS39202 - Database Management System

Proposal for Term Project - Spring 2023

Team Name: Outer Rim

Team Members : Ashwani Kumar Kamal, Hardik Pravin Soni, Shiladitya De, Sourabh Soumyakanta Das, Archit Mangrulkar

Project Name: Rule based query rewriting

Problem Description: Specify some fixed rules for query optimization. Write a query rewriter for simple queries which takes as input a relational algebra query and returns its optimal version.

Proposal:

- ### Introduction:

  Query optimization is a critical step in relational database management systems (RDBMS). The process involves identifying the most efficient way to execute a query and optimizing it to minimize execution time and resource usage. In this proposal, we outline a project to specify fixed rules for query optimization and develop a query rewriter for simple queries. The project's objective is to provide a tool that takes as input a relational algebra query and returns its optimal version by applying the specified query optimization rules.

- ### Project Scope and Objectives

  The project's scope is to specify fixed rules for query optimization and develop a query rewriter for simple queries. The project will involve the following objectives:

  1. Specification of fixed rules for query optimization.
  2. Design and implementation of a query rewriter tool.
  3. Testing the query rewriter on a suitable testset.
  4. Documentation of the design, implementation, and representing the results in a comprehensive report.

- ### Project Deliverables

  The project will deliver the following:

  1. Fixed rules for query optimization.
  2. A query rewriter tool that takes as input a relational algebra query and returns its optimal version by applying the query optimization rules.
  3. A report documenting the design, implementation, and benchmarking results of the query rewriter.

- **Methodology:**
    - **Specifying syntax for giving relational algebra query:**

      The first task is to define a proper syntax to give relational algebra query input by the user. The syntax finalized by us is as follows:

        - σ == `` `SELECT` ``
        - Π == `` `PROJECT` ``
        - ⋈ == `` `JOIN` ``
        - × == `` `PROD` ``
        - $\sigma_{(a \wedge b)}(T)$ == `` `SELECT[a AND b](T)` `` etc.
    - **Some rules to be used for query optimization:**

      We are considering for now only those rules which are heuristic based and not based on the actual table statistics (cost based ones). If time permits then we will try to incorporate those as well.

        - $\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$
        - $\pi_{L_1}(\pi_{L_2}(\ldots(\pi_{L_n}(E))\ldots)) = \pi_{L_1}(E)$
        - $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
        - $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$
        - $\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$
        - $\pi_L(E_1 \cup E_2) = (\pi_L(E_1)) \cup (\pi_L(E_2))$

    - **Approach towards solving the problem:**

      Our approach to solve the problem by using a compiler based approach. In this approach we are going to first parse the input query using a lexer like *flex* and then we are going to do a bottom up parsing of it using tools like *yacc* or *bison*.

      For doing the above we need to specify a grammar which will be written by us keeping in mind the syntax of the queries. Once the parsing is done we will get a tree which we will go on flattening by optimizing the queries by writing corresponding rules.

    - **Testing:**

We are going to test the program on a standard set of queries and report the same in the report.

- **References**

1. Chaudhuri, S. (1998). An overview of query optimization in relational systems. ACM SIGMOD Record, 27(2), 34-49.
2. Graefe, G. (1995). Volcano-an extensible and parallel query evaluation system. IEEE Transactions on Knowledge and Data Engineering, 7(4), 563-575.
3. Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979). Access path selection in a relational database management system. In Proceedings

Proposal for Query Optimization Rules and Query Rewriter

Introduction

Query optimization is a critical step in relational database management systems (RDBMS). The process involves identifying the most efficient way to execute a query and optimizing it to minimize execution time and resource usage. In this proposal, we outline a project to specify fixed rules for query optimization and develop a query rewriter for simple queries. The project's objective is to provide a tool that takes as input a relational algebra query and returns its optimal version by applying the specified query optimization rules.

Project Scope and Objectives

The project's scope is to specify fixed rules for query optimization and develop a query rewriter for simple queries. The project will involve the following objectives:

5. Specification of fixed rules for query optimization.
6. Design and implementation of a query rewriter tool.
7. Integration of the query rewriter with a relational algebra parser to process input queries.
8. Testing and benchmarking the query rewriter on a suitable testset.
9. Documentation of the design, implementation, and benchmarking results in a comprehensive report.

Project Deliverables

The project will deliver the following:

4. Fixed rules for query optimization.
5. A query rewriter tool that takes as input a relational algebra query and returns its optimal version by applying the query optimization rules.
6. Integration of the query rewriter with a relational algebra parser.
7. A report documenting the design, implementation, and benchmarking results of the query rewriter.
8. A comparison of the performance of the query rewriter with existing query optimization tools.

Methodology

The project will follow a phased approach for the specification of fixed rules for query optimization and the development of the query rewriter. The following steps will be followed:

1. Research: The project team will conduct research on existing query optimization techniques and best practices for developing a query rewriter.
2. Specification of fixed rules: We will specify fixed rules for query optimization based on the research findings.
3. Design and implementation: The project team will design and implement a query rewriter tool that takes as input a relational algebra query and returns its optimal version by applying the query optimization rules.
4. Integration: The project team will integrate the query rewriter with a relational algebra parser to process input queries.
5. Testing and benchmarking: The project team will test and benchmark the query rewriter on a suitable dataset to evaluate its performance.
6. Report: The project team will document the project's design, implementation, and benchmarking results in a comprehensive report.

Conclusion

In conclusion, the proposed project to specify fixed rules for query optimization and develop a query rewriter for simple queries is a critical step towards improving query processing and optimizing resource usage in RDBMS. The project will follow a phased approach, including research, specification of fixed rules, design and implementation, integration, testing and benchmarking, and report documentation. The project will deliver a set of fixed rules for query optimization, a query rewriter tool, and a report documenting the design, implementation, and benchmarking results. The project will contribute to the advancement of computer science research and serve as a reference for similar projects worldwide.

References

4. Chaudhuri, S. (1998). An overview of query optimization in relational systems. ACM SIGMOD Record, 27(2), 34-49.
5. Graefe, G. (1995). Volcano-an extensible and parallel query evaluation system. IEEE Transactions on Knowledge and Data Engineering, 7(4), 563-575.
6. Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979). Access path selection in a relational database management system. In Proceedings