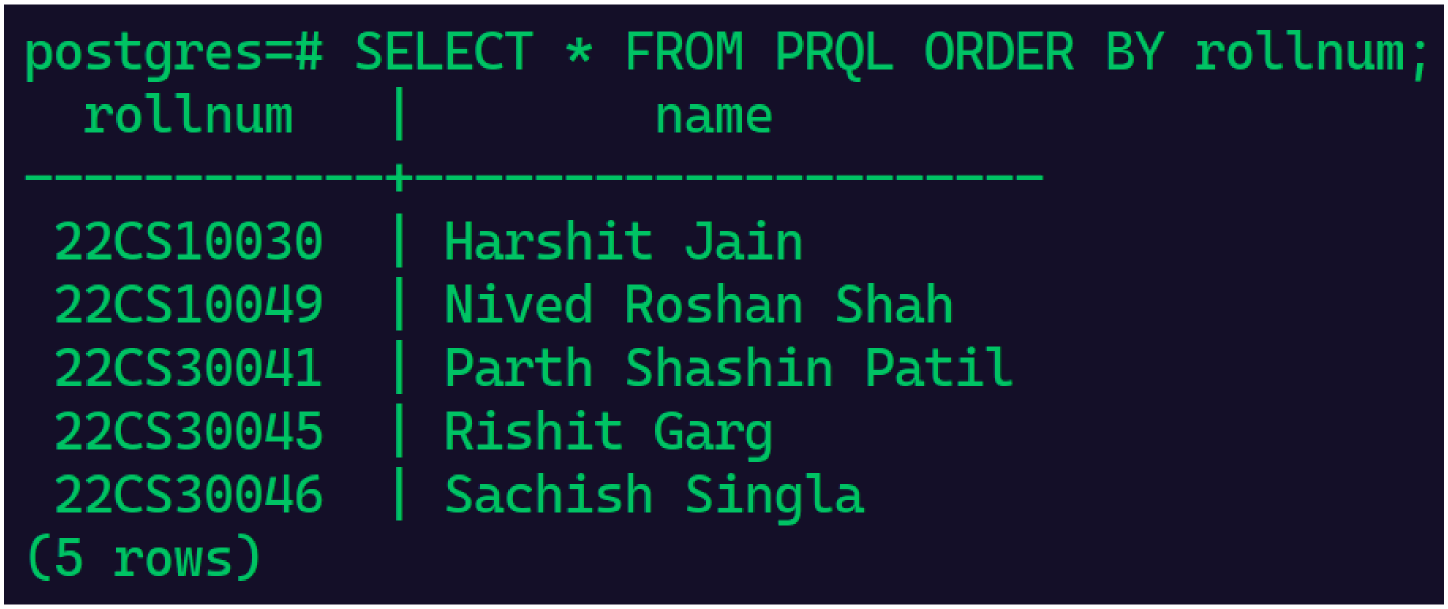
DBMS Term Project Proposal

Query Processing & Optimization

Team



```
postgres=# SELECT * FROM PRQL ORDER BY rollnum;
  rollnum    |         name
------------+---------------------
 22CS10030  | Harshit Jain
 22CS10049  | Nived Roshan Shah
 22CS30041  | Parth Shashin Patil
 22CS30045  | Rishit Garg
 22CS30046  | Sachish Singla
(5 rows)
```

Abstract

This project develops an SQL query optimization system to enhance database performance through parsing, cost-based optimization, and relational algebra transformations. The implementation focuses on a selected subset of simple SQL statements to create a proof-of-concept for the practical application of theoretical database optimization principles.

The system architecture consists of three primary components:

1. **SQL Parser and Analyzer**: A parser that transforms raw SQL statements into a structured internal representation suitable for optimization analysis.
2. **Cost-Based Optimizer**: An optimization engine that utilizes PostgreSQL's native statistics collection mechanisms to make data-driven decisions about query execution strategies. This component implements optimization techniques such as:
    - Join operation reordering based on cardinality estimates
    - WHERE clause predicate reorganization by selectivity
    - Automatic transformation of suboptimal constructs (such as IN clauses to EXISTS) based on data distribution patterns
3. **Relational Algebra Transformation Engine**: A component that converts SQL queries into their corresponding relational algebra expressions and applies transformation rules derived from Silberschatz's foundational database systems book. These transformations include selective predicate pushdown, elimination of redundant projection operations, and strategic decomposition of complex select statements.

Motivation

Database query performance remains a critical challenge in data-intensive applications. As data volumes grow, inefficient queries can lead to significant performance bottlenecks and poor user experience. This project addresses this gap by creating a custom query optimization layer that leverages database statistics to make cost-aware optimization decisions and implements optimization techniques from established literature.

Implementation Plan

Week 1: SQL Parsing and Query Representation

- Design and implement a parser for the target subset of SQL statements
- Create internal data structures to represent parsed queries

Week 2: Cost-Based Optimization

- Extract table and column statistics from PostgreSQL
- Implement join reordering algorithms and WHERE clause condition reordering
- Transform IN clauses to EXISTS where beneficial based on statistics

Week 3: Relational Algebra Transformation

- Convert optimized SQL queries to relational algebra expressions
- Implement algebraic optimization rules from the Silberschatz textbook
- Apply techniques for pushing down SELECT operations and removing redundant projections

Week 4: Integration, Testing, and Evaluation

- Integrate all components into a cohesive system
- Develop test suite and evaluate performance improvements
- Document optimization techniques and their impact

Technical Requirements

- Programming language: Python, C++
- PostgreSQL database for statistics collection and query execution
- Access to system performance monitoring tools

Expected Outcomes

1. A functional query optimization system capable of parsing and optimizing SQL queries
2. Interface for entering unoptimized queries and receiving results of the performed optimizations
3. Insights into the effects of optimization strategies utilized