

**CS 632 PROJECT**

**Exploiting Uniqueness in Query Optimization**

**Vinit Deodhar**

**Ajay Gupta**

**Supervisor : Prof. S. Sudarshan**

**4th May, 2013**

1. **Introduction**

The project aims to remove the DISTINCT clause if not required. The DISTINCT clause is used to invoke duplicate elimination on the result. For this, the database engine needs to perform sorting of the result data after which a scan of the result is performed to eliminate duplicate entries. But what if the result obtained before the DISTINCT clause is processed already has only a unique value for each tuple. We can avoid the sort operator and the scan of the result.

1. **Exploiting Uniqueness**

Consider the following 3 relations

SUPPLIER(SNO,SNAME, SCITY, BUDGET, STATUS)

PARTS (SNO, PNO, PNAME, COLOR)

AGENTS (SNO, ANO, ANAME, ACITY)

A query of the form

SELECT DISTINCT S.SNO, P.PNO, P.PNAME

FROM SUPPLIER S, PARTS P

WHERE S.SNO = P.SNO AND

P.COLOR = ‘RED’

This query does not require a DISTINCT in SELECT clause since each tuple in result is uniquely identified by SNO and PNO.

For a Query involving Cartesian product of two relations R and S and having constraints CR on Relation R attributes, CS on Relation S attributes and CR,S involving attributes from both relations R and S, we would like to determine if it can contain duplicate rows.

This will be true either if

1. the primary keys from both relations R and S are present in the result or
2. if a subset of primary keys are present in the result and values of other key columns are equated to constant in the WHERE clause or can be inferred throught the WHERE clause.
3. **Goals of the Project**

To implement in Postgresql optimizer code, the elimination of DISTINCT clause if possible by checking the required set of conditions to be met for the same as specified in the paper [1].

To evaluate the performance of query without / with elimination of DISTINCT clause.

**4 Tasks Performed**

Some assumptions in the overall project

1. Queries will be of the form SELECT ……. FROM …… WHERE i.e. there will be no GROUP BY and HAVING CLAUSES.
2. The WHERE Clause will only have conjunctive constraints.
3. No two relations can have the same attribute name.

**Extracting information from the query**

This involved extracting the tables used in the SELECT query, the constraints in the WHERE clause and the columns in the result tuples.

The parser when generating the parse tree from the original query stores the information in structures from which it can be easily retrived. However, before control reaches the optimizer, all information gets stored in a complex form. Retrieving all the information from the query proved to be a challenge.

**Finding Primary key for tables used in SELECT clause**

**Applying the Algorithm for removing DISTINCT**

All these optimizations were performed just before the remaining part of optimizer is executed.

**Testing the performance for the changes implemented**

**5 Additional Features which can be implemented**

The shortage of time forced us to make certain assumptions in the input data. The algorithm can work better if we can implement it without restrictions.

The selection condition can also make use of candidate keys to implement the task. Hence considering attributes with UNIQUE constraint can be used.

**6 Conclusion**

We have implemented in Postgresql optimizer, the removal of DISTINCT clause under certain conditions as specified in [1] and with some additional constraints considering the duration in which it needs to be implemented. We have tested the performance for the changes made which look promising.

**REFERENCES**

[1] G.N. Paulley and Per-Ake-Larson Exploiting Uniqueness in Query Optimization.