Jonathan Petersen
A01236750
CS 5800 - Database Systems

SQL Query Optimization

Some notes:

- No matter what I did, the cost of Postgres.App's Examine function never seemed to change. I'm not sure if I set something up wrong or if the Query Optimizer is just that much smarter than me, but no amount of fiddling seemed to make the numbers budge. I'm just assuming that the optimizer is creating an optimal solution for these problems.

- Creating indexes also did not seem to help. I read online that PostgreSQL often ignores indexes if it thinks they won't be helpful, and perhaps that was the case with these queries. Regardless, I created indexes on every column used in my WHERE and JOIN clauses, but nothing seemed to change.

Query 1

Baseline cost=1559.08..1559.12

Optimized cost=1559.08..1559.12

I optimized this query by isolating the a.w and t.name constraints into their own subqueries, then forming a table joining those two results on teamID, lgID and yearID, and then doing a SELECT on the result. I then created indexes on the teamID, lgID, and yearID columns of Teams and Pitching, and the w column of Pitching and the name column of Teams.

Jonathan Petersen
A01236750
CS 5800 - Database Systems

Query 2

Baseline cost=158.34..158.49

Optimized cost=158.34..158.49

I created a subquery to isolate the LIKE '%Utah State%' query on Schools, then I formed a join with schoolsplayers on schoolID, joined with Master on masterID, joined with Batting on battingID, and then doing a SELECT on the result with a top-level WHERE ab IS NOT NULL.

Query 3

Baseline cost=6502.44..6549.24

Optimized cost=6502.44..6549.24

I created a subqery to find all of the entries of Master for Derek Jeter, then joined that with Appearances on masterID. I then did another join on Appearances on yearID and teamID, but also where jeter.masterID <> jeterT.masterID. I repeated this process for the other two levels, then ran SELECT on the result.

Query 4

Baseline cost=0.00..105581.21

Optimized cost=0.00..105581.21

I couldn't find a way to optimize this query. It seemed like most of the time was getting taken up by the MAX() function, and I couldn't think of a way to optimize that.

Jonathan Petersen
A01236750
CS 5800 - Database Systems

Query 5

Baseline cost=14300.57..14586.77

Optimized cost=14300.57..14586.77

All I found on this query was to move the A.avg >= B.avg WHERE clause into a JOIN on teamID, lgID, and yearID. I also moved the relation from C to D into a JOIN on cnt ≥ 0.75, and then ran SELECT on that.

Query 6

Baseline cost=6564.54..6564.55

Optimized cost=6564.54..6564.55

I created a subquery for Teams.name LIKE '%New York Yankees%', JOINed that to Batting on teamID, lgID, and yearID. Then I moved the constraints from the WHERE clause into JOINs on id and year+1 = year.

Query 7

Baseline cost=1616.76..1667.76

Optimized cost=1616.76..1667.76

I created a subquery to look at Salaries grouped by yearID, teamID, lgID, and then JOINed it to itself on teamID and lgID, and also on a.yearID + 1 = b.yearID and a.s *2 = b.s. I then ran SELECT on this result.