Project IV - Postgres – Query Processing

UC Berkeley Computer Science 186 Fall 2002
Introduction to Database Systems
November 13, 2002
Due Thursday, November 14, 5:00 pm

Overview of Project IV – Postgres Query Optimizer:

In this project you will carry out a number of exercises involving the optimization of relational queries using the postgres optimizer and the visualization command `EXPLAIN`. You will not have to write any code, and your answers should be turned in as a text file. `You will carry out this assignment individually`.

You will need to read parts of the "PostgreSQL 7.2.1 Documentation" to be able to complete the project (specific links are provided in the sub-sections). In particular you need to get familiar with the `EXPLAIN` command
`http://www.postgresql.org/idocs/index.php?sql-explain.html`
You can get all the files provided for this assignment from ~cs186/opt/.

You may find it easier to develop your code/website at home or on other machines you have access to. The instructions provided are designed to work with the EECS instructional machines and may not work as smoothly elsewhere. The TAs and Professor will only support EECS instructional machines.

Steps of Phase IV

1. Load the data given into the database
2. Analyze/Run the provided queries modifying the parameters as instructed
3. Write the answers to the questions for each of the queries in a text file
4. Submit

Please spend some time to read this assignment completely before beginning.

A. Loading the database into postgres

Using the instructions online (http://inst.eecs.berkeley.edu/~cs186/doc/postgres.html), create a PostgreSQL database called *opt.* Create the following tables: (you can use the schema file provided)

```
CREATE TABLE companies (
     co_id serial,
     co_name varchar(64),
     co_lastchg timestamp
);
```

```
CREATE TABLE products (
      pr_code varchar(6) PRIMARY KEY,
      pr_desc varchar(64)
);

CREATE TABLE orders (
      ord_id serial,
      ord_company int4 REFERENCES companies(co_id),
      ord_product varchar(6) REFERENCES products(pr_code),
      ord_qty int4,
      ord_placed date,
      ord_delivered date,
      ord_paid date
);
```

Load the data into these tables using the provided files *companies.data*, *products.data* and *orders.data*. Create an index `ord_placed_idx` on the attribute *`ord_placed`* for relation *`Orders`*.

Before continuing you should read sections VII:1-2, 9 of the"PostgreSQL 7.2.1 User's Guide" - http://www.postgresql.org/idocs/index.php?indexes.html to learn about indexes and their manipulation in postgres, section XI:1 http://www.postgresql.org/idocs/index.php?performance-tips.html for documentation on the `EXPLAIN` command (for visualizing query plans). Also read the sections III:4 of the "PostgreSQL 7.2.1 Administrator's Guide" http://www.postgresql.org/idocs/index.php?runtime-config.html for documentation on run time environment. You will need to learn about the `SET` command and variables for enabling/disabling access and join methods.

After completing step 1, you should be able to *analyze query plans* produced by the optimizer for a given query. You should also be able to modify the runtime variables for enabling/disabling join and access methods. You should update the postgres statistics after adding or deleting indexes. The command for updating statistics is `VACUUM ANALYZE.` You should update the statistics once before starting the exercises.

B. The Exercises

1. WARMUP

Examine the set of tables and access methods in the database you just created. In particular examine the relation *Orders.*

Answer the following questions:

a) What are the attributes of the *Orders* relation?
b) How many indexes are built on this relation? Name them and write down their type.
c) How many tuples are there in the *Orders* relation?

d) Use Postgres catalog to find the number of distinct values of each of the attributes in the *Orders* relation? Write down the query you used to find the above. Documentation for Postgres catalogs can be found at http://developer.postgresql.org/docs/postgres/catalogs.html

2. GETTING FAMILIARISED WITH QUERY PLAN VIEWER

This question requires you to use the postgres query plan visualization command `EXPLAIN`. The documentation on it can be found on `http://www.postgresql.org/idocs/index.php?sql-explain.html`

Consider the following query

```
SELECT * FROM orders
WHERE ord_qty = 56;
```

Answer the following questions looking at the query plan generated by the `EXPLAIN` command:

a) What is the estimated total cost of running the (estimated) best plan? What does the cost of a plan mean?
b) What is the estimated result cardinality for this plan? The estimated result cardinality is the number of orders that the optimizer estimates to have quantity 56
c) Given your knowledge about the number of different *`ord_qty`* values, is the estimate that the optimizer makes, a reasonable one? Explain very briefly.
d) In what order would the tuples be returned by this plan? Why?

3. SELECTS WITH INDEXES

Consider the same query from previous question

```
SELECT * FROM orders
WHERE ord_qty = 56;
```

Answer the following questions looking at the plans and the access methods

a) Create an index on the attribute *`ord_qty`* of the relation *`Orders`*. Find the best plan now. Which access methods does this plan use?
b) Disable the access method used by the best plan by using the `SET` command. Which access method does the optimizer consider to be the best now?
c) What is it about the way the best plan (from a. above) accesses tuples that makes it cheaper than the others?

4. RANGE SELECTS

Now analyze the query plan that postgres comes up for the following query that finds all orders made in the first 8 months of the year 2002.

```
SELECT * FROM orders
WHERE ord_placed < '2002-09-11';
```

Answer the following questions:

a) How many order tuples that have *ord_placed < '2002-09-11'* does the optimizer think there are?
b) How does the optimizer arrive at this estimate of the number of orders? That is, what calculations does it perform, and where does the supporting data come from?
c) In what order will the tuples be returned by this plan?
d) Disable the access method used by postgres in the plan in step a) What happens to the cost now? Does the order in which the tuples are returned change? Why?
e) Explain why one of the access methods is costlier than the other.

5. SIMPLE JOIN

Analyze the query plan for the following query that finds all products with quantity less than 5.

```
SELECT DISTINCT (pr_desc)
FROM orders, products
WHERE ord_product = pr_code
AND ord_qty < 5;
```

Answer the following questions

a) Write down the best plan estimated by the postgres optimizer. What is the estimated cost?
b) What kind of join is used by the (estimated) best plan?
c) How many tuples does the optimizer estimate it will retrieve from the `Orders` relation?
d) How big are the tuples that result from this query? (i.e. how many bytes does each tuple occupy?)

6. JOIN OPTIMIZATION

For the query in 5 disable the join type used by the optimizer in the (estimated) best plan. Can you think of a better plan than what the optimizer comes up with now without using the join type you have disabled? Write down the plan and the estimated cost in each case.

a) With existing access methods available on the tables.

b) With any additional indexes you want to add that would make the query run faster.
c) Can you write the query differently so that the cost of getting the same results is now less? Find the estimated cost for running this query.

7. THREE WAY JOIN

Answer the following questions referring to the query below

```
SELECT pr_desc, co_name
FROM orders, products, companies
WHERE ord_company = co_id
AND ord_product = pr_code;
```

a) Write down the best plan estimated by the optimizer. List the joins and access methods it uses, and the order in which the relations are joined
b) Modify the query by adding a condition ord_qty < 5. Describe the plan, explain any differences.

8. PLAYING WITH JOINS

Consider the following join query

```
SELECT * from orders, companies
WHERE ord_company = co_id
AND ord_qty < 40;
```

a) What type of join algorithm does the best query plan use? Write down the plan and the estimated cost?
b) Disable this join and recompute the estimated cost for the best plan. Describe the best plan produced by the optimizer in this case and note the differences from the plan above.
c) Disable the join used in b) above. Now what is the join used by the optimizer. Describe the best plan produced.

Submit

You should turn in brief answers to questions on a template provided. Print out the template and fill it in with your answers. *Note that the plans have to be written down in the tree form.* You should drop your answer sheets in the box provided in 2nd Floor Soda.