Carnegie Mellon University
15-415 Database Applications
Spring 2012, Faloutsos
Assignment 5: Query Optimization
Due: 3/20, 1:30 pm, in class – hard copy

Reminders - Important

● What to hand in: hard copy of your answers.
● Please type your answers.

Reminders – For your information

● Weight: 15% of the homework grade.
● The points of this homework add up to 100.
● Lead TA: U Kang
● Rough time-estimates: 2~4 hours.
Database Setup

We will work with PostgreSQL - a popular open-source database. We have put up a readme file on the course homepage with detailed instructions; link: http://www.cs.cmu.edu/~christos/courses/dbms.S12/hws/HW2/PostgreSQLReadme.htm

Here’s the quick summary to get you started:

1. We use the same machine newcastle.db.cs.cmu.edu as in homework 2. We assume you already have your id and password from homework 2.

2. Please backup your ~/db folder beforehand. On first login, run ../ukang415/setup_db.sh, press “y” to continue when prompted.

3. Run pg_ctl start -o -i, then press “Enter”.

4. Run psql.

5. Run SELECT COUNT(*) FROM movies;, the count should equal 2,680.

6. Run SELECT COUNT(*) FROM play_in2;, the count should equal 74,772.

7. Run \q to quit PostgreSQL.

8. IMPORTANT: Please stop the server using pg_ctl stop before logging out.
Getting Started

We use the following database tables. Notice that the movies table is the same as the one in homework 2, but the play_in2 table is different from the play_in table in homework 2. The only difference is the field ‘year’).

movies (mid, title, year, num_ratings, rating)
play_in2 (mid, name, year, cast_position)

The rest of the attributes are exactly as in homework2, and are repeated here, for your convenience:

In the table movies, mid is the unique identifier for each movie, title is the movie’s title, and year is the movie’s year-of-release. Each movie receives a total number of num_ratings ratings from users, and the average rating is rating on a scale of 0.0-10.0.

The table play_in2 contains the main cast of movies. name is the actor’s name (assume each actor has an unique name). year is the movie’s release year. cast_position is the order of the actor where he/she appears on the movie cast list (For example, in the movie Titanic, the cast_positions of Leonardo DiCaprio and Kate Winslet are 1 and 2, respectively).

[Resources]
The following documents are useful for the purpose of this assignment.

- Syntax of EXPLAIN command
  http://www.postgresql.org/docs/8.3/static/sql-explain.html
- How to use EXPLAIN command and understand its output
- Create an Index for a table
- Query Planner in PostgreSQL

[Index Management in PostgreSQL]
Check the commands “CREATE INDEX” and “DROP INDEX”.
To list all the indexes created, use the “\di” command in postgresql.

[Query Planner in PostgreSQL]
The SET command can be used to change the behavior of the query planner. For example, “SET enable_nestloop=false;” command in postgresql disables the query planner’s use of nested-loop join plans.
Question 1: Basic Query Optimization [20 points]

The goal of this question is to show you how to discover what plan the query optimizer is using, and whether (or not) an index makes a difference.

[Q1.1] Run the following sql query. What is the execution plan of the query? Use the EXPLAIN statement of postgresql to find out the execution plan. [4 points]

```sql
SELECT * from play_in2 WHERE cast_position=1;
```

[Q1.2] What is the estimated cost of the plan in Q1.1? [4 points]

[Q1.3] Build an index on `cast_position`. Run the same query in Q1.1. Now, what is the execution plan of the query? [4 points]

[Q1.4] What is the estimated cost of the plan in Q1.3? [4 points]

[Q1.5] Did the execution plan and the cost change after the addition of the index? Explain why they are the same or different. [4 points]

---

Question 2: Like Query Optimization [20 points]

The goal of this question is to show you cases where an index has a big impact, as well as cases when an index is not even used.

[Q2.1] Run the following sql query. What is the execution plan of the query? [4 points]

```sql
SELECT * from play_in2 WHERE name like '%smith%';
```

[Q2.2] What is the estimated cost of the plan in Q2.1? [4 points]

[Q2.3] Build an index on the `name` attribute of the relation `play_in2`. Run the same query in Q2.1. Now, what is the execution plan of the query? [4 points]

[Q2.4] What is the estimated cost of the plan in Q2.3? [4 points]

[Q2.5] Did the execution plan and the cost change after the addition of the index? Explain why they are the same or different. [4 points]
Question 3: Numerical Comparison Query Optimization [20 points]

The goal of this question is the same as Q2: to show you cases where an index has a big impact, as well as cases when an index is not even used.

[Q3.1] Run the following sql query. What is the execution plan of the query? [4 points]

```
SELECT * from movies WHERE rating*3 > 20;
```

[Q3.2] What is the estimated cost of the plan in Q3.1? [4 points]

[Q3.3] Create the index on the `rating` attribute of the relation `movies`. Run the query shown in Q3.1. Now, what is the execution plan of the query? [4 points]

[Q3.4] What is the estimated cost of the plan in Q3.3? [4 points]

[Q3.5] Did the execution plan and the cost change after the addition of the index? Explain why they are the same or different. [4 points]

Question 4: Join Optimization  [20 points]

The goal is to show you the impact of indices on joins

[Q4.1] Run the following sql query. [4 points]

```
SELECT * from movies, play_in2 WHERE movies.year=play_in2.year;
```

What is the execution plan of the query? What is the estimated cost of the plan?

[Q4.2] Make an index on `movies.year`, and run the query shown in Q4.1. Show the plan and the estimated cost. [4 points]

[Q4.3] Did the execution plan and the cost change after the addition of the index at Q4.2? Explain why they are the same or different. [4 points]

[Q4.4] Make an additional index on `play_in2.year`. Now there are two indexes created, each for `movies.year` and `play_in2.year`. Run the query shown in Q4.1. Show the plan and the estimated cost. [4 points]

[Q4.5] Did the execution plan and the cost change after the addition of the index at Q4.4? Explain why they are the same or different. [4 points]
Question 5: Enforcing Join Algorithm [20 points]

The goal is to show you how to manipulate the query optimizer, in case you (as the DBA) disagree with the plan it has picked

Q5.1 Run the following sql query.

```
SELECT * FROM movies, play_in2 WHERE movies.mid=play_in2.mid;
```

What is the execution plan of the query? What is the estimated cost of the plan? Which join algorithm is used? [8 points]

Q5.2 Disable the join algorithm used in the plan of the query in Q5.1, by using the “SET” command inpostgresql. Now, what is the execution plan of the query? What is the estimated cost of the plan? Which join algorithm is used? [6 points]

Q5.3 Disable the join algorithm used in the plan in Q5.2, by using the “SET” command inpostgresql. At this point, you disabled the two join algorithms (one from Q5.2, the other from Q5.3). Now, what is the execution plan of the query? What is the estimated cost of the plan? Which join algorithm is used? [6 points]