
Assigned: Monday, Sep 28, 2015
Due: Wednesday, Oct 14, 2015, 11:59 PM
Submit to Stellar: [https://stellar.mit.edu/S/course/6/fa15/6.830/homework/]

The purpose of this problem set is to give you some practice with concepts related to schema design, query planning, and query processing. Start early as this assignment is long.

Part 1 - Query Plans and Cost Models

In this part of the problem set, you will examine query plans that PostgreSQL uses to execute queries, and try to understand why it produces the plan it does for a certain query.

We are using the same database as for Problem Set 1. The command to connect has changed slightly, however:

```
psql -h geops.csail.mit.edu -p 5433 wikidb
```

Make sure your PostgreSQL client is 9.3+ so that your results are consistent with the solutions. Athena already has client version 9.3.9 installed, so you can simply ssh into athena.dialup.mit.edu and get started. In case you want to work on your own Debian/Ubuntu machine, you can install the `postgresql-client` package by running the following command in your shell.

```
sudo apt-get install postgresql-client
```

To help understand database query plans, PostgreSQL includes the `EXPLAIN` command. It prints the physical query plan for the input query, including all of the physical operators and internal access methods being used. For example, the SQL command displays the query plan for a very simple query:

```
postgres@wikidb# explain select * from page;
QUERY PLAN
---------------------------------------------------------------
Seq Scan on page  (cost=0.00..10329.80 rows=374880 width=106)
(1 row)
```

To be able to interpret plans like the one above, you should refer to the `explain basics` section in the Postgres documentation.

We have run `VACUUM FULL ANALYZE` on all of the tables in the database, which means that all of the statistics used by PostgreSQL server should be up to date.
1. [1 points]: Which indices exist for table page in wikidb?

2. [2 points]: Which query plan does postgres choose for select page_id from page?. Is it different from the plan shown in the previous page? Given the indices we have defined on our table, are there any other possible query plans?

3. [1 points]: In one sentence, describe the difference between the plan from the previous question and the plan for query: select page_id from page order by page_id.

4. [3 points]: Both of the queries above (select page_id from page and select page_id from page order by page_id) need to only use the value of a column which has already been indexed. Because of this, Ben Bitdiddle is convinced that both these queries run fastest using an index-only scan. Can you use Postgres to provide some evidence that confirms or disproves Ben's theories about access methods?

5. [3 points]: Consider the two following queries and their plans from wikidb:

```sql
wikidb=> explain analyze select page_namespace, page_title from page where page_title = 'abc';
```

```
QUERY PLAN

Index Only Scan using page_page_namespace_page_title_idx on page
(cost=0.42..10332.03 rows=1 width=20) (actual time=31.226..31.226 rows=0 loops=1)
  Index Cond: (page_title = 'abc'::text)
  Heap Fetches: 0
Planning time: 0.107 ms
Execution time: 31.264 ms
```

```sql
wikidb=> explain analyze select page_namespace, page_title from page where page_namespace = 11;
```

```
QUERY PLAN

Index Only Scan using page_page_namespace_page_title_idx on page
(cost=0.42..308.87 rows=172 width=20) (actual time=0.122..0.399 rows=628 loops=1)
  Index Cond: (page_namespace = 11)
  Heap Fetches: 628
Planning time: 0.093 ms
Execution time: 0.460 ms
```

The two queries and their plans are very similar and make use of the same index. Why are the costs (both the estimates and actual) so different?

Now consider the queries generated by replacing $0$ with 10, 20 and 30 in the following template. The query computes the number of links between pages of different sizes. (You can call the three queries Q10, Q20 and Q30 respectively).
explain select p0.page_id, p1.page_id from page as p0, pagelinks as pl01, page as p1
where (p0.page_id = pl01.pl_from)
and (pl01.pl_title, pl01.pl_namespace) = (p1.page_title, p1.page_namespace)
and p0.page_len > 4000 and p1.page_len < $0;

6. [2 points]: What physical plan does PostgreSQL use for each of them? Your answer should consist of a drawing of the three query trees and annotations on each node.

7. [1 points]: Which access methods are used? (also label them in the diagrams)

8. [1 points]: Which join algorithms? (also label them in the diagrams)

9. [2 points]: By running some queries to compute the sizes of the intermediate results in the query, and/or using EXPLAIN ANALYZE, can you see are there any final or intermediate results where PostgreSQL’s estimate is less than half (or more than double) the actual size?

10. [4 points]: At which values of p1.page_len (in the range of 0 to 30) do the plans change? Do you believe the query planner is switching at the correct points? (justify your answer quantitatively)
Part 2 – Query Plans and Access Methods

In this problem, your goal is to estimate the cost of different query plans and think about the best physical query plan for a SQL expression.

You have been hired to optimize the database of a hot new startup, “yomomma.com”, where users can send their friends “Yo’ Momma” jokes (these are jokes like “Your momma’s so stinky, I thought she was cheese.”). The interface to the application is very simply – you can invite and add friends, and then tap on a friend to send them a new joke they have not received before. The website maintains a large and growing database of jokes. Friends in yomomma are symmetric (i.e., if A is B’s friend, B is A’s friend), and this friendship will be represented by two tuples in the friends table. When user receives a joke, both the user and their friends can choose to “like” it.

The database contains the following tables:

```sql
-- user u_uid with a name, gender, and birthday
CREATE TABLE users(
u_id int PRIMARY KEY,
u_name char(50),
u_gender char,
u_bday date);

-- table indicating f_uid1 and f_uid2 are friends, and the date they became friends
-- if table contains a friendship pair (X,Y), it will also include the pair (Y,X)
CREATE TABLE friends(
f_fship_id int PRIMARY KEY,
f_uid1 int REFERENCES users(u_id),
f_uid2 int REFERENCES users(u_id),
f_friended date);

-- table of jokes
CREATE TABLE jokes(
j_id int PRIMARY KEY,
j_joke char(400) -- all jokes are allocated to be 400 chars
);

-- table of jokes sent, from one user to another
CREATE TABLE sent(
s_id int PRIMARY KEY,
s_from int REFERENCES users(u_id),
s_to int REFERENCES users(u_id),
s_j_id int REFERENCES jokes(j_id)
);

-- table of likes, by a particular user, of a particular sent joke
CREATE TABLE likes(
l_id int PRIMARY KEY,
l_s_id int REFERENCES sent(s_id),
l_uid int REFERENCES users(u_id)
);
In this database, \texttt{int} and \texttt{date} values are 8 bytes each and characters are 1 byte. All tuples have an additional 8 byte header. This means, that, for example, the size of a single joke record is $8 + 8 + 400 = 416$ bytes.

You create these tables in a row-oriented database. The system supports heap files and B+-trees (clustered and unclustered). B+-tree leaf pages point to records in the heap file. Assume you can cluster each heap file in accordance to exactly one B+-tree, and that the database system has up-to-date statistics on the cardinality of the tables, and can accurately estimate the selectivity of every predicate. Assume B+-tree pages are 50% full, on average. Assume disk seeks take 10 ms, and the disk can sequentially read 100 MB/sec. In your calculations, you can assume that I/O time dominates CPU time (i.e., you do not need to account for CPU time.)

For the queries below, you are given the following statistics:

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Number of friends</td>
<td>$10^7$</td>
</tr>
<tr>
<td>Number of jokes</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Number of jokes sent</td>
<td>$10^8$</td>
</tr>
<tr>
<td>Number of likes</td>
<td>$10^9$</td>
</tr>
</tbody>
</table>

In the absence of other information, assume that attribute values are uniformly distributed (e.g., that there are approximately the same number of friends per user, jokes sent per user, likes per joke, etc).

Suppose you are running the query \texttt{SELECT COUNT(*) FROM likes WHERE l_id = 1}. Answer the following:

11. [1 points]: In one sentence, what would be the best plan for the DBMS to use assuming no indexes? Approximately how long would this plan take to run, using the costs and table sizes given above?

12. [2 points]: In one sentence, what would be the best plan for the DBMS to use assuming a clustered B+tree index on \texttt{l_id}? Approximately how long would this plan take?

13. [2 points]: In one sentence, what would be the best plan for the DBMS to use assuming an unclustered B+tree index on \texttt{l_id}? Approximately how long would this plan take?

Now consider the following query. For a given user U, this query computes the number of non-friends of U who 'liked' a joke sent by U.

\begin{verbatim}
SELECT COUNT(*)
FROM likes AS l, sent AS s, users AS u
WHERE l.l_uid NOT IN (SELECT f_uid2
                      FROM friends AS f
                      WHERE f.f_uid1 = u.u_id)
AND u.u_name = 'Bob Smith'
AND u.u_id = s.s_from
AND l.l_s_id = s.s_id
\end{verbatim}

Note that the subquery references the \texttt{u.u_uid} value from the outer query (this value represents the user who sent the joke). One possible way to execute this subquery is to simply run a separate subquery over the \texttt{friends} table for each like. You can represent this in a query plan by a select node where the select condition is a subquery (you may be able to come up with a more efficient plan but this is not required.)
14. **[3 points]**: Suppose only heap files are available (i.e., there are no indexes), and that the system supports grace (hybrid) hash, merge join, and nested loops joins. For each node in your query plan indicate (on the drawing, if you wish), the approximate output cardinality (number of tuples produced.)

15. **[2 points]**: Estimate the runtime of the plan you drew, in seconds.

16. **[2 points]**: Now, suppose that there are clustered B+Trees on u_id, l_uid, s_from, and f_uid1, and an unclustered B+Tree on u_name, s_id, and l_s_id. Draw the new plan you think the database would use and estimate its runtime methods available to it.

17. **[2 points]**: Suppose you could choose your own indexes (assuming at most one clustered index per table) for this plan; what would you choose, and why? Justify your answer quantitatively.
Part 3 – Schema Design and Query Execution

A medical lab testing company has several testing centers all over the country. In this problem, you will design a schema to keep track of the testing centers, tests and order information.

Specifically, you will need to keep track of:

1. The address, city, state and manager for each testing center.
2. The equipment at each testing center including the name of the machine, manufacturing year, and status of the machine (Functional, in repair, phased out). Note that one center may have multiple machines of the same kind.
3. The tests that a center runs including the test id, name, type (blood, urine, stool etc), time to run and price. The price of the test may depend on the testing center.
4. The orders for tests including the test id, doctor who submitted the test, the date the test was submitted, the patient for whom the test was ordered.
5. The results of tests including the date the test was run, time taken to run and results. For this problem, assume that each test produces a single real number.

18. [2 points]: Write out a list of functional dependencies for this schema. Not every fact listed above may be expressible as a functional dependency.

19. [3 points]: Draw an ER diagram representing your database. Include a few sentences of justification for why you drew it the way you did.

20. [2 points]: Write out a schema for your database in BCNF. You may include views. Include a few sentences of justification for why you chose the tables you did.

21. [2 points]: Is you schema redundancy and anomaly free? Justify your answer.

22. [3 points]: Suppose you wanted to ensure that no test can be run without a doctor ordering it. How can you enforce this constraint? In addition, you want to ensure that no more than 10% of active machines (not phased out) in a testing center can be in repair at any time. How can you enforce this constraint?