6.830 Problem Set 2 (2016)

Assigned: Monday, Sep 26, 2016

Due: Monday, Oct 17, 2016, 11:59 PM
Submit to Gradescope: [https://gradescope.com/courses/4342](https://gradescope.com/courses/4342)

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. To access the server, you can log in to [athena.dialup.mit.edu](http://athena.dialup.mit.edu) and start a session with:

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

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:

```
explain select * from business;
```

```
QUERY PLAN
---------------------------------------------------------------
Seq Scan on business  (cost=0.00..121.86 rows=4086 width=125)
(1 row)
```

To be able to interpret plans like the one above, you should refer to the [explain basics section in the Postgres documentation](https://www.postgresql.org/docs/current/). 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.
Note: To identify an index, it is enough for you to name the ordered sequence of columns that are indexed. Eg, an index on columns foo and bar is identified as (foo, bar).

1. **[1 points]:** Which indexes exist for table business in yelp?
   
   **Answer:** There is one on business.id and one on business.city.

2. **[2 points]:** Which query plan does Postgres choose for `select city from business;`? Is it different from the plan shown in the previous page? Given the indexes we have defined on our table, are there any other possible query plans?
   
   **Answer:** Postgres picks a sequential scan on the page.
   
   ```sql
   Seq Scan on business (cost=0.00..121.86 rows=4086 width=10)
   ``
   
   It's the same as the one in the previous page. It's also possible for Postgres to choose an index scan.
   
   ```sql
   Index Only Scan using business.city on business (cost=0.28..133.57 rows=4086 width=10).
   ```

3. **[1 points]:** In one sentence, describe the difference between the plan from the previous question and the plan for query `select city from business order by city;`.
   
   **Answer:** Postgres uses an index scan instead of a seq scan.

We now add another table `business_wide` into yelp. `business_wide` has an additional column with type character(10000). Consider the two following queries and their plans from yelp:

```sql
explain select city from business;
```

```
QUERY PLAN
-----------------------------
Seq Scan on business
  (cost=0.00..125.86 rows=4086 width=10)
(1 row)
```

```sql
explain select city from business_wide;
```

```
QUERY PLAN
-----------------------------
Index Only Scan using business_wide.city on business_wide
  (cost=0.28..133.57 rows=4086 width=10)
(1 row)
```

4. **[2 points]:** Both of the queries only need to use the value of a column which has already been indexed. From the perspective of PostgreSQL's optimizer, why does it choose a seq scan on business but an index only scan on business_wide?
   
   **Answer:** PostgreSQL's optimizer thinks it's cheaper to do a seq scan over business instead of an index scan. However, in the case of business_wide, an index only scan can save lots of I/Os because of the wide column with 10,000 bytes.

5. **[2 points]:** Now run this scan on both tables, using index only scans and sequential scans. How does their actual performance compare? Is Postgres's optimizer correct? If not, what do you think it is doing wrong?
You can coerce Postgres into using a variety of query plans by using the `enable_seqscan`, `enable_indexonlyscan`, `enable_indexscan` and the `enable_bitmapscan` flags with the `\pset` command.

**Answer:** Postgres’s optimizer is wrong on table `business`. An index scan is faster than a seq scan. Postgres’s optimizer is correct on table `business_wide`. The answer to this problem depends on the internal state of Postgres. As long as reasonable evidence is provided in your write up, you will receive full credit.

---

6. [3 points]: Consider the two following queries and their plans from `yelp`:

```sql
explain analyze select business_id, user_id from reviews
where business_id = '5UmKMjUEUNdYWqANhGckJw';
```

---

Index Only Scan using `reviews_business_id_user_id` on `reviews`
6.830 Problem Set 2

explain analyze select business_id, user_id from reviews
where user_id = 'PUFPaY9KxDAcGfsorJp3Q';

Index Only Scan using reviews_business_id_user_id on reviews
(cost=0.42..4565.03 rows=5 width=46)
(actual time=0.517..10.307 rows=9 loops=1)
Index Cond: (user_id = 'PUFPaY9KxDAcGfsorJp3Q'::bpchar)
Heap Fetches: 9
Planning time: 0.108 ms
Execution time: 10.334 ms
(5 rows)

The two queries are similar, but the order of the index key matters.
The index on (business_id, user_id) can tell exactly where the data block is given a business_id: On the other hand, we have to scan much more of the index in order find the data block given a user_id!.

Now consider the queries generated by replacing $0 with ‘10’ and ‘100’ in the following template. (You can call the two queries Q10 and Q100 respectively).

explain select * from reviews
join business on reviews.business_id = business.business_id
join users on reviews.user_id = users.user_id
where date > '2008-01-01' and fans > $0;

7. [2 points]: What physical plan does PostgreSQL use for each of them? Your answer should consist of a drawing of the two query trees and annotations on each node. Answer: see Figure[1]

8. [1 points]: Which access methods are used? (also label them in the diagrams) Answer: see Figure[1]

9. [1 points]: Which join algorithms? (also label them in the diagrams) Answer: see Figure[1]

10. [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 there are any final or intermediate results where PostgreSQL’s estimate is less than half (or more than double) the actual size? Answer: see Figure[1]
11. [4 points]: At which values of \texttt{fans} (in the range of 0 to 1354) do the plans change? Do you believe the query planner is switching at the correct points? (justify your answer quantitatively).

Answer:

The switch point happens when \texttt{fans} = 30. We can disable nested loop join and run the query with \texttt{fans} = 31. We can observer hash join(79.018 ms) is faster than a nested loop join(114.600ms). Therefore, postgres switched early.
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.

Your job is to optimize the database for a new online service, “petrentals.com”, where users can rent pets for a few days at a time. The database needs to store information about pets, who they are rented to, as well as the past history of rentals and the condition in which they were returned.

The database contains the following tables:

```sql
-- list of types of animals, including their species (e.g. dog) and breed (e.g., goldendoodle)
CREATE TABLE animal_types (  
a_id integer PRIMARY_KEY,  
a_species character(100),  
a_breed character(100),  
a_description character(400)
);

-- list of available pets, and a description and price
CREATE TABLE pets (  
p_id integer PRIMARY KEY,  
p_a_id integer REFERENCES animal_types(a_id),  
p_description character(400),  
p_rental_price_per_day float
);

-- list of users of petrentals.com
CREATE TABLE users (  
u_id integer PRIMARY KEY,  
u_name character(50),  
u_address character(100),  
u_ccno character(16)
);

-- list of rentals that have been performed
-- r_return_date will be null if the animal has not been returned
-- r_return_condition is 0 if the animal was returned alive, 1 otherwise
-- dates range from 1/1/2015 to 9/20/2016
CREATE TABLE rentals (  
r_id integer PRIMARY KEY,  
r_rental_date date,  
r_return_date date,  
r_return_condition integer,  
r_p_id integer REFERENCES pets(p_id),  
r_u_id integer REFERENCES users(u_id)
);

-- list of reviews that users have input for particular rentals
-- stars is between 1 and 5
CREATE TABLE reviews (  
rev_id integer PRIMARY KEY,  
rev_r_id integer REFERENCES rentals(r_id),  
review character(400),  
stars integer
);
```
In this database, **int**, **float**, and **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 **users** record is 8 + 8 + 50 + 100 + 16 = 182 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 according 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 pets</td>
<td>10^5</td>
</tr>
<tr>
<td>Number of rentals</td>
<td>10^8</td>
</tr>
<tr>
<td>Number of reviews</td>
<td>10^6</td>
</tr>
<tr>
<td>Number of animal types</td>
<td>10^3</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 rentals per user, rentals sent per pet, reviews per rental, etc).

Suppose you are running the query

```sql
SELECT r.p_id FROM rentals WHERE r.rental_date > X GROUP BY r.p_id HAVING COUNT(*) > Y
```

Answer the following:

12. [1 points]: Suppose that X is 9/15/2016. Would you recommend building any indexes for this query? If yes, which ones (and would you cluster them or not)? If not, why not?

**Answer:** Yes, on `r.rental_date`, `r.p_id`

A index will dramatically reduce the amount of data that has to be read, adding `r.p_id` will allow the query to be answered directly from the index. Clustered is not important.

13. [2 points]: Suppose that X is 1/15/2015. Would you recommend building any indexes for this query? If yes, which ones (and would you cluster them or not)? If not, why not?

**Answer:** Even though the selectivity reduction isn?t great, the benefit of not going to the heap file (answering directly from index) will help quite a bit.

14. [2 points]: Assuming Y > 0, do you expect the value of Y to affect the query plan that is generated in any way?

**Answer:** No
(SELECT AVG(stars) as rating
FROM reviews, rentals, pets
WHERE rev_r_id = r_id
AND r_p_id = p_id
GROUP BY p_id)

Note that the outer query repeats the joins in the inner query; short of using window functions or temporary tables, this is necessary in SQL because there is no way to find the minimum average rating and the pets with that lowest average in a single query.

15. [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. Draw the query plan you think a database would use for this query. For each node in your query plan indicate (on the drawing, if you wish), the approximate output cardinality (number of tuples produced.)

Answer:

16. [2 points]: Now, suppose that there are clustered B+Trees on \( p\_id, r\_id, \) and \( rev\_r\_id \) and an unclustered B+Tree on \( r\_p\_id \). Draw the new plan you think the database would use and estimate its runtime, assuming the same join algorithms as in the previous question plus index nested loops joins are available.

Answer:
Part 3 – Schema Design and Query Execution

Ben Bitiddle wants to create a video-sharing website called MiniTube. In this problem, you will help him to design a schema for the MiniTube server.

Specifically, you will need to keep track of:

1. The name, password (suitably salted/hashed, of course!), avatar, and short bio for each registered user.
2. The videos shared by each user including its name, length, description, user id of poster, and the video url.
3. Comments on videos, which include user id, video id, content, and timestamp.
4. Users can upvote or downvote a video.

17. [2 points]: Write out a list of functional dependencies for this schema. Answer:
   
   - uid \Rightarrow \text{name, password, avatar, bio}
   - vid \Rightarrow \text{name, length, description, uid, poster, url}
   - cid \Rightarrow \text{uid, vid, content, timestamp}
   - voteid \Rightarrow \text{uid, vid, vote}

18. [3 points]: Draw an ER diagram representing your database. Include a few sentences of justification for why you drew it the way you did.

19. [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.
20. **[2 points]**: Is your schema redundancy and anomaly free? Justify your answer.

**Answer:** Yes, since the schema is in BCNF, it is redundancy and anomaly free with respect to functional dependencies.

21. **[3 points]**: Suppose you wanted to ensure that each user can either upvote or downvote each video at most once. How can you enforce this constraint? In addition, you want to ensure that no more than 90% of votes on videos with more than 100 votes can be downvoted at any time, in detection of inappropriate content. How can you enforce this constraint?

**Answer:** The first constraint can be enforced with a unique constraint on (uid, vid). The second can be enforced through the use of triggers.

22. **[3 points]**: Ben Bitbiddle decides to add a new feature to allow users to comment on other user’s comments, i.e. the comments are now structured in a tree. How would you change your schema? Is the new schema still redundancy and anomaly free? If not, can you fix it?

**Answer:** We can add a parent id field indicating the parent comment. No, parent comment and child comments all refer to the vid. We could fix it by treating the top level comment differently, i.e. having vid in the top-level comments and parent id on other comments.