6.830 Problem Set 2 (2014)

Assigned: Monday, Sep 22, 2014

Due: Wednesday, Oct 8, 2014, 11:59 PM
Submit to Stellar: `https://stellar.mit.edu/S/course/6/fa14/6.830/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.

The data set you will use has the same schema as the MIMIC-II dataset you used in problem set 1. Rather than running your own instance of SQLite, however, you will be connecting to our PostgreSQL server, since PostgreSQL produces more interesting query plans than SQLite.

To understand what query plan is being used, PostgreSQL includes the `EXPLAIN` command. It prints the plan for a query, including all of the physical operators and access methods being used. For example, the following SQL command displays the query plan for the `SELECT`:

```
EXPLAIN SELECT * FROM d_meditems WHERE label LIKE '%Folate%';
```

In this problem, you will find `\di` and `\d tablename` commands useful. In order to use these, you must install PostgreSQL command-line client. **Make sure you use PostgreSQL 8.3+ so that your results are consistent with the solutions.**

Athena already has version 9.3.5 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
```

You can then connect to our PostgreSQL server by running the following command.

```
psql -h vise3.csail.mit.edu -U mimic2 -d mimic2
```

Note that we currently only allow connections from MIT IPs so you will need to connect from on campus or by ssh'ing into Athena.

To understand the output of `EXPLAIN`, you will probably want to read the performance tips chapter of the PostgreSQL documentation:

`http://www.postgresql.org/docs/9.2/static/performance-tips.html`

In general to understand the plans that are generated you may need to spend a bit of time searching the Internet.

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. [15 points]: *Query Plans*

Run the following query (using EXPLAIN) in PostgreSQL and answer questions (a) – (f) below:

```
EXPLAIN SELECT ci.itemid, ci.label, count(*)
FROM d_chartitems ci,
     chartevents ce,
     d_patients p,
     demographic_detail d
WHERE ci.itemid = ce.itemid
AND d.subject_id = p.subject_id
AND ce.subject_id = p.subject_id
AND d.overall_payor_group_descr = 'MEDICAID'
AND p.sex = 'F'
AND ce.icustay_id IS NOT NULL
GROUP BY ci.itemid, ci.label
ORDER BY ci.itemid, ci.label;
```

a. What physical plan does PostgreSQL use? Your answer should consist of a drawing of a query tree annotated with the access method types and join algorithms (note that you can use the pgadmin3 tool shown in class to draw plans, but will need to annotate them by hand.)

b. Why do you think PostgreSQL selected this particular plan?

c. What does PostgreSQL estimate the size of the result set to be?

d. When you actually run the query, how big is the result set?

e. Run some queries to compute the sizes of the intermediate results in the query. Where do PostgreSQL's estimates differ from the actual intermediate result cardinalities?

f. Given the tables and indices we have created, do you think PostgreSQL selected the best plan? You can list all indices with `\di`, or list the indices for a particular table with `\d tablename`. If not, what would be a better plan? Why?

2. [15 points]: *Estimating Cost of Query Plans*

Use `EXPLAIN` to find the query plans for the following two queries and answer question (a).

1. ```
   EXPLAIN SELECT m.label
   FROM d_chartitems AS c,
        chartevents AS ce,
        medevents AS me,
        d_meditems AS m
   WHERE c.itemid=ce.itemid
   AND ce.subject_id=me.subject_id
   AND me.itemid=m.itemid
   AND ce.subject_id > 100
   AND m.itemid>370;
   ```
2. ```
   EXPLAIN SELECT m.label
   FROM d_chartitems AS c,
        chartevents AS ce,
        medevents AS me,
        d_meditems AS m
   WHERE c.itemid=ce.itemid
   AND ce.subject_id=me.subject_id
   AND me.itemid=m.itemid
   AND ce.subject_id > 27000
   AND m.itemid>370;
   ```

a. Notice that the query plans for the two queries above are different, even though they have the same general form. Explain the difference in the query plan that PostgreSQL chooses, and explain why you think the plans are different.

Now use `EXPLAIN` to find the query plan for the following query, and answer questions (b) – (g).

3. ```
   EXPLAIN SELECT m.label
   FROM d_chartitems AS c,
        chartevents AS ce,
        medevents AS me,
        d_meditems AS m
   WHERE c.itemid=ce.itemid
   AND ce.subject_id=me.subject_id
   AND me.itemid=m.itemid
   AND ce.subject_id > 26300
   AND m.itemid>370;
   ```

b. What is PostgreSQL doing for this query? How is it different from the previous two plans that are generated? You may find it useful to draw out the plans (or use pgadmin3) to get a visual representation of the differences, though you are not required to submit drawings of the plans in your answer.

c. Run some more EXPLAIN commands using this query, sweeping the constant after the '$>$' sign from 100 to 27000. For what value of $subject\_id$ does PostgreSQL switch plans?
Note: There might be additional plans other than these three. If you find this to be the case, please write the estimated last value of subject_id for which PostgreSQL switches query plans.

d. Why do you think it decides to switch plans at the values you found? Please be as quantitative as possible in your explanation.

e. Suppose the crossover point (from the previous question) between queries 2 and 3 is $subject\_id_{23}$. Compare the actual running times corresponding to the two alternative query plans at the crossover point. How much do they differ? Do the same for all switch points.
Inside `psql`, you can measure the query time by using the `\timing` command to turn timing on for all queries. To get accurate timing, you may also wish to redirect output to `/dev/null`, using the command `\o /dev/null`;
   ```

later on you can stop redirection just by issuing the \o command without any file name. You may also wish to run each query several times, throwing out the first time (which may be longer if the data is not resident in the buffer pool) and averaging the remaining runs.

f. Based on your answers to the previous two problems, are those switch points actually the best place to switch plans, or are they overestimate/underestimate of the best crossover points? If they are not the actual crossover point, can you estimate the actual best crossover points without running queries against the database? State assumptions underlying your answer, if any.

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.

Suppose you are running a web service, "sickerthanyou.com", where users can upload their genome data, and that uses genome processing algorithms to provide reports to users about their genetic disorders.

Data is uploaded as files, with metadata about those files stored in a database table. Each file is passed through one or more fixed processing pipelines, which produce information about the disorders a patient has. These results are stored in another database table.

The database contains the following tables:

```
// user u_uid with a name, gender, age, and race
CREATE TABLE users(u_uid int PRIMARY KEY,
u_name char(50),
u_gender char,
u_age int,
u_race int)

// relatives table records familial relationships between users
// relationships are not symmetric, so e.g., if A is B's child, then
//  there will be two entries: (A,B,child) and (B,A,parent)
CREATE TABLE relatives(rl_uid1 int,
         rl_uid2 int,
         rel_type int,
       PRIMARY KEY (rl_uid1, rl_uid2));

// data set collected about a specific user on a specific
//  genome sequencing platform,
//  with results stored in a specified file
CREATE TABLE datasets(d_did int PRIMARY KEY,
d_time timestamp,
d_platform char(20),
d_uid int REFERENCES users(u_uid),
d_file char(100));

// results produced by processing pipeline
// each disorder column represents a specific condition
//  (e.g., disorder1 might be diabetes, disorder2 might be a type of cancer, etc.)
CREATE TABLE results(r_rid int PRIMARY KEY,
r_did int REFERENCES datasets(d_did),
        r_uid int REFERENCES users(u_uid),
r_pipeline_version int,
r_timestamp int,
r_disorder1probability float,
...
r_disorder50probability float)
```

In this database, `int`, `timestamp`, and `float` 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 `results` record is $8 + 8 \times 4 + 8 \times 50 = 440$ 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 that you can have at most one clustered index per file, 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:

Statistic	Value
Number of users	10^6
Number of datasets	10^7
Number of results	10^8
Number of relatives	10^7

In the absence of other information, assume that attribute values are uniformly distributed (e.g., that there are approximately the same number of relatives per user, datasets per user, results per dataset, etc).

Suppose you are running the query `SELECT COUNT(*) FROM results WHERE d_uid = 1`. Answer the following:

3. [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?

4. [2 points]: In one sentence, what would be the best plan for the DBMS to use assuming a clustered B+tree index on `d_uid`? Approximately how long would this plan take?

5. [2 points]: In one sentence, what would be the best plan for the DBMS to use assuming an unclustered B+tree index on `d_uid`? Approximately how long would this plan take?

Now consider the following query that finds genome files of people who have female relatives with a high likelihood of having a particular disorder:

```
SELECT d_file
FROM users as u, relatives as rl, datasets as d, results as r
WHERE d_uid = rl_uid1
AND rl_uid2 = u_uid
AND r_uid = u_uid
AND u_gender = 'F'
AND r_disorder1probability > .85
AND r_timestamp = (SELECT MAX(r_timestamp) FROM results WHERE r_uid = u_uid)
```

Note that most database systems will execute the subquery as a join node where the right hand (inner) side of the join is the aggregate query, parameterized by `u_uid`, and the left hand is a table or intermediate result that contains `u_uid`.

6. [3 points]: Suppose only heap files are available (i.e., there are no indexes), and that the system has 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.)

7. [2 points]: Estimate the runtime of the plan you drew, in seconds.

8. [2 points]: Now, suppose that there are clustered B+Trees on `u_uid, d_uid, rl_uid1, r_uid`, and an unclustered B+Tree on `rl_uid2`. Draw the new plan you think the database would use and estimate its runtime methods available to it.

9. [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.

10. [3 points]: Now suppose you load the same data into a column-store. Suppose there are no unclustered indexes, but that you can sort each table in some specific key, and that the database can quickly lookup records according to the sort key (e.g., if you sort `results` on `r_disorderprobability1`, you can quickly find all `r_disorderprobability1` records with value $> .85$. Further assume that the column-store has 1 byte of overhead per column (instead of the 8 bytes of overhead per record in the row-store.) For each table, list the sort key you would choose, and estimate the runtime of the above query in the column store (you don't need to draw out the query plan – just estimate the I/O costs.)

Part 3 – Schema Design and Query Execution

An online-shopping company has its own delivery system including several warehouses. Your job in this problem set is to design a schema for keeping track of the warehouses, orders and the logistics.

Specifically, you will need to keep track of:

1. The address, city, state, manager (unique), employees of every warehouse
2. The name, phone-number, salary of every employee and manager
3. The order_ID, product_ID, custumer_ID of every order
4. The shipping information of an order, which may include several records about shipping from one warehouse to another, with shipping time, delivery time, and processors information (one at each end)
5. The maximum numbers of orders that can be delivered between each two warehouses everyday

11. [2 points]: Write out a list of functional dependencies for this schema. Not every fact listed above may be expressible as a functional dependency.

12. [3 points]: Draw an ER diagram representing your database. Include a few sentences of justification for why you drew it the way you did.

13. [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.

14. [2 points]: Is you schema redundancy and anomaly free? Justify your answer.

15. [3 points]: Suppose you wanted to ensure that on each day, the number of orders that are shipped is less than 95% of the maximum delivering capability between two warehouses. Can you think of a way to enforce that via the schema? (i.e., by creating a table or modifying one of your existing tables?) How else might you enforce this?