6.830 Problem Set 2 (2010)

Assigned: September 28, 2010

Due: October 12, 2010, 11:59 PM

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

We show two SQL queries over the NSF database used in Problem Set 1 below. For these two queries, you will examine the query plan that Postgres uses to answer the query, and try to understand why it produces the plan it does.

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 name FROM researchers WHERE researchers.name like '%Stonebraker%' ;
```

In this problem set, you will find `\di` and `\d tablename` useful. In order to use these, you must install Postgres 7.4 and cd into the directory in order to use Postgres 7.4. **Make sure you use Postgres 7.4 so that your results are consistent with the solutions.**

If you did not acquire Postgres 7.4 in the previous problem set, you can do so by doing the following commands in athena.*

```
wget http://db.csail.mit.edu/6.830/psql-7.4.tar.gz
tar xvzf psql-7.4.tar.gz psql-7.4
cd psql-7.4/
```

You can run the explain command by first connecting to hancock's PostgreSQL server with

```
./psql -h hancock.csail.mit.edu -p 5432 6830-2010 username
```

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/8.3/static/performance-tips.html`

We have run `VACUUM FULL ANALYZE` on all of the tables in the database, which means that all of the statistics used by PostgreSQL should be up to date.

For the following two queries, answer these questions:

a. What physical plan does Postgres use? Your answer should consist of a drawing of a query tree annotated with the access method types and join algorithms.

*Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.t

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

c. What does Postgres 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 Postgres's estimates differ from the actual intermediate result cardinalities?

f. Given the tables and indices we have created, do you think Postgres has 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?

1. [10 points]: Query 1:

```
SELECT r.name as researcher_name, o1.name as researcher_org, o2.name as grant_org
FROM researchers as r, grants as g, grant_researchers as gr, orgs as o1, orgs as o2
WHERE r.id = gr.researcherid
AND gr.grantid = g.id
AND r.org = 8
AND g.org = 0
AND r.org = o1.id
AND g.org = o2.id;
```

2. [10 points]: Query 2:

```
SELECT f.name as field_name, g.title as title
FROM fields as f, grants as g, grant_programs as gp, grant_fields gf, researchers r
WHERE g.id = gf.grantid
AND g.pi = r.id
AND r.org != 4
AND gf.fieldid = f.id
AND gp.grantid = g.id
AND gp.programid > 400;
```

3. [15 points]:

Use `EXPLAIN` to find the query plans (including access methods types and join algorithms) for the following four queries, draw the query tree for each of them, and answer the questions below.

1. ```
   SELECT managers.name
   FROM managers, grants
   WHERE grants.manager = managers.id
   AND grants.started > '1980-01-01';
   ```
2. ```
   SELECT managers.name
   FROM managers, grants
   WHERE grants.manager = managers.id
   AND grants.started > '2009-01-01';
   ```
3. ```
   SELECT managers.name
   FROM managers, grants
   WHERE grants.manager = managers.id
   AND grants.started > '2009-11-01';
   ```
   ```

4.
```
SELECT managers.name
FROM managers, grants
WHERE grants.manager = managers.id
AND grants.started > '2010-01-01';
```

a. Draw the query plans.

b. You may notice that the query plans for the four queries above are different, even though they all have the same general form. Explain in a sentence or two for each transition why the query plan changes.

c. More generally, if we're running the queries above where we vary the date d after the '$>$', for what value of d does Postgres switch between queries 1 and 2? *Note: Please write the last date that query 1 executes. This means that query 2 will execute in the following day of the year. The format is 'yyyy-mm-dd'.*

d. Why does it decide to switch plans at that value? Please be as quantitative as possible in your explanation.

e. Suppose the crossover point (from the previous question) between queries 1 and 2 is d_0. Compare the actual running times corresponding to the two alternative query plans at the crossover point. How much do they differ? 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`; you can stop redirection just by issuing the `\o` command without a file name. You may also wish to run each query several times, throwing out the first time (which may be longer as data is loaded into the buffer pool) and averaging the remaining runs.

f. Based on your answers to the previous two problems, is d_0 actually the best place to switch plans, or is it an overestimate/underestimate of the best crossover point? If d_0 is not the actual crossover point, can you estimate the actual best crossover point without running queries against the database? State assumptions underlying your answer, if any.

Part 2 - Access Methods

Ben Bitdiddle is building a database to catatog the sport he invented, Y2K-Ball. Ben created this sport at the start of the new millennium. The rules are awfully confusing so he is not going to describe them to you. He has been writing information about players, teams, and matchups throughout the years in his database, but needs your help.

Ben has four tables:

- A table of matchups that describe a game between two teams `M`, with schema <*hometeamid int, awayteamid int, homescore int, awayscore int, description char(100), t timestamp, num_injuries int*> and |`M`| records in it,
- A table with metadata about teams in it `T`, with schema <*id int, name char(30), city char(30), mascot char(20), captainid int*> and |`T`| records in it, and
- A many-to-many mapping table specifying which players are in a particular team `PT`, with schema <*pid int, tid int*>, and
- A table containing a list of players and the metadata about a player `P`, with schema <*id int, name char(50), nickname char(20), favorite_food char(30), background char(200)*> and |`P`| records in it.

You may assume ints, floats, and timestamps are 4 bytes, and characters are 1 byte. Disk pages are 10 kilobytes. In the following three questions, we provide you with a physical database design and a query workload, and ask you to explain why the database system chooses a particular plan or what plan you think the database would select.

Ben's database supports heap files and B+Trees (clustered and unclustered.)

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 over CPU time.

For each query, assume that no pages are already in memory. Assume that during the query, any page that has been read remains in the buffer pool until the end of the query (no pages are evicted).

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

Statistic	Value
Number of Players `|P|`	2×10^5
Number of Teams `|T|`	10^4
Number of Matchups `|M|`	2×10^6
First Matchup	1/1/2000
Distinct Favorite Foods	20
Distinct Cities	200

In the absence of other information, assume that attribute values are uniformly distributed (e.g., that there are approximately the same number of each favorite food type, etc.)

Ben creates an unclustered B+Tree over `P.favorite_food`. He then runs the query `SELECT * FROM P WHERE favorite_food = 'candy'`. He finds that when he runs this query using the `EXPLAIN` command, the database does not use the B+Tree he just created.

4. [5 points]: Why not? What does the database probably do instead? How much time is devoted in I/O in the plan the database takes? How much time is devoted to I/O if the database used the unclustered B+ Tree?

5. [5 points]: Would the results change if he used a clustered B+ Tree rather than unclustered? Justify your answer.

Ben hates when his favorite team, The Rumblers, loses at home. He wants to find out which players contributed to beating his team at home, so he can seek vengeance. (Note: multiple players can show up more than once). You can also assume that The Rumblers win at home about 50% of the time. The Rumblers have been his favorite team since January 1st, 2005, so he only cares about matchups after then.

```
SELECT P.name FROM P, T, PT, M
WHERE T.name = 'The Rumblers'
AND M.hometeamid = T.id
AND M.awayteamid = PT.tid
AND M.awayscore > M.homescore
AND M.t > '2005-01-01'
AND P.id = PT.pid .
```

For the following questions, limit your plans to left-deep joins.

6. [5 points]: Draw the query plan you think the database would use for this query, supposing that only heap files were available.

7. [5 points]: Draw the query plan you think the database would use for this query given that Ben had created a clustered B+Tree over `P.id`, a clustered B+Tree over `M.hometeamid`, a clustered B+Tree over `T.name`, and a clustered B+Tree over `PT.tid`.

8. [5 points]: Estimate the cost difference between the above two plans.

9. [5 points]: Now suppose that the indices were not clustered (assume the keys are randomly distributed throughout the heap file.) Estimate the cost of the plan from Q7 if the indices are not clustered.

Part 3 – Schema Design and Query Execution

Mary Monster has been creating a game with monsters. Sometimes monsters attack one another. Mary wants to keep track of which monsters fight and the place they fought. She wants to store the following information in her database.

1. It must record the name, type, birthday, and meanness of a monster.
2. It must record the place an attack occurs and who participates in the attack. An attack can happen only in one place. Many monsters can participate in an attack, but only 1 monster can win.
3. It must record the name of the place where the attack occurred and the latitude/longitude pair of the location

Mary also knows that the meanness of a monster is based on the type of the monster. Also, a monster can have only one type, but there may be many monsters of a specific type.

Not having taken 6.830, Mary isn't sure how to use this information to generate a schema, so she decides to make something up. Specifically, she represents her database with two tables; a "monsters" table which includes information about a monster such as name, type etc., and a table of "attacks" in which the fights occurred and there was a winner.

monsters : {`m_id | m_name | m_type | m_bday | m_meanness` }

attacks : {`a_id | place_name | time | participant_id REFS monsters.m_id| winner_id REFS monsters.m_id | place_lat | place_long` }

The `attacks` table includes one row for each monster that participates in an attack and a monster may appear multiple times if it was in multiple fights. Additionally, an attack record may be removed (due to diplomatic reasons), or updated (if an error is discovered).

This representation works fine for a few months, but as Mary's database grows, she finds it is very hard to maintain, particularly as the monsters have more and more attacks.

10. [5 points]: Based on the information about Mary's database, list three problems that Mary's schema may cause as her database grows and is subject to lots of inserts and deletes. (Please state any assumptions you made about the data to come to these conclusions).

11. [5 points]: Draw an Entity Relationship (ER) diagram for the data Mary wants to store.

12. [6 points]: Use your ER diagram to derive good schema for Mary's database that addresses the problems you mentioned above. You may want to introduce different or additional attributes (e.g., `id` fields) than those Mary chose to use in her initial schema. Justify your answer based on how this new schema addresses the problems related to insertions and deletions and other factors such as overall size.

13. [4 points]: Explain the tradeoffs between the schema Mary suggested and the one you designed. In what kinds of scenarios is Mary's schema better/worse than your schema? (You may want to think about reading/analyzing data versus updating/adding data. What if monsters stopped attacking each other and wanted to learn about the history of attacks?)

6.830 / 6.814 Database Systems

Fall 2010