Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2015, Prakash**

Homework 4: Query Processing, Query Optimization
<u>(due March 16th, 2015, 4:00pm, in class—hard-copy please)</u>

Reminders:

a. Out of 100 points. Contains 6 pages.
b. Rough time-estimates: 2~4 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Lead TA for this HW: Elaheh Raisi.

Q1. Relational Operators [25 points]

Consider the natural join $R \bowtie S$ of relations R(a, b, c) and S(c, d, e) given the following:

- Relation R contains NR = 20,000 tuples with 25 tuples per page.
- Relation S has NS = 45,000 tuples with 30 tuples per page.
- B = 601 buffer pages are available.

Assume that both relations are stored as simple heap files and that neither relation has any indexes built on it.

Q1.1. (5 points) Which is the smaller relation i.e. which is the one with the fewer number of pages? Write down the number of pages in the smaller relation (call it M) and the number of pages in the larger relation (call it N).

Q1.2. (20=5x4 points) Find the costs (in terms of disk accesses) of the following joins (make sure you also write the formulae you use for each one in terms of M, N, and B):

A. Nested Loops Join (page 454 of textbook or check slides)
B. Block nested loops Join (page 455 of textbook or check slides)
C. Sort-Merge Join (page 461 of textbook or check slides)
D. Hash Join (page 464 of textbook or check slides)

Q2. Query Optimization (Pen-n-Paper) [25 points]

Consider the following schema:
Sailors(sid, sname, rating, age)

Reserves(sid, did, day) Boats(bid, bname, size)

Reserves.sid is a foreign key to Sailors and Reserves.bid is a foreign key to Boats.bid. We are given the following information about the database: Reserves contains 10,000 records with 40 records per page. Sailors contains 1000 records with 20 records per page. Boats contains 100 records with 10 records per page.

There are 50 values for Reserves.bid.
There are 10 values for Sailors.rating (1..10).
There are 10 values for Boat.size
There are 500 values for Reserves.day.

Consider the following queries:

Query 1:
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid=R.sid AND R.bid = B.bid

Query 2:
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid=R.sid AND R.bid = B.bid AND B.size>5 AND R.day='July 4, 2003'

Q2.1. (5 points) Assuming uniform distribution of values and column independence, estimate the number of tuples returned by *Query 2*. Also assume you are given that the size of *Query 1* is 10^4 tuples. (*Hint*: Estimate the selectivities of Boat.size>5 and Reserves.day='July 4, 2003')

Q2.2. (5 points) Draw all possible left-deep join query trees for Query 1.

Q2.3. (15 points) For the first join in each query plan you get in Q2.2 (i.e. the join at the bottom of the tree), what join algorithm would work best (i.e. cost the least)? Assume that you have 50 pages of buffer memory. There are no indexes; so indexed nested loop is not an option. Consider the Page Oriented Nested Join (PNJ), Block-NJ, Sort-Merge-J, and Hash-J. Make sure you also write down the formula you use for each case (check slides).

Q3. Query Optimization (Stats and Range Queries) [20 points]

Consider the following two relations, used for book rating:

- Book (ISBN,title,author,year,publisher,image_s,image_m,image_l)

- User_Book_rating (user_id,ISBN, book_rating)
- Users (id, location,age)

The above information was gathered in a 4-week crawl from the book-crossing.com community website where users rate books. The Book table has 'ISBN', 'Book title', 'Book author', 'Year of publication', 'Publisher', 'Image-URL-S' which is the URL for small size image of the book (like a thumbnail), 'Image-URL-M' which indicates the URL for medium size image, and 'Image-URL-L' which is the URL for the large size image; The content of 'Book' table was obtained from Amazon Web Service. In the 'User_Book_rating' table we have the 'user_id', which is the id of the user who rates a Book, 'ISBN' of the book and the 'book rating' of the user for the book. Each user can rate multiple books. The 'Users' table has the 'id', 'location' and 'age' of the Users who rate the books.

Assume there are no existing indexes on these tables and relations are stored as simple heap files.

These three tables are stored in cs4604.cs.vt.edu server. This is the first time you will be accessing the PostgreSQL server, so refer to the guidelines here (account information etc.):

http://courses.cs.vt.edu/~cs4604/Spring15/project/postgresql.html

Use the following commands to copy the tables to your private database:

- pg_dump -U YOUR-PID -t Book cs4604s15 | psql -d YOUR-PID
- pg_dump -U YOUR-PID -t User_Book_rating cs4604s15 | psql -d YOUR- PID
- pg_dump -U YOUR-PID -t Users cs4604s15 | psql -d YOUR-PID

Sanity Check: run the following two statements and verify the output.

- select count(*) from book; // output – count = 250012
- select count(*) from user_book_rating; // output – count = 1149769
- select count(*) from users; // output – count = 278683

For this question, it may help to familiarize you with the pg_class and pg_stats tables, provided by PostgreSQL as part of their catalog. Please see the links below:

http://www.postgresql.org/docs/8.4/static/view-pg-stats.html

http://www.postgresql.org/docs/8.4/static/catalog-pg-class.html

Now answer the following questions:

Q3.1. (6 points) Using a single SQL SELECT query on the 'pg_class' table, for each relation 'Book', 'User_Book_rating' and 'Users' to find (a) the number of rows, (b) the number of attributes and (c) if the relation has a primary key. Write down the SQL query you used and also paste the output.

Q3.2. (8 points) We want to find the top three years in which most publications happen. There are two ways to do that:

A. (4 points) Write a SQL query that uses the book table find the top three years in which most publications happen. Paste the output too when you run the query.

B. (4 points) Now write a SQL query that uses only the 'pg_stats' table instead to find the top three years in which most publications happen. Again, paste the output.

Note: to get the first three elements of a 'list' type (e.g. 'mylist'), you can use 'mylist[1:3]', and to get the first three elements of 'anyarray' type (e.g. 'myarry'), you can use (myarray::varchar::varchar[])[1:3] see this link:

http://www.postgresql.org/docs/8.4/static/view-pg-stats.html

Q3.3. (6 points) Consider the following query that retrieves all the books published after 1880. *Note:* You do not need to run anything on the server for this part.

```
Query 3:    SELECT *
            FROM book
            WHERE year > 1880;
```

A. (3 points) Recall there is no index on year attribute. Will the optimizer use an Index scan or a simple File Sequential Scan? Explain in only 1 line.

B. (3 points) What would happen if someone creates a non-clustered B+-Tree index on 'year' attribute? Will it help to speed up the retrieval? Again explain in only 1.

Q4. Query Optimization (Joins) [30 points]

Again consider the same three relations and the database tables given in Q3 before. In this question, we want to explore the effect of creating indexes on join optimization. Again start with the assumption that there are no indexes on the tables.

It will help to familiarize yourself with the EXPLAIN and ANALYZE commands of PostgreSQL. Please visit the links given at the end for the same and study how to run it on a given SQL query and what output these commands return.

Q4.1. (5 points) Consider the same Query 3 from Q3.3 in the previous page. Run the 'explain analyze' command sequence on it and answer the following questions.

A. (1 point) Copy-paste the output of using EXPLAIN ANALYZE on Query 3.

B. (2 points) What is the estimated result cardinality of Query 3?

C. (2 points) What is the number of rows Query 3 actually returns?

Note: No need to paste the actual output rows of Query 3 since it will be too large. We just want the output of running EXPLAIN ANALYZE.

Q4.2 (10 points) Consider the following query

Query 4:
```
SELECT book.title,user_book_rating.book_rating
FROM user_book_rating,book,users
WHERE book.isbn = user_book_rating.isbn
        AND users.id = user_book_rating.user_id
        AND book.year >= 2000
        AND users.age = 20
        AND user_book_rating.book_rating = 7;
```

Run the 'explain analyze' command sequence on Query 4 and get the query plan returned by the optimizer to answer the following questions.

A. (2 points) Explain what does Query 4 do?

B. (1 points) Copy-paste the output of using EXPLAIN ANALYZE on Query 4.

C. (5 points) Draw the query plan returned using tree and relational algebra notations as given in lecture slides.

D. (2 points) Report the estimated result cardinality, actual result cardinality (i.e. the true number of rows returned) and the total runtime of executing Query 4.

Note: Again, we do not want the actual output rows of Query 4 itself since it will be too large.

Q4.3 (5 points) For the Query 4, which attributes for the three tables you think should have index? create index on these three tables for the proper attributes. Write the SQL queries you used to create these indexes (give these indexes any names you want).

Q4.4 (10 points) Update the statistics using the VACCUM and ANALYZE commands. Now run the EXPLAIN ANALYZE command again on the Query 4 given in Q4.2 and get the new query plan returned by the optimizer to answer the following questions.

A. (2 points) Copy-paste the output of using EXPLAIN ANALYZE on Query 4.

B. (6 points) Draw the query plan returned using tree and relational algebra notations as given in lecture slides.

C. (2 points) Report the actual runtime of Query 4 now, and compare it with your previous answer in Q4.2(C). Was it worth constructing the indexes?

Hints:

1. Check the statistics collected by PostgreSQL: http://www.postgresql.org/docs/8.4/static/planner-stats.html

2. How to use EXPLAIN command and understand its output: http://www.postgresql.org/docs/8.4/static/sql-explain.html http://www.postgresql.org/docs/8.4/static/performance-tips.html