Database System Implementation- CSE 507

Homework 2

Due date: February 18, 2016 11:59pm

Instructions:

- All submissions must be made through usebackpack site for this course (https://www.usebackpack.com/iiitd/w2016/cse507)
- Only one submission per team would be considered and graded. It would be assumed that all members of the team have participated equally and same score would be given to all members of the team.
- Your submission should have names of all the members of your team.
- Any assumptions made while solving the problem should be clearly stated in the solution.
- Question 3 and Question 4 are for teams of size 3. These questions will not be graded for teams for size 2 or less.
- For this homework, you would be working with PostgreSQL. Please install this software on your computers before proceeding. This software is available as a free download at this website: http://www.postgresql.org/download/
- Also we would be using "EXPLAIN" and "ANALYZE" statements available in PostgreSQL extensively in this homework. Please refer to their websites www.postgresql.org/docs/current/static/sql-explain.html, http://www.postgresql.org/docs/9.1/static/using-explain.html and http://www.postgresql.org/docs/current/static/sql-analyze.html to familiarize yourselves with the syntax of these statements.
- Remember to VACCUM ANALYZE your table after inserting all the records: http://www.postgresql.org/docs/9.1/static/sql-vacuum.html
- In fact from time to time keep running the ANALYZE command;

Question 1 (Programming based question)

Part A: Range Query and Index (15 points)

1. Note that there is a secondary index on the VOTES column of MOVIE table.
2. Write SQL statements for the following VOTES based range queries.
   - Q1: List the movies with votes between 10 and 100.
   - Q2: List the movies with votes between 20 and 59000.
3. Run the explain plan for each Q1 and Q2 using the instructions given above.
4. Submit the explain plan output of above two queries, explain these plans and compare them.
5. Explain why the index on VOTES is not always used for range queries based on VOTES attribute.

Part B: Selectivity (15 points)

1. Note there is a secondary index on the VOTES and the YR column of MOVIE table.
2. Consider following range queries on VOTES and YR columns.
   - Q1: SELECT title FROM movie WHERE votes < 10000;
   - Q2: SELECT title FROM movie WHERE votes between 60000 and 100000;
   - Q3: SELECT title FROM movie WHERE yr between 1900 and 1990;
   - Q4: SELECT title FROM movie WHERE yr between 1890 and 1900;
3. Run the explain plan for Q1, Q2, Q3 and Q4 and Submit the output.
4. Compare the number of tuples selected from each query. Which query had lower selectivity (i.e. returns fewer rows)?
5. Is the index on VOTES used for Q1 and Q2 queries? Give an intuitive explanation of the observed results.

6. Is the index on YR used for both Q3 and Q4 queries? Give an intuitive explanation of the observed results.

Part C: String Matching and Index (15 points)

1. Note that there is a secondary index on the NAME column of ACTOR table.
2. Consider following SQL queries.
   - Q1: SELECT * FROM actor WHERE name LIKE 'B%';
   - Q2: SELECT * FROM actor WHERE name LIKE '%b';
   - Q3: SELECT * FROM actor WHERE substr(name, 1, 1) = 'B';
3. Run the explain plan for each Q1, Q2 and Q3, and submit the output.
4. Compare the execution plans of query Q1 and Q2. What's the difference? Explain.
5. Compare the execution plans of query Q1 and Q3. Explain difference, Wwy the index on NAME is not always used for queries based on NAME attribute.

Part D: Aggregate Operations (15 points)

1. Consider following two equivalent SQL queries.
   - Q1: SELECT title FROM movie WHERE votes <= (SELECT MIN(votes) FROM movie);
   - Q2: SELECT title FROM movie WHERE votes <= ALL (SELECT votes FROM movie);
2. Run the explain plan for each Q1 and Q2 and submit the output.
3. Is there any difference in the execution plans for Q1 and Q2? Which query is more efficient to execute? Explain.

Part E: Alternate Join Strategies (20 points)

1. Recall that there are indices on CASTING.ACTORID, CASTING.ORD and ACTOR.ID.
2. Run the ANALYZE command;
3. Consider following three SQL queries for retrieving name of the movie and id's of their lead actors under different conditions.

   Q1: SELECT m.title, c.actorid FROM casting c, movie m WHERE m.id=c.movieid;

   Q2: SELECT m.title, c.actorid FROM casting c, movie m WHERE m.id=c.movieid AND c.ord = 21;

   Q3: SELECT m.title, c.actorid FROM casting c, movie m WHERE m.id=c.movieid AND AND m.yr between 1890 and 1900;
4. Run the explain plan for each Q1, Q2 and Q3 and submit the output.
5. Compare and contrast three execution plans for Q1, Q2 and Q3 in terms of join algorithm and indexes used? Is there any difference? Why?

Question 2: Rank order the four join strategies (J1 through J4 on Page 689—690 of the textbook) for the following join operation:

Join table DEPARTMENT and EMPLOYEE **on the condition** Mgr_ssn = Ssn

Justify your answer using the cost models discussed on Page 716—717 of the textbook.

Given:

Number of records in DEPARTMENT table = 125

Total number of disk blocks for DEPARTMENT table = 13

Number of records in EMPLOYEE table = 10000

Total number of disk blocks for the EMPLOYEE table = 2000

Assume that you are given the COMPANY relational database schema shown in Figure 3.6 (page 72) of the textbook. Further, you have a primary index on ssn of EMPLOYEE with x_ssn=4 levels and secondary index on Mgr_ssn of DEPARTMENT with selection cardinality s_Mgr-ssn =1 and levels x_Mgr-ssn = 2. Also, assume that join selectivity of the given condition is (1/|DEPARTMENT|) = 1/125. Also assume that blocking factor for the resulting join file is 4 records per block. In case of Nested loop algorithm, assume that we have three memory buffers.

Question 3: Develop cost functions for MAX and INTESECTION operations as discussed in sections 19.4 and 19.5 of the textbook. While developing the cost models it is vital to show all the intermediate steps that lead to final equation. Please provide an intuitive explanation for the choices and assumptions made.

Question 4: Consider the following SQL query for the COMPANY relational database schema shown in Figure 3.6 (page 72) of the textbook.

Select Pnumber, Pname, Lname, Ssn

From PROJECT, WORKS_ON, EMPLOYEE

Where Pnumber=Pno **AND** Ssn=Essn **AND** Hours>20.0

(a) Draw an initial (canonical tree) query tree where each join operation required by the query is represented with a Cartesian product. A sample canonical tree for a certain query is given in Figure 19.5(a). (Ref: Page 704 of the textbook).

(b) Draw two other query trees (different from (a)) that represent the above query. Under what circumstances would you use each of your query trees?

(c) Optimize the query tree of part (a) using the heuristic algebraic optimization algorithm given on Page 708 in textbook. Please show all the intermediate steps with brief explanations.