Homework 2: Query Processing/Optimization, Transactions/Recovery
(due February 25th, 2019, 2:30pm, in class—hard-copy please)

Reminders:
- Out of 100 points. Contains 7 pages.
- Rough time-estimates: 5-8 hours.
- 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.
- There could be more than one correct answer. We shall accept them all.
- Whenever you are making an assumption, please state it clearly.
- Each HW has to be done individually, without taking any help from non-class resources (e.g., websites etc).

Q1. Sorting [20 points]
Suppose you have a file with \( N = 5 \times 10^7 \) pages.

Q1.1. (5 points) What is the total I/O cost of sorting the file using the two-way merge sort (with three buffer pages)?

Now suppose you have \( B = 129 \) buffer pages. Answer the following questions using the general external sorting algorithm outlined in section 13.3 of the textbook (page 424). Please write the formula you used in calculating the answers.

Q1.2. (5 points) How many runs will you produce in the first pass?

Q1.3. (5 points) Now assume that we have a disk with an average seek time of 10ms, average rotation delay of 5ms and a transfer time of 1ms for each page. Assuming the cost of reading/writing a page is the sum of those values (i.e. 16ms) and do not distinguish between sequential and random disk-access – any access is 16ms, what is the total running time to sort the file?

Q1.4. (5 points) With 129 buffer pages, what’s the maximum size of the file (in number of pages) that we can sort with 3 passes?

Q2. Query Optimization (Pen-n-Paper) [20 points]
Consider the following schema:
Sailors(sid, sname, rating, age) Reserve(sid, did, day) Boats(bid, bname, size)
Reserver.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

**Query 2:**
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B

Q2.1. (4 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. (4 points) Draw all possible left-deep join query trees for Query 1.

Q2.3. (12 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 (Hands-on) [25 points]**
Consider the following two relations:
employees(emp_no, birth_date, first_name, last_name, gender, hire_date);
emp_salaries(emp_no, salary, from_date, to_date);
The employees table has the id of the employee, birth date, first name, last name, gender (M or F) and hire_date. Each employee could have many salaries through their careers. These salaries are present in the emp_salaries table that has the emp_no, the salary and the date range he had this particular salary.

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

We have created sample instances of these two tables on the cs4604.cs.vt.edu server. This is the first time you will be accessing the PostgreSQL server, so refer to the guidelines (do not wait till the HW due date to check your access to the server!):

Use the following commands on the command prompt (before logging in to psql) to copy the tables to your private database:
- `pg_dump -U YOUR-PID -t employees cs5614s19 | psql -d YOUR-PID`
- `pg_dump -U YOUR-PID -t emp_salaries cs5614s19 | psql -d YOUR-PID`

**Sanity Check**: run the following two statements and verify the output.
- `select count(*) from employees;` // output count = 300024
- `select count(*) from emp_salaries;` // output count = 2844047

For this question, it may help to familiarize yourselves 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. (5 points) Using a single SQL SELECT query on the ‘pg_class’ table, find if the relations have primary keys, the number of attributes and the number of disks pages occupied by each relation (employees and emp_salaries). Write down the SQL query you used and also paste the output.

Q3.2. (5x3=15 points) We want to find if the employees are mostly males or females, using the ‘gender’ attribute in the ‘employees’ table. We will do this via two different ways.

A. Write a SQL query that uses the ‘employees’ table to analyze and find the most frequent gender between the males and females employees. Paste the output too.
B. Now write a SQL query that uses only the ‘pg_stats’ table instead to analyze and find the most frequent gender. Again paste the output.

C. Use the EXPLAIN ANALYZE command to find out the total runtime and analyze the execution plan that the PostgreSQL planner generates for the two queries from parts A and B. Also compare the outputs you get in parts A and B. What do you observe?

Q3.3. (2 points) Consider the following query that retrieves every salary that was granted after Feb 9, 1952. Note: You do not need to run anything on the server for this part.

select *
from emp_salaries
where from_date > '1953-09-02';

Will a non-clustered B+-Tree index on the ‘from_date’ attribute help to speed up the retrieval? Explain in only 1 line.

Q3.4. (3 points) Create an index on your database that will decrease the runtime of the query in Q3.3. Write the index and the time improvement.

Q4: Concurrency Control and Deadlocks [15 points]

Q4.1. (5 points) Consider the schedule below. R(.) stands for Read and W(.) for write.

<table>
<thead>
<tr>
<th>time</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>t8</th>
<th>t9</th>
<th>t10</th>
<th>t11</th>
<th>t12</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R(C)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R(B)</td>
<td>W(B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>R(A)</td>
<td>W(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R(C)</td>
<td>W(C)</td>
<td>R(B)</td>
<td>W(B)</td>
</tr>
</tbody>
</table>

Answer the following questions.

a. (1 point) Is the schedule serial?

b. (2 points) Give the dependency graph. Is the schedule Conflict-Serializable? If yes, give the equivalent serial schedule, else explain briefly.

c. (2 points) Could this schedule have been produced by 2PL? Either show the sequence of locks and unlocks obeying 2PL or explain why not.

Q4.2. (4 points) Consider the schedule below. R(.) stands for Read and W(.) for write.
Answer the following questions:

a. (2 points) Could this schedule have been produced by 2PL? Either show a valid sequence of locks and unlocks obeying 2PL or explain why not.

b. (2 points) Could this schedule have been produced by strict-2PL? Either show a valid sequence of locks and unlocks obeying strict-2PL or explain why not.

Q4.3. (6 points) Consider the two lock schedules below. S(.) stands for a shared lock and X(.) denotes an exclusive lock.
Answer the following questions:

a. (2 points) For both schedules, which locks will be granted or blocked by the lock manager?
b. (2 points) Give the waits-for graphs for both the schedules.
c. (2 points) Is there a deadlock at the end of Schedule 1? What about Schedule 2? For each explain in 1 line.

Q5: Logging and Recovery [20 points]

Consider the log below. The records are of the form: (t-id, object-id, old-value, new-value).

Assumptions: a. The PrevLSN has been omitted (it’s easy to figure it out yourself); b. for simplicity we assume that A, B, C and D each represents a page;

1. (T1, start)
2. (T1, A, 45, 10)
3. (T2, start)
4. (T2, B, 5, 10)
5. (T2, C, 35, 10)
6. (T1, D, 15, 5)
7. (T1, commit)
8. (T3, start)
9. (T3, A, 10, 15)
10. (T2, D, 5, 20)
11. (begin checkpoint, end checkpoint)
12. (T2, commit)
13. (T4, start)
14. (T4, D, 20, 30)
15. (T3, C, 10, 15)
16. (T3, commit)
17. (T4, commit)

Q5.1. (3x4=12 points) What are the values of pages A, B, C and D in the buffer pool after recovery? Also specify which transactions have been ReDo(ne) and which transactions have been UnDo(ne). You are not required to show the details of the intermediate step.

a. If the system crashes just before line 6 is written to the stable storage?
b. If the system crashes just before line 10 is written to the stable storage?
c. If the system crashes just before line 13 is written to the stable storage?
d. If the system crashes just before line 17 is written to the stable storage?
Q5.2. (8 points) Assume only the crash as listed in Q5.1c has really happened and a recovery has then been performed, and the dirty pages caused by T1 have been flushed to disk before line 8.

a. (2 points) Show the content of the transaction table and the dirty page table at both the beginning and the end of the Analysis phase. You may assume that both the transaction table and dirty page table are empty at the beginning of line 1 of the log.

b. (6 points) Show the content of log when the recovery has completed.