CS4320 Prelim Exam

October 29th, 2019

(60 minutes working time)

Name: _____ Cornell NETID: _____

I understand and will adhere to the Cornell Code of Academic Integrity.

Signature

Maximum number of points possible: 50. This exam counts for 25% of your overall grade. Questions vary in difficulty. Do not get stuck on one question.

In all problems, whenever you think a problem is underspecified, make assumptions and clearly state them.

Good luck!

Note – you have 60 minutes working time for this exam, NOT 2 hours as on some other prelims.

Prelim Part A) SQL Queries. (20 points)

Consider the database schema created by the following SQL commands:

```
CREATE TABLE Sailors (sid integer PRIMARY KEY,
sname varchar(20), rating integer, age real);

CREATE TABLE Boats (bid integer PRIMARY KEY,
bname varchar(20), color varchar(20));

CREATE TABLE Reserves(sid integer, bid integer,
day date, PRIMARY KEY (sid, bid, day),
FOREIGN KEY (sid) references Sailors (sid),
FOREIGN KEY (bid) references Boats (bid));
```

The database stores information on sailors (table `Sailors`), on boats (table `Boats`), and reservations for boats by sailors (table `Reserves`). Note that no duplicate elimination is necessary for your solution queries (i.e., no need to use the SQL keyword `DISTINCT`).

A.1) Write an SQL query retrieving the boats (attribute `bid`) for which no sailor has made more reservations than the sailor with ID 1 (attribute `sid`). E.g., if sailor number one has made three reservations for one specific boat while sailor number two made four for it then that boat does not appear in the result. (5 points)

Select bid from Boats B where not exists (select * from Sailors S where sid <> 1 and (select count(*) from reserves R where S.sid = R.sid and B.bid = R.bid) > (select count(*) from reserves R2 where R2.sid = 1 and R2.bid = B.bid));

A.2) Write an SQL query retrieving the names (attribute `sname`) of all sailors whose age (attribute `age`) is at most half the average age among all sailors with the same rating (attribute `rating`). E.g., if sailor one has rating five and age 20 then that sailor appears in the result if the average age for that rating is at least 40. (5 points)

Select sname from Sailors S where S.age * 2 <= (select avg(age) from Sailors S2 where S.rating = S2.rating);

A.3) Write an SQL query retrieving the ratings (attribute `rating`) for which all sailors with that rating have an age (attribute `age`) below twenty and have made at least two reservations. E.g., if a sailor with rating five made less than two reservations then rating five does not appear in the result. You only need to consider ratings for which at least one sailor is registered in the database. (5 points)

Select rating from Sailors S group by rating having every (S.age < 20) and every ((select count(*) from reserves R where R.sid = S.sid) >= 2);

A.4) Write an SQL query retrieving for each sailor the number of reservations made for red boats (attribute `color`). The result contains two columns, the sailor ID (attribute `sid`) and the number of reservations. Sailors who made no reservations for red boats must also appear in the result (with count zero). (5 points)

Select S.sid, count(R2.bid) from sailors S left outer join (select R.sid, R.bid from reserves R, boats B where B.color = 'red' and R.bid = B.bid) as R2 on S.sid = R2.sid group by S.sid;

Prelim Part B) Indexing. (10 points)

Assume we store 80,000 sailors in the sailors table, defined as in Part A). We build a B+ tree index on the rating attribute. The B+ tree has four levels (including root and leaf level) that are all stored on hard disk (no part of the index is cached in main memory). We store data entries of type "Alternative 2" (data entries are pairs of index keys and record IDs).

Assume that each row of the sailors table consumes 100 bytes, each rating field consumes four bytes and each record ID consumes four bytes as well. Each page stores 800 bytes and we assume that pages are always full (e.g., we store the maximal number of data entries in each index leaf node).

We use the index to retrieve all sailors with rating five. Assume that there are 10 distinct rating values and that sailors are uniformly distributed over ratings. For the following questions, use the cost model and default assumptions (e.g., with regards to the cost of using unclustered indices) seen in class.

B.1) Calculate the cost of using the index and retrieving sailors with rating five if the index is clustered. (5 points)

We have cost four for traversing the index tree from root to leaf node. As the index is clustered, we can work directly on the data and do not need to consult the index anymore. Assuming a uniform distribution, there are 80,000 * 0.1 = 8,000 sailors with rating five. Each page fits eight sailors (800 bytes / 100 bytes). Hence, we need to read 1,000 pages. In total, the cost is 1,000 + 4 = 1,004 pages.

B.2) Calculate the cost of using the index and retrieving sailors with rating five if the index is unclustered. (5 points)

Each data entry consumes eight bytes (four bytes for the key and four for the record ID). Hence, we can store 100 data entries per page. As outlined before, we assume that 8,000 sailors satisfy the condition on rating. Hence, we need to retrieve 80 index leaf pages. We need to retrieve three more pages from the index (to traverse the upper tree levels). Finally, we must count one page access per data entry (making worst case assumptions for unclustered indices, as seen in class). Hence, we have 3 + 80 + 8,000 = 8,083 page IOs.

Prelim Part C) Relational Operators. (10 points)

C.1) We join two tables (R and S) on disk with R as outer and S as inner operand. R consumes 10 pages and S consumes 100 pages. We use the block-nested loops join. How many buffer pages do we need at least (in total, considering all required buffers) such that the join has a cost of 210 page I/Os (not counting the cost of writing out the join result)? Show your calculations to justify your answer.
(5 points)

We need 10 page I/Os to read the outer operand. Hence, 200 page I/Os remain for reading the inner operand twice. Hence, one block must fit at least 5 pages (as 10/5=2). Counting one output buffer page and one input buffer for the inner relation, we therefore need 7 buffer pages in total.

C.2) We sort a table that consumes 100 pages. We use the (general) external merge sort algorithm seen in class. Exactly how many buffer pages (in total, counting all required buffers) do we need at least to sort the entire table with cost 300 page I/Os (not counting the cost of writing out the final result)? Justify your answer by showing your calculations.
(5 points)

300 page I/Os allow us to make two passes over the data (200 page I/Os for the first pass and 100 page I/Os for the second pass, as we do not count the cost of writing the final result). With 10 buffer pages, we would have 10 runs of length 10 after the first pass but could only merge nine of them at once (since we need to reserve one buffer page as output buffer). With 11 buffer pages, we would have 10 runs of length at most 11 (the last run consists of only one page) after the first pass. We can merge together all ten of them as we have 10 input buffers. Hence, we need at least 11 buffer pages.

Prelim Part D) Miscellaneous. (10 points)

D.1) We have seen how different RAID variants combine multiple hard disks to improve metrics such as read or write performance, capacity, or resilience against disk failures. RAID Level 1 uses two disks, each of which stores an identical copy of the data (data is "mirrored").

Assume that one single disk can read or write exactly 1,000 kilobytes per second. How many bytes can be read and how many bytes can be written per second with a RAID Level 1 setup? Justify your reply in one to three sentences, pointing out simplifying assumptions you make. (5 points)

We can read from both disks at the same time and therefore read 2,000 kilobytes per second. Writing speed is1,000 kilobytes (it does not increase compared to single disks) as data has to be written to both disks.

D.2) Consider a new join operator that works as follows. Given two relations (called "inner" and "outer" relation in the following) to join via an equality join condition, it first sorts the inner relation on the join column, using the (general) external merge sort algorithm seen in class. It writes the sorted relation back to hard disk.

Next, it reads the other, "outer", relation page by page. For each tuple in the outer relation, it uses binary search to find all matching tuples in the sorted inner relation on disk.

Propose a cost formula for that join operator, expressing the number of pages read and written (except for the cost of writing out the **final** result) as a function of

a) the number of pages n in the outer relation,
b) the number of tuples t per page of the outer relation,
c) the number of pages m in the inner relation, and
d) the number of pages B in the buffer space.

Assume that each tuple in the outer relation has exactly one matching tuple in the inner relation. Justify your cost formula and clearly state your assumptions.
(5 points)

We first need to count the cost of sorting the inner relation which is (ceil(log_[B-1](ceil(m/B))) +1)*2*m.
Next, we need to count the cost of reading the outer relation once which is n.
The cost of performing binary search once (on the innner relation) is log_2(m), we multiply by the number n*t of tuples in the outer relation. In total, we obtain (ceil(log_[B-1](ceil(m/B)))+1)*2*m+n+n*t*log_2(m).

CS 4320 Prelim Exam

This page will be used for grading your exam. Do not write anything on this page.

SECTION	QUESTION	SCORE	SECTION TOTAL
Part A	A.1 (Max: 5 points)		(Max: 20 points)
	A.2 (Max: 5 points)		
	A.3 (Max: 5 points)		
	A.4 (Max: 5 points)		
Part B	B.1 (Max: 5 points)		(Max: 10 points)
	B.2 (Max: 5 points)		
Part C	C.1 (Max: 5 points)		(Max: 10 points)
	C.2 (Max: 5 points)		
Part D	D.1 (Max: 5 points)		(Max: 10 points)
	D.2 (Max: 5 points)		
Total (Max: 50 points)			