Name: _____ **Roll number:** _____

CS 317: Database and Information Systems Quiz 2, 26 Oct 2018

Answer in the question paper except where indicated otherwise.
Write your roll number on the question paper and answersheet, and submit both.
Duration: 55 min (8.30 - 9.25) **Marks: 25**

1. Software RAID (without NVRAM support) requires an expensive check when recovering from a power failure. Consider RAID 1, and explain (i) what this check does, (ii) what is the need for this check, and (iii) what is its impact on mean-time to data loss compared to hardware RAID with NVRAM, and why? ...2

   Answer: The check compares every block in the two disks to see the values differ for any pair of blocks. The check is needed since one block may have been written, but not the other, or one may be partially written. A failure of a disk before the inconsistency is fixed may result in loss of data if the other copy is partially written, or was not updated and had an old value. The mean time to data loss would be lower than with NVRAM, since every power failure event can potentially lead to data loss until the recovery is completed.

2. Bulk loading techniques are very important for B^+-trees. Explain very briefly how bulk loading could be done efficiently for an R-tree. (Hint: use ideas from write-optimized indexing.) ...3

   Answer: Use the buffer tree to insert data. The buffer tree will move data to a lower level only when the buffer is full, so multiple entries will be moved to a lower level block at a time

3. Consider the query `select r.A, s.B from r, s where r.B=s.B order by r.A limit 10`. What would be a good plan for this query if `s` has an index on $s.B$ and r has a non-clustered B^+-tree index on `(A,B)`. Show the plan as an annotated tree, and explain what algorithm would be used at each node. ...5

   Answer:

   Since there is an order by r.A with a limit clause, we are best off accessing r in sorted order, and performing an indexed nested loops join on s, and stopping once we get 10 tuples (this can be done by a limit operator, but since we have not covered it it's OK if you don't mention the operator explicitly, and just say stop after 10 results).

   To get r in sorted order, we use an index-only scan on r, since both the required attributes r.A and r.B are available in the index, and the index-only scan gives the tuples in sorted order on (A,B), and thus in sorted order on (A).

4. Consider the query `select * from r, s, t where r.A=s.A and r.B=s.B and r.C=t.C`. What are all the possible interesting sort orders on `r` for this query? Explain why each of them is interesting. ...3

   Answer: r.C for the join with t; (r.A,r.B) as well as (r.B, r.A) for the join with s.

5. Finding a block in a large file that has enough space to store a new record can be expensive if it involves sequentially searching through the file. A free-space map is an array that stores (say) a byte for each page, recording the fraction of the page that is free.

   Explain how you can create an index structure that can be built on a large free-space map array, to efficiently find a page with a specified amount of free space. Explain also how the search would work. ...4

   Answer: We create an array of size $1/n$ of the main freespace map array, where the first few bits/byte store the max of the values for the main map. (2 marks)

We search sequentially in in the smaller array to find an entry with enough space, and then search the corresponding n entries in the lower level array to find a block with the required free space. We are guaranteed to find such a block. (2 marks).

We can create more levels if required.

6. The multiset version of the semijoin operator $r \ltimes_\theta s$ is defined as follows: if a tuple r_i appears n times in r, it appears n times in the result of $r \ltimes_\theta$ if there is at least one tuple s_j such that r_i and s_j together satisfy predicate θ; otherwise r_i does not appear in the result.

   Use this definition to rewrite the following nested SQL query into relational algebra:

```
select r.A, r.B from r where r.B = (
                    select sum(s.B) from s where s.A = r.A)
```

   (Hint: grouping on an attribute will be needed.)

   ...4

   Answer: $r \ltimes_{r.A=sq.A \wedge r.B=sumB} (\rho_{sq(A,sumB)}({}_A\gamma_{sum(B)}(s)))$

   Can be written without using renaming also, by renaming attributes in the groupby/aggregate using **as** clause.

   Partial marking: semijoin: 1 mark, each semijoin predicate: 1/2 mark, aggregate op: 1 mark, each argument to aggregate op: 1/2 mark.

7. The heuristic "perform selection operations as early as possible" can increase the cost if used naively with the basic join order optimization algorithm. Give a relational algebra query on the schema $r(A, B)$ and $s(B, C)$, where this problem could occur; explicitly mention any indexes that you assume are present. Also explain how the join order optimization algorithm is tweaked to handle this situation. ...4

   Answer: With schema $r(A, B)$ and $s(B, C)$ an index on $s.B$ and query $\sigma_{s.C<5}(r \bowtie_{r.B=s.B} s$, performing the selection early would result in the index on $s.B$ not being usable. Otherwise, an indexed nested loops join may have been the best option. The optimization algorithm checks specially if an input is a relation that has an index and a selection has been pushed in, and pulls out the selection.

Total Marks = 25