CS 441: Midsemester Exam, Fall 2000

Write your ID on the top left of every page.
This is a closed-book exam.
Return the first two pages with the rest of your answer script.

I. Fill the blank (if any) and complete the following sentences in the space below. ...20

1. It is / is not possible in general to have two primary indices on the same relation for different search keys because

2. Latency time / data transfer rate is affected by the decision to keep related data together on a disk because

3. I would / would-not recommend a hashed file organization for a relation where the hash function is computed on the primary key attribute of the relation because

4. An example of a database application in which the pointer method of representing variable-length records is preferable to the reserved-space method is

   because

5. When doing merge–join, the preferred buffer management algorithm is

6. If L_i and L_j are leaf nodes in a B+ tree index and $i > j$, then every search key value in L_i is ________________________ every search key value in L_j because

7. A significant advantage of a B+ tree index over an extendible hashing index for an attribute A of a relation R is

8. A significant advantage of an extendible hashing index over a B+ tree index for an attribute A of a relation R is

9. The query processing cost of joining three relations is always/never/sometimes smaller than that for joining four relations

10. If I could recommend one "change" to this course, it would be

II. A large data warehouse is being designed to be continuously available, that is, without *any* downtime. You are asked to choose between (a) mirrored disks and (b) block interleaved distributed parity RAID. ...10

1. which would you choose and why?

2. would your answer change if you are told, in addition, that the warehouse is updated very rarely.

III. Consider the following self-explanatory relational database schema: ...40

`Student (ID, name, dept-no, hostel-no, ...)` (ID is a key)
`Department (dept-no, building, ...)` (dept-no is a key)
`Hostel (hostel-no, warden-name, ...)` (hostel-no is a key)

1. Write a relational algebra expression that returns each student's ID along with the building for that student's department.

2. Write an SQL query that returns the same result.

3. Suppose there are 5000 students, 100 departments, and 10 hostels, and suppose that in the underlying storage manager for these relations, 10 student tuples can fit in one block, 25 department tuples can fit in one block, and 15 hostel tuples fit in one block. If a straightforward block nested loop join algorithm is used to compute the query from part (2), approximately how many disk block reads can be expected?

   You may assume: (i) the relations are not sorted and there are no indices. (ii) relation `student` is used as the "outer" relation in the nested loop join algorithm. (iii) a block is read into the buffer pool every time it is needed, and blocks are pinned (fixed) while they are in active use. (In other words, you needn't consider the size of the buffer pool or the replacement strategy in computing your answer.) Show your computation, and state any additional assumptions that you make.

4. Use the same scenario as part (3), but drop assumption (i) and assume in its place: (ia) Relation `Student` is sorted on attribute `Student.dept-no.` and (ib) Relation `Department` has a dense index on attribute `Department.dept-no.` If the nested loop join algorithm capitalizes on these new assumptions in the obvious way, approximately how many block reads can be expected? Show your computation, and state any additional assumptions that you make.

5. Use the same scenario as part (4), but assume that a sort-merge join algorithm is used to compute the query. Approximately how many block reads can be expected? Show your computation, and state any additional assumptions that you make.

6. Use the same scenario in (3) but assume that you have a buffer pool with 4 blocks. How would your answer to (3) change?

7. Suppose the most often executed query is "which students stay in hostel n_1 through n_2.

   Consider two different requirements: (a) minimize additional storage costs, and (b) minimize processing cost.

   For each of these requirements how would you store the `Student` relation (also indicate which indices you would maintain, if any)?

8. Two such queries are posed together. The first is interested in students staying in hostels 3 through 7 and the second in students staying in hostels 5 through 9.

   How would you optimize these two queries given that they are posed together? Show the query graph that will be used for the final query processing.

IV. Consider the following equivalence:

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \pi_{L_1 \cup L_2}((\pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\pi_{L_2 \cup L_4}(E_2)))$$

Here L_1 and L_2 are sets of attributes from E_1 and E_2 respectively.
L_3 is the set of attributes of E_1 involved in θ but not in L_1 or L_2.
L_4 is the set of attributes of E_2 involved in θ but not in L_1 or L_2. ...15

1. Explain why this transformation is correctness-preserving.
2. From a performance perspective, when would you recommend using the transformed expression during heuristic query optimization.
3. From a performance perspective, when would you *not* recommend using the transformed expression, that is, recommend using the expression as is.

V. $r \bowtie s$ is sorted on an attribute of r and is estimated to produce n records. But a user is really interested only in the top m records, where $m << n$. ...15
How would you evaluate the query

1. when the join is on a foreign key of r referencing s.
2. when the join is not on a foreign key.

Total Marks = 100