Exercises in Query Processing

You are given two relations r and s. The details about the relations are given below. The main information is shown without any emphasis, but the derived information is shown in italics. In general, you should carefully verify that the derived information could in fact be derived, because it may not always be provided to you explicitly.

Feature	Relation r	Relation s
Attributes	A, B, C, D; 10 bytes each	B, E, F, G, H; 10 bytes each
Primary key	B	B, E
Number of tuples	$n_1 = 20{,}000$	$n_2 = 30{,}000$
Size of a page	1,000 bytes	1,000 bytes
Tuple size	*40 Bytes*	*50 Bytes*
Number of tuples per page	*25*	*20*
Number of data pages	$N_1 = 800$	$N_2 = 1{,}500$
Structure	• Parts 1-6: Ad hoc • Parts 7, 8, and 9: Assume that a permanent B_+-tree on the primary key, with a height of 3 and 700 pages in the sequence set, has been created.	Ad hoc
Selectivity of condition ...	A > 55 is 20%	H = 100 is 2%
Size of natural join involving relations r and s, or their fragments	Natural join involves the common attribute B. Because attribute B is a unique key in r, a given tuple of s can have at most one matching tuple in r. Therefore, in the worst case the number of tuples in the join can not exceed the number of tuples in s (or a fragment of s). Use this as the worst case scenario whenever you need to estimate the size of the join. For example, if r and s are joined without any prior filtering, their join has (at most) 30,000 tuples.	

You are given the following SQL query. Assume that for evaluating this query you have 10 buffers available for your use. The size of a buffer is same as size of a page on the disk. The cost of a plan to execute a query is measured in terms of disk accesses. A disk access consists of either reading a single page from disk into a buffer, or writing a buffer onto a single page. Therefore, e.g., the cost of reading 5 pages is 5,

irrespective of how and when those 5 pages are read. You are not required to do any duplication removal; this makes projection operator straightforward.

```
select x.A
from r x, s y
where x.B = y.B
   and x.A > 55
   and y.H = 100
```

1. Express the above query as an algebraic expression by converting the from clause to a cross product, where clause to a single selection, and the select clause to a projection.

2. Replace the cross product in the expression of Part 1 by a natural join.

3. Transform the algebraic expression of Part 2 to an expression tree. Your conversion should be literal at this point, without any consideration for algebraic optimization.

4. Estimate the cost of evaluating the expression tree of Part 3. Remember that you have 10 buffers available to you in all parts. Don't forget to use buffers to your advantage whenever you can. The cost estimations should include the number of page accesses and numbers of buffers required. This applies to all exercises.

5. Now restructure your expression tree of Part 3 to make it as optimal as you can.

Note that when you do this, you implicitly use algebraic identities. These identities have a lot of different varieties. For example, (a) breaking a selection involving two conditions into a sequence of two selections involving a single condition each (b) Commuting a selection with a join, and (c) creating explicit projections in order to reduce the number of attributes that are put into the pipeline as input to a join operator.

6. Estimate the cost of evaluating your expression tree of Part 5. This is similar to Part 4, but produces a more optimal performance. Consider preprocess r and/or s writing the results as temporary relation(s). Preprocessing means applying selection and projection operators first to reduce the size of inputs for the natural join.

7. Proceed along the lines of Part 6. However, for this part assume that the selectivity for the condition "H = 100" were 4%.

8. Now use the B+-tree on relation r. Estimate the cost of evaluating your expression tree of Part 5 by using index nested loop to compute the join. To do this, use s as the outer relation and r, the one with the index, as the inner relation. Scan relation s page by page, and within a page tuple by tuple. Verify if a tuple satisfies H = 100. If it

does, catch hold of its B-value. For this B value, try to find a match in the B+-tree of relation r, by going through its index.

9. Now consider a sort-merge and estimate the cost of evaluating your expression tree of Part 5. For the sake of simplicity assume that the selection and projections will be applied to r and s first. Outputs will be written to the disk. Then both relations will be sorted. Then their join will be performed by merging them and performing the final projection on the fly. The cost of sorting N pages using k buffers is $2N\log_{k-1}N$. To ease your calculations for cost of sorting, assume that you have 11 buffers available to you for sorting.