CIS560 – Database System Concepts **Name:_____**

Homework Assignment 9 [20 points] – due November 15th by midnight

Note 1: Please remember that you are allowed to discuss the assigned exercises, but you should write your own solution. Identical solutions will receive 0 points.

Note 2: For full credit, show your work (not only the final answers).

Exercise I (Operator algorithms) [6 points]

1. Explain how two relations R and S can be joined together using a two-pass partitioned hash join algorithm. In your explanation, use the following facts: R has 90 pages; S has 80 pages; there are 11 pages of memory. Please provide a detailed explanation that includes (1) the goal of each step, (2) how many pages are allocated as input buffer(s), (3) how they are used, (4) how many pages are allocated as output buffer(s), (5) how they are used, etc. You can assume that hash tables have no overhead (a hash table for a relation that has 9 pages will require 9 pages of memory). You can assume a uniform data distribution.

2. Consider joining two relations R(x, y) and S(x, z) on their common attribute x using a sort-merge join algorithm. The size of relation R is 1000 blocks and the size of relation S is 500 blocks. Suppose the memory buffer has 101 blocks. Assume that attribute x of relation R has two distinct values (x1 and x2) and the values are evenly distributed in R. Similarly, attribute x of relation S also has the same two values (x1 and x2) and the values are evenly distributed in S. Describe the sort-merge join algorithm and compute the total number of disk I/Os that are needed for the algorithm.

   Remember that the two-pass sort-merge join algorithm that we discussed and analyzed in class requires that all tuples with a common value for the join attribute fit in the memory. If this condition is not satisfied, the sort-merge join algorithm needs additional disk I/O's.

Exercise II (Selection cost) [4 points]

Consider a relation R(a, b, c, d) that has a clustering index on a and non-clustering indexes on each of the other attributes. The relevant parameters are:

B(R) = 1000, T (R) = 5000, V (R, a) = 20, V (R, b) = 1000, V (R, c) = 5000, V (R, d) = 500.

Give the best query plan and the disk I/O cost for each of the following selection queries:

(1) $\sigma_{(a=1)AND(b=2)AND(c=3)}(R)$

(2) $\sigma_{(a=1)AND(b=2)AND(c<3)}(R)$

Exercise III (Physical query plans) [5 points]

Consider two relations R(a, b, c) and S(x, y, z) that have the following characteristics:

B(R) = 600 B(S) = 800
T(R) = 3000 T(S) = 4000

V(R, a) = 300 V(S, x) = 100
V(R, b) = 100 V(S, y) = 400
V(R, c) = 50 V(S, z) = 40

We also have M = 1000 (number of memory blocks).

Relation R has a clustered index on attribute a and an unclustered index on attribute c. Relation S has a clustered index on attribute x and an unclustered index on attribute z. All indexes are B+ trees.

Specify and justify a good physical plan for performing the join

$R \bowtie_{a=z} S$ (i.e., join R and S using the condition R.a = S.z)

Your answer should specify the physical join operator used (hash, nested loop, sort-merge, or other) and the access methods used to read relations R and S (sequential scan, index, etc.). Be sure to give essential details: i.e., if you use a hash join, which relations(s) are included in hash tables; if you use nested loops, which relation(s) are accessed in the inner and outer loops, etc.

Give the estimated cost of your solution in terms of number of disk I/O operations needed (you should ignore CPU time and the cost to read any index blocks).

You should give a brief justification why this is the best (cheapest) way to implement the join. You do not need to exhaustively analyze all the other possible join implementations and access methods, but you should give a brief discussion of why your solution is the preferred one compared to the other possibilities.

Exercise IV (Logical query plans) [5 points]

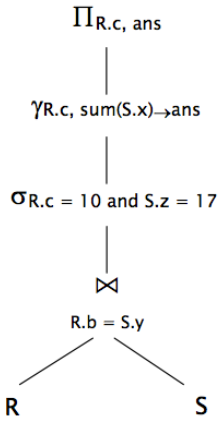
Consider again the two relations R(a, b, c) and S(x, y, z) that we used in Exercise III, with the same characteristics:

B(R) = 600 B(S) = 800
T(R) = 3000 T(S) = 4000

V(R, a) = 300 V(S, x) = 100
V(R, b) = 100 V(S, y) = 400
V(R, c) = 50 V(S, z) = 40

We also have M = 1000 (number of memory blocks).

As in III, relation R has a clustered index on attribute a and an unclustered index on attribute c. Relation S has a clustered index on attribute x and an unclustered index on attribute z. All indices are B+ trees. Now, consider the following logical query plan for a query involving these two relations.



Change or rearrange the original logical query plan to produce one that is equivalent (has the same final results), but which is estimated to be significantly faster, if possible. Recall that logical query optimization does not consider the final physical operators used to execute the query, but only things at the logical level, such as the sizes of relations and estimated sizes of intermediate results.
You should include a brief but specific explanation of how much you expect your changes to improve the speed of the query and why.