1. Suppose you need to sort a relation r using sort-merge and merge-join the result with an already sorted relation s. Is it possible to pipeline the output of sort-merge of r to the merge join? Explain.

- Assumption: the memory is large enough for the simultaneous execution of the two operations.
- Using pipelining, output from the sort-merge of r is written to a buffer B at the final merge pass.
- When B is full, the merge-join processes tuples from B, joining them with tuples from s until B is empty.
- At this point, the sort-merge operation is resumed and B is refilled.
- This process continues until the merge-join is complete.

2. Consider the query $\prod_{A,B,C,D}(R \bowtie_{A=C} S)$. Given the following information:

- R is 10 blocks long, and R tuples are 300 bytes long.
- S is 100 blocks long, and S tuples are 500 bytes long.
- No common attribute in R and S.
- The block size is 1024 bytes.
- Tuples do not span across blocks.
- Each S tuple joins with exactly one R tuple.
- The combined size of attributes A, B, C, and D is 450 bytes.
- A and B are in R and have a combined size of 200 bytes; C and D are in S.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

a) What is the size of the final result, in terms of number of tuples and number of blocks, assuming that the number of duplicates is negligible?
b) Transform the query so that projection is done before join.
c) Suppose that three memory blocks are available and ***block nested-loop join*** is used. Suppose that projection (and duplicate elimination) is based on sorting. Compute the cost in terms of number of block transfers for each of the following orders.
   (i) Join followed by projection
   (ii) Projection followed by the join.

a)

- b_R: 10 blocks; $f_R = \lfloor 1024/300 \rfloor = 3$; n_R: 30 tuples
- b_S: 100 blocks; $f_S = \lfloor 1024/500 \rfloor = 2$; n_S: 200 tuples
- Since every S tuple joins with exactly one R tuple, there can be 200 tuples after the join.
- Since the size of the result is 450 bytes/tuple, 2 tuples fit on a block. This means 200 / 2 = 100 blocks in the final result.

b) By Rule 8(a), we have
$\prod_{A,B,C,D}(R \bowtie_{A=C} S) \equiv (\prod_{A,B}(R)) \bowtie_{A=C} (\prod_{C,D}(S))$

c)

(i) Cost of join followed by projection:

- Cost of block nested-loop join = (10 + 10*100) = **1010** block transfers
- Number of tuples after join is 200 tuples, each of size 800 bytes. Thus, only one result tuple fits on a block, and **200** blocks have to be written.
- Projection is sort-based using 3 memory blocks. During the initial sorted run creation, unwanted attributes are eliminated on-the-fly to produce tuples of size 450 bytes, i.e., 2 tuples per block. Thus, 200 blocks are scanned and 100 blocks written with $\lceil 100/3 \rceil = 34$ sorted runs created.
- These sorted runs are merged pairwise in $\lceil \log_2 34 \rceil = 6$ passes.
- Projection (and duplicate elimination) cost = 200+100+100 * (2*6-1) = **1400** block transfers.
- Total cost = 1010 + 200 + 1400 = 2610 block transfers.

(ii) Cost of projection followed by join

- Projection is sort-based using 3 memory blocks.
- Sort relation S. During the initial sorted run creation, unwanted attributes are eliminated on-the-fly to produce tuples of size 250 bytes, i.e., 4 tuples per block. Thus, 100 blocks are scanned and 50 blocks written with $\lceil 50/3 \rceil = 17$ sorted runs created.
- These sorted runs are merged pairwise in $\lceil \log_2 17 \rceil = 5$ passes.
- Projection (and duplicate elimination) cost (including writing the result to disk) of S = 100 + 50 + 50 * (2*5) = **650** block transfers.
- Sort relation R. During the initial sorted run creation, unwanted attributes are eliminated on-the-fly to produce tuples of size 200 bytes, i.e., 5 tuples per block. Thus, 10 blocks are scanned and 6 blocks written with $\lceil 6/3 \rceil = 2$ sorted runs created.
- These sorted runs are merged pairwise in 1 pass.
- Projection (and duplicate elimination) cost (including writing the result to disk) of R = 10 + 6 + 6 * 2 = **28** block transfers.
- The cost of block nested-loop join is (6 + 6*50) = **306** block transfers.
- Total cost = 650 + 28 + 306 = 984 block transfers.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder