Gradiance Online Accelerated Learning

Dian Hao

- Home Page
- Assignments Due
- Progress Report
- Handouts
- Tutorials
- Homeworks
- Lab Projects
- Log Out

Help

Submission number:	616133
Submission certificate:	EE741305
Submission time:	2022-02-22 07:54:42 PST (GMT - 8:00)

Number of questions:	5
Positive points per question:	3.0
Negative points per question:	1.0
Your score:	15

relational operators

1. Suppose we compute the nested-loop join of two relations, one of which occupies 2000 blocks and the other occupies 500 blocks. There are 40 main-memory buffers available to hold blocks of one of these two relations. How many disk blocks must be read to compute their nested-loop join? Note that we do not count the space or disk I/O's needed to accumulate or write the tuples of the result.

a) 26,500

b) 25,500

c) 26,142

d) 28,000

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

The number of disk-block reads needed for this nested-loop join is computed by:

1. Take the smaller of the number of blocks required by the two relations, that is, min(2000, 500).
2. Divide it by (40 - 1), that is, one less than the number of buffers available.
3. If the result is not an integer, raise the quotient to the next higher integer.
4. Multiply that integer by the larger of 2000 and 500.
5. Add in the smaller of 2000 and 500.

In this example, the result is 26,500.

2. Relation R(a,b,c) has 10,000 tuples. *R* is stored in 200 blocks, each holding 50 tuples of *R*. Attribute *a* has 80 different values; *b* has 100 different values, and *c* has 40 different values. There is a clustering index on *a*, with tuples packed completely contiguously (i.e., no intervening, out-of-order tuples). There is a nonclustering index on *b*, and no index on *c*. Assume that for each attribute, each possible value occurs the same number of times, and assume that the values of the three attributes are completely uncorrolated. Finally, assume that 10 is a value that occurs in all three attributes. Neglecting any index blocks retrieved, how many blocks of *R* that must be retrieved to answer each of the selection queries $\sigma_{a=10}(R)$, $\sigma_{b=10}(R)$, and $\sigma_{c=10}(R)$? *Note*: in some cases, the correct answer is a range of possible numbers of

blocks. Based on your conclusions, pick the correct statement from the list below.

a) To answer $\sigma_{b=10}(R)$, exactly 100 blocks may be retrieved.

b) To answer $\sigma_{c=10}(R)$, 250 blocks must be retrieved.

c) To answer $\sigma_{c=10}(R)$, at most 100 blocks must be retrieved.

d) To answer $\sigma_{b=10}(R)$, exactly 1 block may be retrieved.

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

First, consider attribute *a*. There are 80 different values distributed evenly among 10,000 tuples, so there should be 125 tuples with a=10 (or any particular value that exists in the relation). As there is a clustering index on *a*, these 125 tuples will be consecutive. They require at least 3 blocks, since a block holds only 50 tuples. However, they might be spread over 4 blocks, say 10 on the first, 50 on each of the next two blocks, and 15 on the fourth block.

For attribute *b*, we retrieve 100 tuples with *b* = 10. Since there is a nonclustering index on *b*, those 100 tuples could be on 100 different blocks. Or, they could all be packed into only 2 blocks, or they could appear spread over any number of blocks between 2 and 100.

Finally, consider attribute *c*. There are 250 tuples with *c* = 10. With no index, there is no way to find them other than to retrieve all 200 blocks and search for them.

3. Two relations *R* and *S* are having their bag-union taken by a sort-based algorithm. We have already sorted *R* and *S*, and we assume that their tuples are simply integers, 1 through 5. We also assume that there are only three tuples per block, and that *R* and *S* each consist of nine tuples, arranged in three blocks each, as shown:

	R1	R2	R3
R	1 1 2	2 3 4	4 5 5
S	1 1 1	1 2 2	2 4 5
	S1	S2	S3

If the merge process is allowed to break ties arbitrarily, i.e., take from either relation when the two tuples at the fronts of the lists are the same, then the blocks can be emptied in various orders. Determine which blocks must have all (or at least one) of their records taken before all (or at least one) record from another block. Then, identify which of the following is possible.

a) Some tuple of S3 is taken before any of the tuples of R1.

b) Every tuple of R2 is taken before any of the tuples of S3.

c) Some tuple of S2 is taken before any of the tuples of R1.

d) Every tuple of R2 is taken before the last tuple of S2.

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

Let X and Y be blocks, one from R and the other from S. In order for **every** tuple of X to be taken before any tuple of Y, it is necessary and sufficient that there be no tuple of Y that is strictly less than the highest tuple of X. For

example, it is possible that every tuple of S2 = (1,2,2) is taken before any tuple of R2 = (2,3,4). That would occur, for example, if we always broke ties in favor of S.

For every tuple of X to be taken before the last tuple of Y, it is only necessary that the last tuple of Y be no less than the last tuple of X. For example, it is possible that every tuple of R1 = (1,1,2) is taken before the last tuple of S2 = (1,2,2).

For some tuple of X to appear before any tuple of Y, it is necessary and sufficient that the first tuple of X be no greater than the first tuple of Y. For example, it is possible that the first tuple of S2 = (1,2,2) is taken before any of the tuples of R1 = (1,1,2). That would occur, e.g., if we preferentially took 1's from S before R.

And the condition that some tuple of X is taken before the last tuple of Y is true if and only if the first tuple of X is no greater than the last tuple of Y. For example, the 2 from S3 = (2,4,5) could be taken before the last tuple of R1 = (1,1,2). Again, that would occur if we always broke ties in favor of S.

4. We wish to perform a 2-pass hash join of relations R and S, whose files require $B(R)$ and $B(S)$ disk blocks respectively. We have M one-block buffers available for reading input; we do not count the buffers or disk I/O needed to store or write the result. Write the formula, in terms of $B(R)$ and $B(S)$ for the smallest value of M for which it is possible to perform this join, assuming the hash function divides tuples into buckets as evenly as possible. Demonstrate that you have the correct function by giving the minimum number of needed buffers when the two relations have files requiring 750 and 250 blocks.

a) 17

b) 15

c) 251

d) 28

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

The first pass divides R and S into B partitions, of sizes $B(R)/M$ and $B(S)/M$ blocks each, respectively. In order to complete the join in one more pass, at least one of these numbers of blocks must fit in the available buffers, with one buffer left over to receive blocks of the larger relation, one at a time. That is, we must have $M - 1 \geq \min(B(R)/M, B(S)/M)$. Put another way, the answer is the smallest M such that $M(M - 1) \geq \min(B(R), B(S))$. For the values given, $B(R) = 750$ and $B(S) = 250$, the smallest possible value of M is 17.

5. Relation R has r tuples, of which 100 fit in one block. Relation S has s tuples, of which 200 fit in one block. There are M memory blocks available to take the nested-loop join of R and S. As a function of r, s, and M, how many disk I/O's are needed to execute a nested-loop join of R and S? Demonstrate your understanding by indicating for which of the following values of r, s, and M is D the correct number of disk I/O's required by the nested-loop-join algorithm.

a) $r = 30{,}000$; $s = 40{,}000$; $M = 101$ and $D = 1200$

b) $r = 10{,}000$; $s = 20{,}000$; $M = 101$ and $D = 200$

c) $r = 100{,}000$; $s = 40{,}000$; $M = 101$ and $D = 2400$

d) $r = 50{,}000$; $s = 80{,}000$; $M = 201$ and $D = 1500$

Answer submitted: **b)**

You have answered the question correctly.

Question Explanation:

The precise formula for the number of disk I/O's required by nested-loop join (in its block-based version) is min[B(R),B(S)] + B(S)*B(R)/(M-1), where S is whichever relation takes the smaller number of blocks. In this example, B(R) = r/100 and B(S) = s/200. Identifying the correct choice is thus a simple matter of plugging r, s, and M into this formula.