INFS2200/7903 – Relational Database Systems

School of Information Technology and Electrical Engineering (ITEE), UQ

Tutorial 7

Question 1 Consider a relation R(A,B,C) containing 5,000,000 records, where each data page (i.e., block) of the relation holds 10 records. R is organized as an ordered file that is sorted on R.A. Assume that R.A is a unique key for R, with values lying in the range 0 to 4,999,999. For each of the following relational algebra queries, state which of the following two approaches is most likely to be more efficient (i.e., reads fewer number of blocks) and justify your answer.

Approaches:
1. Access the sorted file for R directly.
2. Use a B+ tree index on attribute R.A.

Relational algebra queries:

A. $\sigma_{A \leq 50,000}(R)$

B. $\sigma_{A \geq 50,000 \text{ and } A < 50,010}(R)$

C. $\sigma_{A \neq 50,000}(R)$

Question 2 Consider the join $R \bowtie_{R.A=S.B} S$, given the following information about the relations to be joined. The cost metric is the number of block read operations.

- Relation R contains 10,000 tuples and has 10 tuples per block.
- Relation S contains 2,000 tuples and also has 10 tuples per block.
- Attribute B of relation S is the primary key for S.
- Both relations are stored as simple heap (unsorted) files.
- Neither relation has any index built on it.
- 52 buffer blocks are available.

A. What is the cost of joining R and S using a tuple-based nested loop join?

B. What is the cost of joining R and S using a page-oriented nested loop join?

C. What is the cost of joining R and S using a block nested loop join?

D. How many tuples does the join of R and S produce at most, and how many blocks are required to store the result of the join back on disk?

E. Would your answer to Q2(A) change, if you were told that R.A is a foreign key that refers to S.B?

Answers for Question 1 are given below:

A. Access the sorted file: Read all the blocks from the beginning of file until the tuple with A=50,000.

Use a B+ tree index: Search the B+ tree for the address of the block containing the record with A=50,000; then read all the blocks up to that one from the beginning of the data file.

The choice of accessing the sorted file is slightly superior to using the B+ tree index simply because of the lookup cost (i.e., search) required on the B+ tree.

B. Access the sorted file: Do a binary search on the file to find the block containing the record with A=50,000.

Use a B+ tree index: Search the B+ tree for the address of the block containing the record with A=50,000; then read that block.

A B+ tree should be superior since searching a B+ tree typically requires reading less blocks than performing a binary search on the sorted file.

C. Access the sorted file: Read all data blocks but do not include the record with A=50,000 in the result.

Use a B+ tree index: Search the B+ tree for the address of the block containing the record with A=0; then read all the data blocks from the file but do not include the record with A=50,000 in the result.

Accessing the sorted file should be slightly more efficient, again because of the lookup cost (i.e., search) required on the B+ tree.

Answers for Question 2 are given below:

A. Let: N_R = 10,000 be the number of records in R;
N_S = 2000 be the number of records in S;
B_R = 1000 be the number of blocks in R;
B_S = 200 be the number of blocks in S; and
N_B = 52 be the number of buffer blocks available.

The basic idea of tuple-based nested loop join is to read each block of the outer relation, and for each record scan the inner relation for matching tuples.

Total cost would be #blocks_in_outer + (#records_in_outer * #blocks_in_inner), which is minimized by having the smaller relation be the outer relation.

Total Cost = $B_S + (N_S * B_R) = 2{,}000{,}200$.

B. The basic idea of page-oriented nested loop join is to read each block of the outer relation, and for each block scan the inner relation for matching tuples.

Total cost would be #blocks_in_outer + (#blocks_in_outer * #blocks_in_inner), which is minimized by having the smaller relation be the outer relation.

Total Cost = $B_S + (B_S * B_R) = 200{,}200$.

C. The basic idea of block nested loop join is to read the outer relation in chunks of blocks, and for each chunk scan the inner relation for matching tuples.

Hence, the outer relation is still read once, but the inner relation is scanned only once for each chunk of blocks.

Total Cost = $B_S + (B_S / (N_B - 2)) * B_R = 4{,}200$.

D. Any tuple in R can match at most one tuple in S because S.B is a primary key (which means the S.B field contains no duplicates). Hence, the maximum number of tuples in the result is equal to the number of tuples in R, which is 10,000.

The size of a tuple in the result could be as large as the size of a R tuple plus the size of an S tuple (minus the size of the shared attributes). This may allow only 5 tuples to be stored on a block. Storing 10,000 tuples at 5 per block would require 2,000 blocks in the result.

E. The foreign key constraint tells us that for every R tuple there is exactly one matching S tuple (because S.B is a key).

If we make R the outer relation, then for each tuple of R we only have to scan S until a match is found. This will require scanning only 50% of S on average.

For tuple-based nested loop join, the new cost would be: Total Cost = $B_R + (N_R * B_S/2) = 1{,}001{,}000$.