

CS411 Database Systems
Fall 2009

HW#4

Due: 3:15pm CST, November 17, 2009

Note: Print your name and NetID in the upper right corner of every page of your submission. Hand in your stapled homework to Donna Coleman in 2106 SC. In case Donna is not in office, slide your homework under the door.

To grade homeworks faster, the homework is partitioned into two parts. **Please, submit each part separately.** For each part, make sure to write down your name and NetID. Handwritten submissions will be graded but they will take longer to grade. For clarity, machine formatted text is preferable: Expect to lose points if your handwritten answer is unclear or misread by the grader.

This homework is partitioned into two parts as follows:

- Part 1: Problem 1 - Problem 3
- Part 2: Problem 4 - Problem 7

Part 1

Problem 1 Nested-Loop Joins (16 points)

1. Suppose $B(R) = B(S) = 10,000$. For what value of M would we need to compute $R \bowtie S$ using the nested-loop join algorithm with no more than the following number of I/Os? (8 points, 4 points each)

Using the equation given in Section 15.3.4 of the textbook, solve for M :

$$I/O = B(S) + \frac{B(S)B(R)}{(M-1)}$$

(a) 100,000

$$100,000 = 10,000 + \frac{(10,000 \times 10,000)}{(M-1)}$$
$$M = 1,112.1 \text{ or } \text{ceil}(M) = 1,113$$

(b) 25,000

$$25,000 = 10,000 + \frac{(10,000 \times 10,000)}{(M-1)}$$
$$M = 6,667.7 \text{ or } \text{ceil}(M) = 6,668$$

2. If two relations R and S are both unclustered, it seems that the nested-loop join algorithm requires about $T(R)T(S)/M$ disk I/Os. How can you do significantly better than this cost? Describe your modified version of the nested-loop algorithm and give the number of disk I/Os required for your algorithm. We assume that M is large enough such that $M \gg 1 \simeq M$, and that $B(R) \ll T(R)$ and $B(S) \ll T(S)$; that is, the number of tuples of a relation is much greater than that of blocks of the relation. (8 points)

Note that the cost of algorithm given in the question is $T(R)T(S)/M$, which means it is using tuple-based nested-loop join. In order to improve the disk I/O cost of nested-loop join algorithm, we need to use block-based nested-loop join. In order to carry out block-based nested loop join efficiently, we need the inner relation clustered, and search structure built on the common attributes of R and S .

Let R be the inner relation (assuming S is smaller):

- Cost of reading all tuples of R , cluster them, and write them back: $T(R) + B(R)$
- Cost of Reading tuples of S , plus the cost of joining them with R in the main memory:

$$T(S) + \frac{B(S)B(R)}{M}$$

Therefore the total cost is $T(R) + B(R) + T(S) + \frac{B(S)B(R)}{M}$.

Problem 2 Two-pass Algorithms Based on Sorting (20 points)

1. Suppose we have a relation with 1,000,000 records and each records requires 10 bytes. Let the disk-block size be 4,096 bytes. (8 points, 4 points each)

- (a) What is the minimum number of blocks in main memory required for using TPMMS (Two-Phase Multiway Merge-Sort) to sort these records?

The size of the relation in bytes is $1,000,000 \times 10 = 10,000,000$ bytes, and each disk-block is 4,096 bytes. The minimum number of blocks to hold the relation is $\text{ceil}(\frac{10,000,000}{4,096}) = 2,442$. The minimum M requirement for TPMMS is $B \leq M^2$, M must at least be $\text{ceil}(\sqrt{2,442}) = \text{ceil}(49.4) = 50$.

- (b) Following (a), how many disk I/Os are needed to sort all the records?

Number of disk I/O for TPMMS is $3B$, which is $3 \times 2,443 = 7,329$

2. We have two relations R and S where $B(R) = B(S) = 10,000$. Give an approximate size of main memory M required and the number of disk I/Os in order to perform the two-pass algorithms for the following operations: (12 points, 4 points each)

- (a) set union

Using the equation given in Section 15.4.9 of the textbook:

$$\text{I/O of set union operation} = 3 \times (B(S) + B(R))$$

$$= 3 \times (10,000 + 10,000)$$

$$= 60,000$$

Approximate M requirement for set union operation is $\sqrt{(B)R + B(S)} = \sqrt{20,000} = 141.42$, which the given M satisfies.

- (b) simple sort-join

$$\text{I/O of set union operation} = 5 \times (B(S) + B(R))$$

$$= 5 \times (10,000 + 10,000)$$

$$= 100,000$$

Note that given figures satisfy the required M, which is $B(S)$ and $B(R) \leq M^2$, i.e. $10,000 \leq 10,00,000$

- (c) the more efficient sort-join described in Section 15.4.8 of the textbook

$$\text{I/O of set union operation} = 3 \times (B(S) + B(R))$$

$$= 3 \times (10,000 + 10,000)$$

$$= 60,000$$

Requirement for M in the efficient sort-join algorithm is $B(R) + B(S) \leq M^2 = 20,000 \leq 1,000,000$ which is satisfied here.

Problem 3 Hash-based and Index-based joins (14 points)

1. If $B(R) = 10,000$ and $B(S) = 30,000$ and $M = 101$, what is the number of disk I/Os required for the hash-join algorithm? (5 points)

disk I/O estimation for hash-join algorithm is $3 \times (B(R) + B(S)) = 3 \times (10,000 + 30,000) = 120,000$.

2. Suppose $B(R) = 10,000$ and $T(R) = 500,000$. Let there be an index on $R.a$, and let $V(R, a) = k$ for some number k . Give the cost of $\sigma_{a=0}(R)$, as a function of k , under the following circumstances. You may neglect disk I/O's needed to access the index itself.

(a) the index is clustering.

$$\text{Cost of } \sigma_{a=0}(R) = \frac{B(R)}{k} = \frac{10,000}{k}$$

Note that if the index is clustered, then the arrangements of the index corresponds to the blocks allowing fewer data block reads.

(b) the index is not clustering.

$$\text{Cost of } \sigma_{a=0}(R) = \frac{T(R)}{k} = \frac{500,000}{k}$$

The cost is higher than scanning entire R , because index is not clustered.

(c) R is clustered, and the index is not used.

If R is clustered but we do not use the index, then we need to retrieve every block of R , which in this case is 10,000 disk I/O's.

Part 2

Problem 4 Algebraic Laws (10 points, 5 points each)

1. We consider two relations $R(A, B, C)$ and $S(C, D, E)$. Convert the following expressions in relational algebra by applying algebraic laws so that we can perform selections and projections as early as possible.

(a) $\sigma_{B=3} \text{AND}_{E=4}(R \bowtie \sigma_{C>10}(S))$

$$\sigma_{(B=3) \text{AND} (C>10)}(R) \bowtie \sigma_{C>10} \text{AND}_{E=4}(S)$$

(b) $\pi_{A,D}(R \bowtie S)$

$$\pi_{A,D}(\pi_{A,C}(R) \bowtie \pi_{C,D}(S))$$

Problem 5 Dynamic Programming (15 points)

Compute the optimal plan for $R \bowtie S \bowtie T \bowtie U$ using the technique of dynamic programming. We make the following assumptions (as we did in the class):

- $B(R) = 400$, $B(S) = 800$, $B(T) = 600$, and $B(U) = 700$.
- The size of a join for two relations $R1$ and $R2$ is estimated as: $B(R1 \bowtie R2) = 0.01 \times B(R1) \times B(R2)$. If a subplan is a single relation and does not involve any join, the size of its intermediate result is zero.
- The cost of a join is estimated to be the cost of the subplans plus the size of the intermediate results.
- The cost of a scan is zero.

Draw the table for dynamic programming, to show how you compute the optimal plan for all possible join orders allowing all trees.

| Subquery | Size | Lowest cost | Plan |
|----------|--------|-------------|----------|
| RS | 3200 | 0 | RS |
| RT | 2400 | 0 | RT |
| RU | 2800 | 0 | RU |
| ST | 4800 | 0 | ST |
| SU | 5600 | 0 | SU |
| TU | 4200 | 0 | TU |
| RST | 19200 | 2400 | (RT)S |
| RSU | 22400 | 2800 | (RU)S |
| RTU | 16800 | 2400 | (RT)U |
| STU | 33600 | 4200 | (TU)S |
| RSTU | 134400 | 7400 | (RS)(TU) |

Figure 1: dynamic programming plan

Problem 6 Cost Estimation (15 points, 3 points each)

Consider two relations R(A, B, C, D) and S (D, E) with the following statistics:

$T(R) = 100$, $V(R, A) = 100$, $V(R, B) = 10$, $V(R, C) = 1$, $V(R, D) = 50$; $T(S) = 500$, $V(S, D) = 30$, $V(S, E) = 100$.

(a) Estimate the number of tuples in $\sigma_{B=25}(R)$

$$T(\sigma_{B=25}(R)) = \frac{T(R)}{V(R,B)} = \frac{100}{10} = 10$$

(b) Estimate the number of tuples in $\sigma_{(B=25)AND(C=30)}(R)$

$$T(\sigma_{B=25ANDC=30}(R)) = \frac{T(R)}{V(R,B) \times V(R,C)} = \frac{100}{10 \times 1} = 10$$

(c) Estimate the number of tuples in $\sigma_{B>25}(R)$

$$T(\sigma_{B>25}(R)) = \frac{T(R)}{3} = \frac{100}{3} = 33$$

(d) Estimate the number of tuples in $\sigma_{(B>25)AND(B=15)}(R)$

$B = 15$ and $B > 25$ are mutually exclusive predicates, therefore $T(\sigma_{(B>25)AND(B=15)}(R)) = 0$

(e) Estimate the number of tuples in $R \bowtie S$

$$T(R \bowtie S) = \frac{T(R)T(S)}{\max(V(R,D), V(S,D))} = \frac{50,000}{50} = 1,000$$

Problem 7 Pipelining Versus Materialization (10 points)

Consider physical query plans for the expression

$$(R(w, x) \bowtie S(x, y)) \bowtie U(y, z)$$

in Example 16.36 on page 831 of the textbook (We covered the same example in the class). If $B(R) = 2,000$, how would you update the table in Figure 16.38 on page 834 of the textbook? Show the revised table. We use the same physical plans for the three cases in the table respectively.

With all the other conditions kept consistent with the example in the textbook, the following are the only changes we need to consider:

- Consider $R \bowtie S$ first. Neither relation fits in main memory so we use two-pass hash-join method. The number of disk I/O for hash-based join is $3(B(R) + B(S))$ given that $\min(B(R), B(S)) \leq M^2$, which is satisfied here with $B(R) = 2,000$; $\min(2,000, 10,000) \leq 101^2$. So the disk I/O for $R \bowtie S$ is $3 \times (2,000 + 10,000) = 36,000$.
- The number of blocks required to store R in the main memory is $\text{ceil}(\frac{2,000}{101}) = 20$. This means the rest of 80 blocks can be used in the join operation, and remaining 1 bucket is used as an output buffer.

The updated table is as follows:

| Range of K | Pipeline or Materialize | Algorithm for final join | Total Disk I/O's |
|-----------------------|-------------------------|--------------------------|------------------|
| $K < 80$ | Pipeline | One-pass | 46,000 |
| $80 \leq k \leq 8000$ | Pipeline | 80-bucket, two-pass | $66,000 + 2k$ |
| $8000 < k$ | Materialize | 100-bucket, two-pass | $66,000 + 4k$ |

Figure 2: cost of physical plans