CS411 Database Systems
Fall 2007

Final Exam Solutions

Problem 1 (*15 points*)

(1) True; (2) False; (3) False; (4) True; (5) False;
(6) False; (7) True; (8) False; (9) False; (10) True;
(11) False; (12) False; (13) False; (14) True; (15) False;

Problem 2 (*10 points*)

The distribution of pointers are like the following.

times of being followed	0	1	2	3
Total	20	80	60	40
Automatically swizzled	6	24	18	12
On-demand swizzled after 1st access	N/A	28	21	14
On-demand swizzled after 2nd access	N/A	N/A	21	14

- For the pointers that are never followed: subtotal is 120
  - cost of automatic swizzling of 30%: $6 \times 20 = 120$
- For the pointers that are followed only once: subtotal is 1904
  - cost of automatic swizzling of 30%: $24 \times 20 = 480$
  - cost of following the swizzled pointers: $24 \times 1 = 24$
  - cost of following the unswizzled pointers: $(80 - 24) \times 10 = 560$
  - cost of on-demand swizzling: $28 \times 30 = 840$
- For the pointers that are followed twice: subtotal is 2307
  - cost of automatic swizzling of 30%: $18 \times 20 = 360$
  - cost of following the swizzled pointers: $18 \times 2 + 21 \times 1 = 57$
  - cost of following the unswizzled pointers: $(21 + 21 \times 2) \times 10 = 630$
  - cost of on-demand swizzling: $(21 + 21) \times 30 = 1260$
- For the pointers that are followed three times: subtotal is 1578
  - cost of automatic swizzling of 30%: $12 \times 20 = 240$
  - cost of following the swizzled pointers: $12 \times 3 + 14 \times 2 + 14 \times 1 = 78$
  - cost of following the unswizzled pointers: $(14 + 14 \times 2) \times 10 = 420$
  - cost of on-demand swizzling: $(14 + 14) \times 30 = 840$

The total expected cost is $120 + 1904 + 2307 + 1578 = 5909$.

(ii) Solution: In such situation a block could be pinned because of swizzled pointers: Suppose a block B_1 has within it a swizzled pointer to some data item in block B_2, and we move block B_2 back to disk. Now, should we follow the pointer in B_1, it will lead us to the buffer, which no longer holds B_2. In effect, the pointer has become dangling.

To unpin a block that is pinned because of swizzled pointers from outside, we must "unswizzle" any pointers to it. Consequently, the translation table must record, for each database address whose data item is in memory, the places in memory where swizzled pointers to that item exist. Two possible approaches are:

1. Keep the list of references to a memory address as a linked list attached to the entry for that address in the translation table.

2. If memory addresses are significantly shorter than database addresses, we can create the linked list in the space used for the pointers themselves. That is, each space used for a database pointer is replaced by (a) The swizzled pointer, and (b) Another pointer that forms part of a linked list of all occurrences of this pointer.

Problem 3 (*10 points*)

(a) With height unchanged, this tree can hold at most $5 \times 5 \times 4 = 100$ records. The number of current records is 18. Therefore, this tree can accommodate 82 more records.

(b) $10 < 50 \rightarrow$ 1st pointer, $8 < 10 < 18 \rightarrow$ 2nd pointer, find 10 in the 2nd key. get records with keys 10, 18,27,32,39,41,45,52,58 by following chains.

(c) See Figure 1.

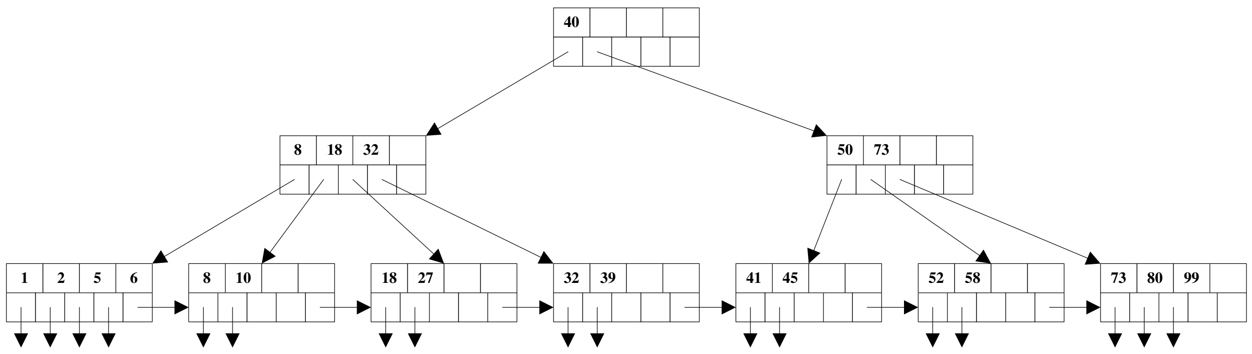


Figure 1: Solution for Problem 4(c)

Problem 4 (*14 points*)

(i) $6,000/20 = 300$

(ii) $T(R)T(S)/\max(V(R,b), V(S,b)) = 10,000 \times 6,000/200 = 300,000$

(iii) Accessing index to find $b = 100$ needs 3 blocks
Accessing index to find $b = 1000$ needs 3 blocks

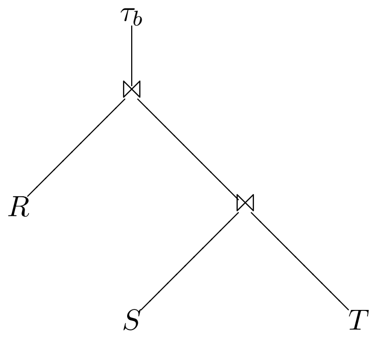
The tuples with $b = 100$ are estimated as $T(S)/V(R,b) = 60$. These will occupy at best 15 blocks but at worst 16 blocks.
Similarly, tuples for $b = 1000$ will occupy at worst 16 blocks.
So, the worst cost of this query should be 3 + 3 + 16 + 16=38 blocks.

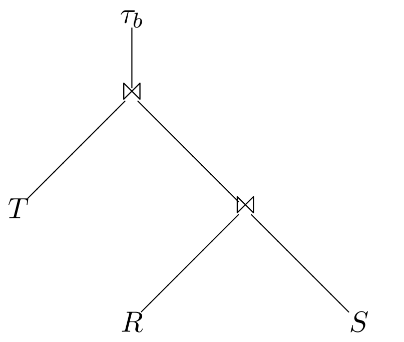
(iv) We shall use 100 blocks to buffer R. Thus, for each iteration, we do 100 disk I/O's to read the chunk of R and read S entirely in the second loop. So, the total number of disk I/O's is (1,000/100)(100+1,500)=16,000

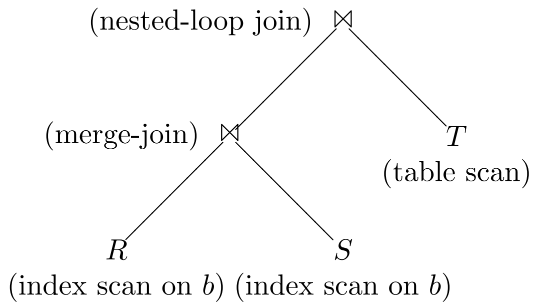
(v) The approach needs $3B(R) + 3B(S) = 75,000$. We have 25 subsorted lists for relations R and S. Therefore, $K \leq M - 25$ must holds.

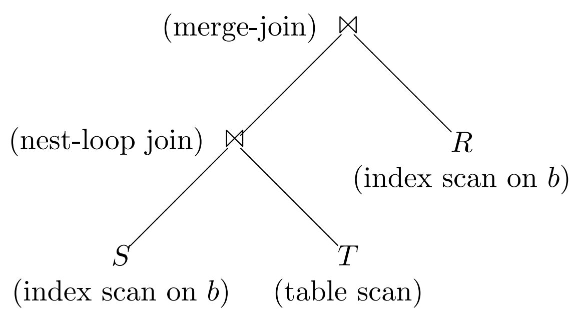
Problem 5 (*17 points*)

(i)









(ii)

There may be several answers.

(iii)

Plan	$R \bowtie S$	interesting order?
Plan A	(table-scan R) nested-loop-join (table-scan S)	no
Plan B	(index-scan R on b) merge-join (index-scan S on b)	yes
Plan C	(index-scan R on b) nested-loop-join (table-scan S)	yes

Plan A: (table-scan R), (table-scan S) produce no order, and nested-loop-join does not produce any sorted order.

Plan B: (index-scan R on b) produces tuples sorted on b, (index-scan S on b) produces tuples sorted on b, and merge-join will produce a sorted output on b: Thus, it will produce an interesting order.

Plan C: (index-scan R on b) produces tuples sorted on b, and using nested-loop-join with a sorted relation as the outer loop will produce a sorted output: Thus, it will produce an interesting order.

(iv)

Plan	estimated cost	keep?
Plan A	2000	yes
Plan B	3000	yes
Plan C	4000	no

Plan A: The plan has the smallest cost, so it will be kept.

Plan B: It produces an interesting order, although does not have the smallest cost, so will be kept.

Plan C: Although it produces an interesting order, its cost is higher than Plan B, so this plan won't be kept.

Problem 6 (*12 points*)

(i) A= 21, B= 11 or 41, C = 30 or 31 or 32 or 33, D = 50, 51, 52

(ii) T1, T6.

(iii) T3, T4, T5

(iv) A = 20 B = 41 C = 31 D = 52
A= 20 because T1 is redone.
B= 41 because T4 is redone.

C= 31 because both T1 and T6 are undone and T3 is redone.
D = 52 because T5 is redone.

Problem 7 (*12 points*)

Answer the following questions:

(i) We do not need to shut down the system while the checkpointing is being made, so new transactions enter the system during checkpointing.

(ii) No, it is not since the graph has cycle.

(iii) Yes, deadlock occurs with such an interleaving of the actions of these transactions. T_1 can gain read lock on X and T_2 can gain read lock on Y, but T_2 cannot gain write lock on X because it has conflict with read lock of transaction T_1, so T_2 has to wait. Similarly, T_1 cannot gain write lock on Y due to the conflict of read lock of the transaction T_1. So, T_2 cannot proceed as well and they will wait forever.

(iv) As T_1 is the first to validate, there is nothing to check; T_1 validates successfully. T_3 validates next. The only other validated transaction is T_1, and T_1 has not yet finished. Thus, both the read- and write-sets of T_3 must be compared with the write-set of T_1. However, T_1 writes only A, and T_3 neither reads nor writes A, so T_3's validation succeeds. Last, T_2 validates. Both T_1 and T_3 finish after T_2 started, so we must compare the read-set of T_2 with the write-sets of both T_1 and T_3. In addition, since T_3 has not finished yet when T_2 is validating, the write-set of T_2 must be compared with the write set of T_3. However, there is no common element in the two sets. Thus, T_2 can also validates.

Problem 8 (*10 points*)

(i)
$l_3(C)$ denied (C locked by 1)
$l_1(B)$ denied (B locked by 2)
$l_4(D)$ denied (D locked by 2)
$l_2(A)$ denied (A locked by 1)

wait-for graph is:

$$\begin{tikzcd}
1 \arrow[r, leftrightarrow] & 2 \\
3 \arrow[u] & 4 \arrow[u]
\end{tikzcd}$$

Yes, a deadlock occurs, because of a cycle between 1 and 2 in the wait-for graph.

(ii)
Yes, it can have a deadlock.

example:
T1 starts
T2 starts
T1 reads A
T1 writes A

T2 writes B
T1 writes B (result: T1 wait for T2 to commit or abort)
T2 reads A (result: T2 wait for T1 to commit or abort)
deadlock occurs