CS411 Database Systems
Fall 2008

Final Exam Solutions

Problem 1 (*15 points*)

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

Problem 2 (*12 points*)

(i) It may be convenient to have the length in the record itself. The length field allows us to avoid examining the record contents if all we need is to find the beginning of the next record quickly.

(ii) (a)
- record header: 12 bytes
- name field: 30 bytes. Since each field must start at a multiple of four, this field actually takes 32 bytes.
- gender field: 1 byte. Since each field must start at a multiple of four, this field actually takes 4 bytes.
- address field: 255 bytes + string's endmarker = 256 bytes.

RecordLength = 12 + 32 + 4 + 256

(b)
- block header: 40 bytes
- actual block length: 2^{20} - 40
- record length: RecordLength

Maximum Number of Records = $\lfloor ActualBlockLength/RecordLength \rfloor$

Problem 3 (*10 points*)

(a)

(b) B-tree index is great for both equality and range queries. However, hash table index is more efficient for equality queries, but it cannot support range queries.

Problem 4 (*12 points*)

(i) Read R into main memory. Then, for each tuple t of S, find those tuples in memory with which t joins, and output the joined tuples. Also mark as "used" all those tuples of R that join with t. After S is exhausted, examine the tuples in main memory, and for each tuple r that is not marked "used," pad r with nulls and output the result.

(ii) We effectively have to perform two nested-loop joins of 500 and 250 blocks, respectively, using 101 blocks of memory. Such a join takes 250 + 500*250/100 = 1500 disk I/O's, so two of them takes 3000. To this number, we must add the cost of sorting the two relations, which takes four disk I/O's per block of the relations, or another 6000. The total disk I/O cost is thus 9000.

(iii) To compute delta(R) using a "hybrid hash" approach, pick a number of buckets small enough that one entire bucket plus one block for each of the other buckets will just fit in memory. The number of buckets would be slightly larger than B(R)/M. On the first pass, keep all the distinct tuples of the first bucket in main memory, and output them the first time they are seen. On the second pass, do a one-pass distinct operation on each of the other buckets. The total number of disk I/O's will be one for M/B(R) of the data and three for the remaining (B(R)-M)/B(R) of the data. Since the data requires B(R) blocks, the total number of disk I/O's is M + 3(B(R) - M) = 3B(R) - 2M, compared with 3B(R) for the standard approach.

Problem 5 (*16 points*)

(i) (a) Assume two relations $R(a,b) = (1,2)$ $S(a,b) = (1,3)$. We have: $\pi_a(R \cup S) = (1),(1)$, but $\pi_a R \cup \pi_a S = (1)$

(b) Assume two relations $R(a,b) = (1,2)$ and $S(a,b) = (1,4)$. We have: $\pi_a(R - S) = (1)$, but $\pi_a R - \pi_a S = \emptyset$

(ii) See Table 1.

(iii) (a) Use an index-scan using the nonclustering index on c. Since V(R,c) = 5000, only one tuple should be retrieved. Filter the retrieved tuple for a=1 and b=3. The expected disk I/O cost is 1.

(b) Use an index-scan using the nonclustering index on b. Since V(R,b) = 1000, 5 blocks should be retrieved. Filter the retrieved blocks for a=1 and c<3. The expected disk I/O cost is 5.

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

Table 1: Costs of physical plans as a function of k.

Problem 6 (*12 points*)

(i) (a) Before event number 12, and after record number 11.

(b) Database: $B \leftarrow 80$; Log: <Abort V>.

(ii) Database: $E \leftarrow 50$,$C \leftarrow 30$,$A \leftarrow 10$
Log: <Abort V>.

(iii) Database: $B \leftarrow 21$,$D \leftarrow 41$,$A \leftarrow 10$, $C \leftarrow 30$
Log: <Abort T>.

Problem 7 (*12 points*)

(i) We can improve the throughput of a system by performing some tasks at CPU while doing I/O activities. Also, we can reduce average waiting time if we can execute a short transaction while executing a long transaction.

(ii) The dependency contains a circular loop, $1 \rightarrow 3 \rightarrow 2 \rightarrow 1$. Therefore, S is not conflict-serializable.

(iii) If $TS(T) < WT(X)$, it means that a newer transaction T' has already written some value on X. When we use a concurrency method with timestamps, we consider a serial schedule where all the actions of each transaction are considered to be done at the starting time of that transaction. Therefore, we must swap T's read action on X with T''s write action on the same element X. However, this is a conflicting swap, and thus we cannot convert the actual schedule to the serial schedule.

(iv) Transaction T_1's validation V_1 succeeds because there is no validated transaction. Transaction T_2's validation V_2 also succeeds because $WS(T_1) \cap RS(T_2) = \{A\} \cap \{B, C\} = \phi$ and $WS(T_1) \cap WS(T_2) = \{A\} \cap \{B\} = \phi$. However, T_3's validation fails because $WS(T_2) \cap WS(T_3) = \{B\} \cap \{B\} = \{B\}$.

Problem 8 (*11 points*)

(i) Transaction T_1 wounds T_2 holding a lock on element on B.

(ii) $(0, 1, P), (0, 2, P), (2, 0, D), (1, 0, R), (0, 2, A), (0, 1, A)$

(iii) If the last log record at site 1 is $<$ *Don't commit* $T >$, the coordinator may or may not have received D message from site 1. If the coordinator has received the D message, it should have reached the global decision to abort T. If the coordinator has not received the message because of the failure of site 1, after a suitable timeout period, it will treat site 1 as if it had sent D message. Therefore, site 1 knows that the coordinator would decide to abort T in both cases, and thus figures out that it should abort T as well.