$R_2(a)$

$R_1(a)$ $W_1(a)$ X. $R_2(a)$ $W_2(a)$ $R_2(c)$ d $R_3(b)$ $R_3(c)$ $W_3(c)$ $R_3(d)$

b, c, d

Database Systems

End Semester Exam

24th April 2017 — Duration: 3 hrs, Marks: 100

1. **No clarifications during the exam.**
2. **Make *reasonable assumptions* and *clearly state* them to answer *ambiguous* questions.**
3. **Show your steps. Be concise and organized.**
4. **Calculators allowed. Sharing of calculators *not* allowed.**

1) **(a)** With respect to transaction serializability, briefly describe what is:
(A) 2PL
(B) Validation test
(C) Timestamp ordering

3

2 → 1

c

[9]

(b) Consider this schedule:

$R_2(a)$, $R_1(a)$, $W_1(a)$, $R_3(b)$, $R_3(c)$, $R_2(a)$, $W_2(a)$, $R_2(c)$, $W_3(c)$, $R_1(d)$, $R_3(d)$, $W_1(d)$

Is this schedule valid in each of the above protocols? Describe why or why not, for each one.

30

[15]

(c)

I. What is the precedence graph for the schedule of part b?

II. Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules ?

[5]

2) The following is a sequence of redo-log records written by two transactions T and U: <START T>; <T, A, 10>; <START U>; <U, B, 20>; <T, C, 30>; <U, D, 40>; <COMMIT U>; <T, E, 50>; <COMMIT T>. Describe the action of the recovery manager, including changes to both disk and the log, if there is a crash and the last log record to appear on disk is:

a. <START U>
b. <COMMIT U>
c. <T, E, 50>
d. <COMMIT T>

20

[10]

3) Consider the following sequence of undo log records: <START S>; <S, A, 60>; <COMMIT S>; <START T>; <T, A, 10>; <START U>; <U, B, 20>; <T, C, 30>; <START V>; <U, D, 40>; <V, F, 70>; <COMMIT U>; <T, E, 50>; <COMMIT T>; <V, B, 80>; <COMMIT V>. Suppose that we begin a non-quiescent checkpoint immediately after one of the following log records has been written (in memory):

a. <S, A, 60>
b. <T, A, 10>
c. <U, D, 40>
d. <T, E, 50>

~~45~~ 30

For each, tell:

i. When the <CKPT> record is written, and

ii. For each possible point at which a crash could occur, how far back in the log we must look to find all possible incomplete transactions.

[16]

4) The Megatron 777 disk has the following characteristics:

1. There are 10 surfaces, with 10,000 tracks per surface.
2. Tracks hold an average of 1000 sectors of 512 bytes each.
3. 20% of each track is used for gaps.
4. The disk rotates at 10,000 rpm.
5. The time it takes the head to move n tracks is 1+0.001*n* milliseconds.

~~30~~ 15

1st part
B

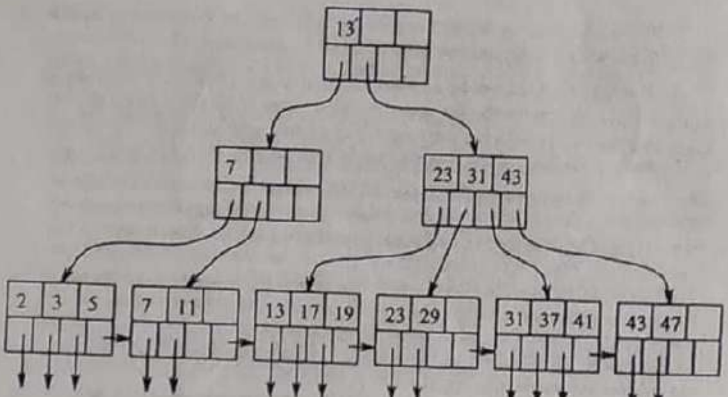
We have a 1 GB sized relation R of 10.000.000 tuples. Each tuple of 100 bytes has several fields, one of which is the sort key field, which may not be a primary key. The machine on which sorting occurs has one Megatron 777 disk and 50 MB of main memory available. Disk blocks are 4096 bytes. How long would it take to sort R using 2-phase, multiway merge sort.

[10]

5) Describe steps, using diagrams if necessary, to execute the following operations on the shown B+ tree:

(a) Lookup all records less than 10
(b) Insert a record with key 6
(c) Delete record with key 7

20



[5]

6) Suppose we are using RAID level 4 (i.e. 1 redundant disk for parity), with 4 data disks and 1 redundant disk. For simplicity, assume blocks are a single byte.

(a) Give the block of the redundant disk if the corresponding blocks of the data disks are: 01010110, 11000000, 00111011, and 11111011.

10

(b) Data disk 1 has failed. Recover the block of that disk if the contents of disks 2 through 4 are: 01010110, 11000000, and 00111011, while the redundant disk holds 11111011.

[10]

7) Suppose blocks hold either 3 records, or 10 key-pointer pairs. As a function of *n*, the number of records, how many blocks do we need to hold a data file and:

10

(a) A dense index? n/3 n/10 (b) A sparse index? n/3 n/30

[10]

8) Write the iterator functions pseudo code for a tuple-based nested-loop join. If B(S) = B(R) = 10,000 and M=1000, what is the number of disk I/O's required for nested-loop join.

[10]

B(S)B(S)

B KB MB GB

10^9 bytes

r = r.getnext()
s = s.getnext()

getnext()
repeat {
if r is null close()
r.close
s = s.getnext
r = r.open()

11111011

else