# Mid Sem 2018, CSE-202, Fundamentals of Database Systems

Name:_____
Roll Number:_____
Maximum Marks: 35 Time:1 hour

---

***Question-1**:* Consider a database that has six elements - A, B, C, D, E and F. The initial values of the elements are:

A = 10, B = 20, C = 30, D = 40, E = 50, F = 60

Let there be three transactions T1, T2 and T3 that modify these elements concurrently.

- T1: A := A-5, B:= B-5, D := D-10
- T2: C := C-5
- T3: E := E-15, F:=F-15

While the elements are being modified by the transactions, the database system crashes. The recovery mechanism depends on the logging scheme we use. In the questions below, we present the contents of the log at the time of crash when the logging scheme used is **steal/force.** (For each log entry we give the relevant data)

< START T1 > ;< T1, A, 10, 5 >; < START T2 >; < T1, B, 20, 15 >; < T2, C, 30, 25 > ;< COMMIT T2 >; < START CKPT >; < START T3 >; < T1, D, 40, 30 > ; < COMMIT T1 >;< T3, E, 50, 35 >; < END CKPT >; < T3, F, 60, 45 >

a) Is the following state of database elements (on disk) possible at the time of crash? Justify your answer. [**1 mark]**
A = 5, B = 15, C = 25, D = 40, E = 35, F = 45
b) What are the values of the database elements on disk (A=?, B=?, C=?, D=?, E=?, F=?) after a successful recovery? [**2 marks]**
c) What different compensation log records will be written in the log during recovery? [**2 marks**]
[Binary marking]

**Answer.**

The steal policy allows the system to write modified blocks to the disk even if the transactions that made those modifications have not all committed. The force policy allows a transaction to force output all modified blocks to the disk when they commit.

(a) No, value of D will be 30 instead of 40. The changes made by transaction T1 will be pushed to the disk even though T1 has not committed due to force policy.
(b) A = 5, B = 15, C = 25, D = 30, E = 50, F = 60. Transactions T3 are undone. T1 and T2 have already committed and their effect would be on disk after recovery.

(c) Compensation log records will be as follows:
<T3, F, 60>
<T3, E, 50>
<T3 Abort>

***Question-2**:* [**5 marks**] Consider the log records below.

<T0 start>
<T0, B, 2000, 2050>
<T0 commit>
<T1 start>
<T1, B, 2050, 2100>
<T1, O4, operation-begin>
<checkpoint {T1}>
<T1, C, 700, 400>
<T1, O4, operation-end, (C,+300)>
<T2 start>
<T2, O5, operation-begin>
<T2, C, 400, 300>
<T2, C, 400>
End of log at crash!

What are the records added during recovery?

**Answer.**

<T2,C,400>
<T2, abort>
<T1, C, 400,700>
<T1, O4, operation-abort>

<T1, B, 2050>

<T1 abort>

[If T2 abort and T1 abort written in correct sequence, then 2 marks.

<T2,C,400> - 1 mark.

Rest operations – 2 marks. If something extra - minus 1 marks. ]

***Question-3**:* [**5 marks**] Consider a database file Employee consisting of disk blocks B1, B2, B3 and B4. B1 contains records o1, o2, o3; B2 contains o4 and o5; B3 contains o6; and B4

contains o7 and o8. Employee File, blocks and data records form a hierarchy of lockable data elements. Show the sequence of lock requests and the final schedule S if 2-phase locking protocol is used with multiple granularity scheme for the following sequence of requests. You may assume all requests occur just before they are needed and all unlock occurs at the end of the transaction. In case of deadlock show partial schedule and discuss why the deadlock will happen. [Hint: Intention locks are used in Multiple granularity scheme or Warning-protocol-based scheduler]

r1(o1); w2(o2); r2(o3); w1(B1)

**Answer.** IS1(Employee), IS1(B1), s1(O1), IX2(Employee), Ix2(B1), X2(O2)

IS2(Employee), IS2(B1), S2(O3)

U2(O3), U2(O2), U2(B1), U2(Employee)

IX1(Employee), X1(B1), U1(O1), U1(B1), U1(Employee),

[2 marks – correct answer at least for couple of branches

5 marks for correct answer]

***Question-4:***

***(a)****:* [**2.5 marks**] Suppose that we are using a RAID level 4 scheme with four data disks and one redundant disk. Assume blocks are single byte. Give the block of the redundant disk if the corresponding blocks of the data disks are: [Binary marking]

01010110, 11000000, 00111011, and 11111011.

**Answer.** *01010110 (Take XOR)*

***(b)****:* [**2.5 marks**] Suppose disks have a failure rate of fraction F per year and it takes H hours to replace a disk. If we use mirrored disks, what is the mean time to data loss, as a function of F and H? [Binary marking]

**Answer.** To compute the mean time to failure, we can compute the probability that the system fails in a given year. The MTTF will be the inverse of that probability. Note that there are 8760 hours in a year. The system fails if the second disk fails while the first is being repaired. The probability of a failure in a year is 2F. The probability that the second disk will fail during the H hours that the other is being prepared is H/8760. Thus, the probability of a failure in any year is 2FH/8760, and the MTTF is 4380/FH.

***(c)**:* [**3 marks**] Suppose a record has the following field in this order: A character string of variable length 40, an integer of 2 bytes, and another character string of variable length 40. Show a record structure using variable length scheme as discussed in the class if the contents are – Siddharth Dawar, 40, CSE. Assume that each character takes 1 byte and the record starts at address 1000. [Binary marking]

**Answer.** | offset(11), 15 | 40 | offset(26),3 | 1 byte null delimiter |Siddharth Dawar| CSE|

1000 1004 1006 1010 1011 1026 1029

***Question-5**:* [**2 marks**] Consider the following schedule:

S1: $r_1(A)$; $r_2(B)$; $w_1(C)$; $r_3(D)$; $r_4(E)$; $w_3(B)$; $w_2(C)$; $w_4(A)$; $w_1(D)$;

Assume that shared locks are requested immediately before each read action, and exclusive locks are requested immediately before every write action. Also, unlocks occur immediately after the final action that a transaction executes. What is the waits-for graph for the above schedule? Does a deadlock happen? [Binary marking]

**Answer.** Waits-for graph: $T_3$ -> $T_2$ -> $T_1$ <- $T_4$ , $T_1$ -> $T_3$

There is a cycle in the Waits-for graph. Hence, deadlock will happen. $T_1$ must abort and relinquish its locks. At that time, $T_2$ and $T_4$ can each get the locks they need. When $T_2$ finishes, $T_3$ can proceed.

***Question-6**:* [**5 marks**] Consider the following sequence of actions , listed in the order they are submitted to DBMS:

For the Strict 2 PL with timestamps used for deadlock prevention (assume wait-die policy), Describe the sequence of actions showing how the concurrency control mechanism handles the above sequence. Assume that the timestamp of transaction Ti is i. For lock-based concurrency control mechanisms, add lock and unlock requests to the sequence of actions as per the protocol. If a transaction is blocked, assume that its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed sequence) of an unblocked transaction. On the other hand, if a transaction rollback, transaction is assumed to come up with a newer timestamp.[Binary marking]

**Answer.** Assume we use Wait-Die policy.

T1 acquires shared-lock on A.

When T2 asks for an exclusive lock on A, since T2 is younger, it will be rolled back.

T3 now gets exclusive-lock on B.

When T1 also asks for an exclusive-lock on B which is still held by T3, since T1 is older than T3, T1 will be blocked waiting.

T3 now finishes write, commits and releases all the lock.

T1 wakes up, acquires the lock, proceeds and finishes.

T2 now can be restarted successfully.

***Question-7**:* [**5 marks**] Consider the following schedule:

| T1 | T2 |
|---|---|
| Read(A) | |
| | Write(B) |
| Read(B) | |

Assume the following order on the data items: A -> B. Is the schedule possible under the tree protocol? If yes, add lock and unlock instructions. [Binary marking]

**Answer.** Yes, it is possible under the tree protocol.

| T1 | T2 |
|---|---|
| Lock-X(A) | |

| | |
|---|---|
| Read(A) | |
| | Lock-X(B) |
| | Write(B) |
| | Unlock(B) |
| Lock-X(B) | |
| Read(B) | |
| Unlock(A) | |
| Unlock(B) | |