Question1. (2013 & 2014)

REDO Logging

After a system crash, the **redo-log** using non-quiescent checkpointing contains the following data:

```
< START T1 >
< T1, A, 10 >
< START T2 >
< T2, B, 5 >
< T1, C, 7 >
< START T3 >
< T3, D, 12 >
< COMMIT T1 >
< START CKPT ???? >
< START T4 >
< T2, E, 5 >
< COMMIT T2 >
< T3, F, 1 >
< T4, G, 15 >
< END CKPT >
< COMMIT T3 >
< START T5 >
< T5, H, 3 >
< START CKPT ???? >
< COMMIT T5 >
```

(a) What are the correct values of the two <START CKPT ????> records? You have to provide two correct values for the two ????s.

Solution:

First START CKPT: < START CKPT (T2, T3) >

Second START CKPT: < START CKPT (T4, T5) >

(b) Indicate and **explain** what fragment of the log the recovery manager needs to read.

Solution:

Since the second START CKTP does not have an associated END CKPT, we cannot be sure that committed transactions prior to the start of this checkpoint had their changes written to disk. Thus, we must search for the previous checkpoint. In the previous START CKPT, T2 and T3 were the two active transactions. Both transactions committed and must thus be redone. T2 was the first one to start. The recovery manager must thus read the log record starting from < START T2 > and must read until the end of the log file.

(c) Assuming that the two < START CKPT ??? > records are correctly stored in the log, according to your answer above, show which elements are recovered by the redo recovery manager and compute their values after recovery.

Solution:

We must redo the changes made by all *committed* transactions there were either active during the first START CKPT or that started after that point. T2 and T3 were active during the first START CKPT and comitted. T4 and T5 started after the checkpoint but only T5 committed. We must thus redo the changes by T2, T3, and T5. Elements B, D, E, F, and H are thus recovered. Their values after recovery are as follows:

- B=5
- D=12
- E=5
- F=1
- H=3

Question 2.

Consider the content of the following **undo log**:

LSN1	<START T1>
LSN2	<T1 X 5>
LSN3	<START T2>
LSN4	<T1 Y 7>
LSN5	<T2 X 9>
LSN6	<START T3>
LSN7	<T3 Z 11>
LSN8	<COMMIT T1>
LSN9	<START CKPT(T2,T3)>
LSN10	<T2 X 13>
LSN11	<T3 Y 15>
	*C*R*A*S*H*

- Show how far back in the recovery manager needs to read the log. Write below the earliest LSN that the recovery manager reads.

  Solution:
  LSN3

- Show below the actions of the recovery manager during recovery:

  Solution:

$$
\begin{aligned}
Y &= 15 \\
X &= 13 \\
Z &= 11 \\
X &= 9
\end{aligned}
$$

- What is the value of X at the end of the recovery ?

  Solution:
  9

Question 3.

A database has four elements, A, B, C, and D. Assume that the following is a normal sequence of undo log records, using non-quiescent checkpointing:

1	<start T1>	6	<start T3>	12	<T4,C,7>
2	<T1,B,40>	7	<commit T1>	13	<T3,A,22>
3	<start T2>	8	<T3,B,12>	14	<commit T4>
4	<T2,A,56>	9	<commit T2>	15	<T3,A,99>
5	<T2,C,34>	10	<T3,D,89>	16	<commit T3>
		11	<start T4>		

(a) When is the *latest time* for transaction T1, T2 that "dirty data" can be flushed onto disk (ie, the time Output(X) for data X can be performed)? [2 points]

For T1: Output(s) B before Log Line 7.
For T2: Output(s) A and C before Log Line 9.

(b) Suppose we start checkpointing right after Log 5, indicate where and what the start checkpointing record would look like. Then, indicate where and what the *earliest* end checkpoint record would look like. [2 points]

Between Log Line 5 and 6, <START Checkpoint(T1, T2)>
Between Log Line 9 and 10, <END Checkpoint>

(c) Continue from (b). Suppose the system crashes right after Log 14 and the end checkpoint has been written out to disk. What is the contents of the earliest log line we must examine? And which transaction records do we need to undo in sequence? [4 points]

Earliest Log File Contents: <START Checkpoint(T1, T2)> Log lines 5 and 6, Transaction Sequence: <T3,A,22>, <T3,D,89>, <T3,B,12>

Question 4.

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 ⋈ S

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

Question 5.

Consider an indexed sequential file consisting of 10,000 blocks. Each block contains 10 fixed sized records. Each key value found in the file is unique. For this problem, assume that:

- Pointers to blocks are 10 bytes long.
- Pointers to records are 20 bytes long.
- Index blocks are 5000 bytes (in addition to the header).
- Search keys for file records are 10 bytes long.

(a) How many blocks do we need to hold a sparse one-level, primary index? (5 points).

ceil(10000*(10+10) / 5000) = 40

Explanation: We need to store search key per block (10 bytes per block), and one block pointer (10 bytes) that points to the first block. Block pointers per block are not required since the blocks are contiguous, and so a block pointer can be computed using the offset from the first block pointer.
Number of blocks: **40**

(b) In (a), how many disk I/Os do we need to find and retrieve a record with a given key at the worst case? (5 points)

Sparse index has pointers for each block, not records. In part a, 21 blocks result. In order to search for the correct index, $\log_2 40$ I/O plus a final I/O to retrieve data;
$\log_2 40 + 1 = 6.32$

(c) Suppose you now construct a one-level, dense secondary index. Compute its minimum size in blocks. (5 points)

ceil(10000 * 10 / floor(5000 / (10+20))) = 603

Note: Taking floor as above is necessary to make sure that records do not span blocks. However, students who assumed that records span blocks, and computed number of blocks as 10000 * 10 / (5000 / (10+20)) = 600 is considered correct.
Number of blocks: **603**

Question 6.

Consider two relations $R(a, b)$ and $S(b, c)$ with the following statistics:

$T(R) = 10,000$, $B(R) = 1,000$ (each block contains 10 tuples),
$V(R, b) = 200$ (number of distinct values of attribute b in R),

$T(S) = 6,000$, $B(S) = 1,500$ (each block contains 4 tuples),
$V(S, b) = 100$ (number of distinct values of attribute b in S),
$V(S, c) = 20$ (number of distinct values of attribute c in S) and $c > 100$.

Also, we assume the number of available memory blocks is $M = 101$.

Please answer the following questions:

(i) Estimate the number of tuples in $\sigma_{c=150}(S)$ (2 points)

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

(ii) Estimate the number of tuples in $R \bowtie \sigma_{c>25}(S)$. (2 points)

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

Question 7.

X from old to new. We consider recovery using this undo/redo log.

1. <T2 start>
2. <T2, B, 10, 11>
3. <T1 start>
4. <T2 commit>
5. <T1, A, 20, 21>
6. <Checkpoint start; Active= {T1}>
7. <T3 start>
8. <T3, C, 30, 31>
9. <T4 start>
10. <T4, B, 40, 41>
11. <T4 commit>
12. <Checkpoint end>
13. <T3, D, 50, 51>
14. <Checkpoint start; Active= {T1, T3}>
15. <T1, C, 31, 32>
16. <T5 start>
17. <T5, D, 51, 52>
18. <T3 commit>
19. <T6 start>
20. <T6, C, 32, 33>
21. <T5 commit>
22. System failed

(i) List all possible values of A, B, C and D. That is, what are the possible data values on the disk at the point of failure (after action 21)? (3 points)

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

(ii) During recovery, what are the transactions that need to be undone? (3 points)

(ii) T1, T6.

(iii) During recovery, what are the transactions that need to be redone? (3 points)

(iii) T3, T4, T5

(iv) What are the values of A, B, C, D after recovery? Explain why? (3 points)

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

Question 8.

Consider the following database about word occurrences in Webpages:

```
Webpage(url, author)
Occurs(url, wid)
Word(wid, text, language)
```

where:

- `Webpage.url` and `Word.wid` are keys.
- `Occurs.url` and `Occurs.wid` are foreign keys to `Webpage` and `Word` respectively.

Assume the following statistics

$$\begin{aligned} T(\texttt{Webpage}) &= V(\texttt{Occurs}, \texttt{url}) = 10^9 \\ T(\texttt{Occurs}) &= 10^{12} \\ T(\texttt{Word}) &= V(\texttt{Occurs}, \texttt{wid}) = 10^6 \\ V(\texttt{Webpage}, \texttt{author}) &= 10^7 \\ V(\texttt{Word}, \texttt{language}) &= 100 \end{aligned}$$

Assume ten records can be fit in one block, hence $B(\texttt{Webage}) = T(\texttt{Webpage})/10$ and similarly for all other tables.

(a) (5 points) Consider the following plan:

$$(\sigma^{\texttt{index-lookup}}_{\texttt{author='John'}}(\texttt{Webpage}) \bowtie^{\texttt{index-join}}_{\texttt{url=url}} \texttt{Occurs})$$

$$\bowtie^{\texttt{main-memory-hash-join}}_{wid=wid} \sigma^{\texttt{index-lookup}}_{\texttt{language='French'}}(\texttt{Word})$$

Compute the cost of the plan in each of the following cases:

1. We have the following indexes:

`Webpage.url`	=	primary index
`Webpage.author`	=	secondary index
`Occurs.url`	=	secondary index
`Occurs.wid`	=	primary index
`Word.wid`	=	primary index
`Word.language`	=	secondary index

Solution:

Unclustered Index Lookup on Author	=	10^2
Unclustered Index Join	=	$10^2 * 10^3$
Unclustered Index Lookup on Language	=	10^4
Main Memory Hash Join	=	0
Total	=	$10^2 + 10^5 + 10^4$

2. We have the following indexes:

`Webpage.url`	=	secondary index
`Webpage.author`	=	primary index
`Occurs.url`	=	primary index
`Occurs.wid`	=	secondary index
`Word.wid`	=	secondary index
`Word.language`	=	primary index

Solution:

Clustered Index Lookup on Author	=	10
Clustered Index Join	=	$10^2 * 10^2$
Clustered Index Lookup on Language	=	10^3
Main Memory Hash Join	=	0
Total	=	$10 + 10^4 + 10^3$

Question 9.

Consider relationsr(A,B) ands(B,C). Assume thatr contains 2000 tuples, and thats contains 5000 tuples. We want to compute v = r JOINr:B=s:Bs.

(a) Without any further assumptions, what is the maximum number of tuples that vmay contain? 3

(b) Now assume that we know that V (B,r) = 500. (That is, inr the attributeB takes on 500 different values.) What is now a reasonable estimate on the size ofv? 2

(b) Finally, assume we know that s satisfies the functional dependencyB ->C (i.e., B is a key). What is now a reasonable estimate on the size ofv? 2

a) Assume all tuples are same

5000 x 2000

b) $\min\left(\frac{2000}{50}, 5000\right)$

$= \min(40, 5000)$

$= 40$

c) As there exists a dependency between B & C in S, so atmost 2000 tuples can be considered as same.

Question 10.

Consider a relation that is fragmented horizontally by plant-number:

employee (name, address, salary, plant-number)

Assume that each fragment has two replicas: one stored at the New York site and one stored locally at the plant site. Describe a good processing strategy for the following queries entered at the San Jose site.

a. Find all employees at the Boca plant.

b. Find the average salary of all employees.

c. Find the highest-paid employee at each of the following sites: Toronto, Edmonton, Vancouver, Montreal.

d. Find the lowest-paid employee in the company

a. i. Send the query $\Pi_{name}(employee)$ to the Boca plant. ii. Have the Boca location send back the answer.
b. i. Compute average at New York. ii. Send answer to San Jose.
c. i. Send the query to find the highest salaried employee to Toronto, Edmonton, Vancouver, and Montreal. ii. Compute the queries at those sites. iii. Return answers to San Jose.
d. i. Send the query to find the lowest salaried employee to New York. ii. Compute the query at New York. iii. Send answer to San Jose.

Question 11.

6. **(12 points) Imagine that you are given the following set of training examples. All the features are Boolean-valued.**

F1	*F2*	*F3*	*Result*
T	T	F	+
F	T	T	+
T	F	T	-
F	T	F	-
F	F	T	-

How would a Naive Bayes approach classify the following *test* example?
Be sure to show your work.

F1 = T *F2* = F *F3* = F

P(F1|+) = P(F1=T|+) = 1/2 = 0.5
P(F2|+) = P(F2=F|+) = 0/2 = 0.0
P(F3|+) = P(F3=F|+) = 1/2 = 0.5

P(F1|-) = P(F1=T|-) = 1/3 =

P(F2|-) = P(F2=F|-) = 2/3 =
P(F3|-) = P(F3=F|-) = 1/3 =

Plugging thse into the above eqn.

P(vj = +)* P(F1=T| +) P(F2=F| +) P(F3=F | +) = (2/5)*(1/2)*(0.0)*(1/2) = 0

P(vj = -)* P(F1| -) P(F2| -) P(F3 | -) = (3/5) * (2/3)*(2/3)*(1/3) = 2/45 = 0.044

The correct hypothesis therefore is -

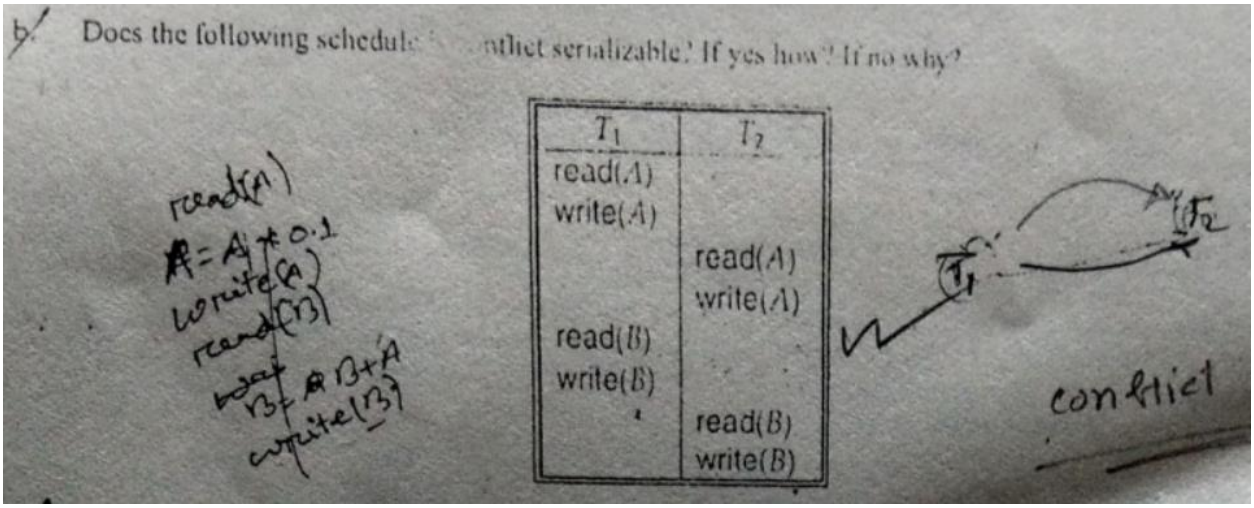
Question 12.

Write a serial schedule for the following
Let T_1 transfer \$50 from A to B, and T [illegible]ansfer 10% of the balance from A to B

- A serial schedule where T_2 is followed by T_1

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	

Question 13.



- Schedule 3 can be transformed into Schedule 6, a serial schedule where T_2 follows T_1, by series of
  - Therefore Schedule 3 is conflict .

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

Schedule 3

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

Schedule 6

Question 14.

4 (a) List possible types of failure in a distributed system and explain 2PC protocol to handle transaction atomicity despite the failures. 2

a. The types of failure that can occur in a distributed system include
   i. Site failure.
   ii. Disk failure.
   iii. Communication failure, leading to disconnection of one or more sites from the network.

a. A site can abort a transaction T (by writing an <**abort** T> log record) only under the following circumstances:

   i. It has not yet written a <**ready** T> log-record. In this case, the co-ordinator could not have got, and will not get a <**ready** T> or <**commit** T> message from this site. Therefore only an abort decision can be made by the co-ordinator.
   ii. It has written the <**ready** T> log record, but on inquiry it found out that some other site has an <**abort** T> log record. In this case it is correct for it to abort, because that other site would have ascertained the co-ordinator's decision (either directly or indirectly) before actually aborting.
   iii. It is itself the co-ordinator. In this case also no site could have committed, or will commit in the future, because commit decisions can be made only by the co-ordinator.

b. A site can commit a transaction T (by writing an <**commit** T> log record) only under the following circumstances:

   i. It has written the <**ready** T> log record, and on inquiry it found out that some other site has a <**commit** T> log record. In this case it is correct for it to commit, because that other site would have ascertained the co-ordinator's decision (either directly or indirectly) before actually committing.
   ii. It is itself the co-ordinator. In this case no other participating site can abort/ would have aborted, because abort decisions are made only by the co-ordinator.