CS 245, Database Management Systems
Endsem Exam, Winter 2022-2023
Department of Computer Science and Engineering
IIT Guwahati
Time: Three hours

1. Consider the following two relational schemas:
   Schema 1: $R(A, B, C)$
   Schema 2: $R1(A, B), R2(A, C)$
   Now suppose that the only dependencies (functional or multivalued) that hold on the relations in these schemas are $BC \rightarrow A$, $A \rightarrow C$, and all dependencies that follow from these two.

   (a) Which is the strongest normal form among 3NF, BCNF, 4NF, or none of these, satisfied by Schema 1. Give reasons. (5)

   Solution: The strongest normal form is 3NF. It is easy to see that the candidate keys for R are AB and BC. Schema 1 is not in BCNF since $A \rightarrow C$ is a dependency and A is not a superkey. It is in 3NF since C is contained in a candidate key BC.

   (b) Which is the strongest normal form among 3NF, BCNF, 4NF, or none of these, satisfied by Schema 2. Give reasons. (5)

   Solution: The strongest normal form is 4NF. Both $R1$ and $R2$ are in BCNF as they have only two attributes. Since the only MVDs are FDs, Schema 2 is in 4NF.

2. Consider the instructor relation shown below.

```
instructor

ID    name                 dept_name            salary
----- -------------------- -------------------- ----------
10101 Srinivasan           Comp. Sci.             65000.00
12121 Wu                   Finance                90000.00
15151 Mozart               Music                  40000.00
22222 Einstein             Physics                95000.00
32343 El Said              History                60000.00
33456 Gold                 Physics                87000.00
45565 Katz                 Comp. Sci.             75000.00
58583 Califieri            History                62000.00
76543 Singh                Finance                80000.00
76766 Crick                Biology                72000.00
83821 Brandt               Comp. Sci.             92000.00
98345 Kim                  Elec. Eng.             80000.00
```

(a) Construct a bitmap index on the attribute salary, dividing salary values into four ranges: below 50,000, 50,000 to below 60,000, 60,000 to below 70,000, and 70,000 and above.

(4)

Solution: Here are the bitmaps for salary intervals, with S_1, S_2, S_3, S_4 representing the given intervals in the same order.

S_1	0	0	1	0	0	0	0	0	0	0	0	0
S_2	0	0	0	0	0	0	0	0	0	0	0	0
S_3	1	0	0	0	1	0	0	1	0	0	0	0
S_4	0	1	0	1	0	1	1	0	1	1	1	1

(b) Consider a query that requests all instructors in the Finance department with salary of 70,000 or more. Outline the steps in answering the query efficiently, using one of the bitmaps above along with an additional one on a different attribute.

(6)

Solution: Along with the above bitmaps for salary intervals, construct bitmaps for the department_name attribute. Then take the intersection of the Finance department bitmap and S_4 bitmap of salary to get the bitmap for the answer to the query.

3. Let r and s be relations with no indices, and assume that the relations are not sorted but are stored on contiguous blocks on the disk. Assume r and s can be stored on disk using b_r and b_s blocks.

(a) Assuming infinite memory, what is the lowest-cost way (in terms of I/O operations, *i.e.*, block transfers, and *not* the time taken) to compute $r \bowtie s$? (5)

Solution: We can store the entire smaller relation in memory, read the larger relation block by block and perform nested loop join using the larger one as the outer relation.

(b) What is the number of disk I/O operations and the amount of memory required for this algorithm? Ignore the block transfers involved in storing the result on disk. Assume just one block of memory holds the current output and it is written to disk when it is full.

(5)

Solution: The number of I/O operations is equal to $b_r + b_s$, and the memory requirement is $\min(b_r, b_s) + 2$ blocks. The two blocks include one block to hold the current block of the bigger relation and one to hold the current output.

4. Let the consistency requirement on the two following transactions be $A = 0 \vee B = 0$ with $A = B = 0$ the initial values.

T_1:

```
read(A);
read(B);
if A = 0 then B := B + 1;
write(B).
```

T_2:

```
read(B);
read(A);
if B = 0 then A := A + 1;
write(A).
```

(a) Show that every serial execution involving these two transactions preserves the consistency of the database.

(5)

Solution: For the serial schedule $T_1; T_2$ the final values of the data items are $A = 0, B = 1$. For the other serial schedule $T_2; T_1$ the final values are $A = 1, B = 0$. In both cases one of the data items is zero, and hence the consistency requirement holds.

(b) Is there a concurrent execution of T_1 and T_2 (other than the two serial ones) that produces a serializable schedule? Justify your answer. Give the equivalent serial schedule if the answer is yes.

(5)

Solution: No concurrent schedule containing T_1 and T_2 is serializable with respect to the consistency requirement $A = 0 \vee B = 0$. To see this, suppose the schedule starts with T_1 : **read**(A). Then the final value of B is set to 1 irrespective of the rest of the schedule, since T_1 is the only transaction that writes B. Now in a concurrent schedule, T_2 must start before the last instruction T_1 : **write**(B) is executed. Therefore, the value read by T_2 : **read**(B) is 0 and hence the final value of A is set to 1, since T_2 is the only transaction that writes A. Thus $A = 1 \wedge B = 1$ holds at the end of the schedule.

The case where the schedule starts with $T_2 : \textbf{read}(B)$ is symmetric.

5. Consider the following transactions where $r_i(A)$ and $w_i(A)$ correspond to a read and write respectively on data item A by transaction T_i.

$$\begin{aligned} &T_1 : r_1(A); w_1(A); w_1(C). \\ &T_2 : r_2(A); r_2(C); r_2(B); w_2(B). \\ &T_3 : r_3(B). \end{aligned}$$

Answer the following questions for the following schedule S involving the three transactions above.

$$r_1(A); r_2(A); r_3(B); w_1(A); r_2(C); r_2(B); w_2(B); w_1(C);$$

(a) What is the precedence graph for the schedule S? (5)

Solution: The precedence graph for S has the set of vertices $V = \{T_1, T_2, T_3\}$ and the following directed edges:

$$\begin{aligned} T_2 \rightarrow T_1 &\qquad \text{(since } r_2(A) \text{ comes before } w_1(A)) \\ T_3 \rightarrow T_2 &\qquad \text{(since } r_3(B) \text{ comes before } w_2(B)) \end{aligned}$$

Note that the fact that $r_2(C)$ comes before $w_1(C)$ does not add any new edge.

(b) Is the schedule S conflict-serializable and why? If the answer is yes, give an equivalent serial schedule. (5)

Solution: Since the precedence graph for S has no cycle, S is conflict-serializable. An equivalent serial schedule is given by $T_3; T_2; T_1$.

6. In the sequence of actions below involving transactions T_1, T_2, T_3 and T_4, the actions $r_i(A)$ and $w_i(A)$ correspond to a read and write respectively on data item A by transaction T_i. Assume that shared locks are requested immediately before each read action, and

exclusive locks are requested immediately before each write action. Upgrading of locks from shared to exclusive is also allowed. Also assume that unlocks occur immediately after the final action that a transaction executes.

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

(a) Which actions in the above sequence have to wait for a data item locked by another transaction? Does deadlock occur in this scenario? Justify your answer. (5)

Solution:

1. The action $w_3(B)$ by T_3 has to wait for T_2 to release the S-lock on B.
2. The action $w_2(C)$ by T_2 has to wait for T_1 to release the X-lock on C.
3. The action $w_4(A)$ by T_4 has to wait for T_1 to release the S-lock on A.
4. Finally, the action $w_1(D)$ by T_1 has to wait for T_3 to release the S-lock on D.

There is a deadlock in this case because of the cycle $T_3 \to T_2 \to T_1 \to T_3$ in the wait-for graph, which has edges $\{T_3 \to T_2, T_2 \to T_1, T_4 \to T_1, T_1 \to T_3\}$.

(b) For the same sequence describe what happens to each operation under the wait-die deadlock avoidance scheme. Assume that the order of deadlock-timestamps is T_1, T_2, T_3, T_4, *i.e.*, T_1 is the oldest and T_4 the youngest transaction. (5)

Solution:

1. When T_3 attempts the action $w_3(B)$ it is rolled back as T_3 is younger than T_2.
2. Similarly, T_2 is rolled back when it attempts the action $w_2(C)$ as it is younger than T_1.
3. T_4 is also rolled back when it attempts the action $w_4(A)$ as it is younger than T_1.
4. When T_1 attempts the action $w_1(D)$ it *does not* have to wait for T_3 since T_3 has been rolled back and has released the lock on D.

7. Consider a database organized in terms of the following hierarchy of objects: The database itself is an object (D), and it contains two files ($F1$ and $F2$), each of which

contains 1000 pages ($P1, \dots, P1000$ and $P1001, \dots, P2000$, respectively). Each page contains 100 records, and records are identified as $p : i$, where p is the page identifier and i is the slot of the record on that page.

Multiple-granularity locking is used, with S, X, IS, IX and SIX locks, and database-level, file-level, page-level and record-level locking. For each of the following operations, indicate the sequence of lock requests that must be generated by a transaction that wants to carry out (just) these operations. The answer should reflect the minimum amount of locking required and maximum allowable concurrency.

(a) Read record $P1200 : 5$. (5)

Solution: IS on D; IS on $F2$; IS on $P1200$; S on $P1200 : 5$.

(b) Read all (records on all) pages in file $F1$. (5)

Solution: IS on D; S on $F1$.

8. Suppose the timestamp-ordering protocol is used for concurrency control and consider the following sequence of events.

$$st_1; st_2; st_3; r_1(A); r_2(B); w_1(C); r_3(B); r_3(C); w_2(B); w_3(A).$$

Here st_i means the transaction T_i starts. The events $r_i(A)$ and $w_i(A)$ denote the read and write operations by transaction T_i on data item A. The sequence shows the intended order of execution of the events and the scheduler allocates timestamps to transactions in the order of their start time. What happens as the above sequence executes? Mention which operations succeed and which ones fail, leading to rollback of the corresponding transaction. Give proper justification for each operation. (10)

Solution:

1. We have $TS(T_1) < TS(T_2) < TS(T_3)$ from the order in which the transactions start.
2. The initial event sequence $r_1(A); r_2(B); w_1(C)$; executes without any problems since they are the first accesses to the corresponding data item. The timestamps $R\text{-}TS(A)$, $R\text{-}TS(B)$ and $W\text{-}TS(C)$ are set to $TS(T_1)$, $TS(T_2)$ and $TS(T_3)$ respectively.
3. The event $r_3(B)$ is reading an item not written before and can proceed. $R\text{-}TS(B)$ is set to $TS(T_3)$, since T_3 is younger than T_2.

4. The event $r_3(C)$ is reading an item written by an older transaction T_1, and hence there is no problem. $R\text{-}TS(C)$ is set to $TS(T_3)$.
5. The event $w_2(B)$ is trying to write an item read by a younger transaction T_3 earlier and hence T_2 is rolled back.
6. The event $w_3(A)$ is writing an item read by a older transaction T_1 earlier, and there is no problem..

9. The following is a sequence of log records written by two transactions T_1 and T_2:

   <T_1 **start**>
   <T_1, A, 10, 11>
   <T_2 **start**>
   <T_2, B, 20, 21 >
   <T_1, C, 30, 31>
   <T_2, D, 40, 41>
   <T_2 **commit**>
   <T_1, E, 50, 51>
   <T_1 **commit**>

   Describe the action of the recovery manager, including changes to the disk, if there is a crash and the last log record to appear on the stable storage is < T_2 `commit` >. You do not have to mention changes to the log.

   (10)

Solution:

The recovery starts by traversing the log from the beginning and redoing every action while also placing the corresponding transaction on the undo list.

When < T_2 `commit` > is encountered, T_2 is removed from the undo list. Since T_1 remains on the undo list when the log ends, the actions of T_1 are undone moving backwards. The net effect is to set B to 21 and D to 41 while redoing T_2 and to set C to 30 and A to 10 while undoing T_1.