NATIONAL UNIVERSITY OF SINGAPORE

CS3223 – DATABASE SYSTEMS IMPLEMENTATION

(Semester 2: AY2020/21)

Time Allowed: 120 minutes

INSTRUCTIONS TO STUDENTS

1. Please write your **Student Number** only. **DO NOT** write your name.
2. This assessment paper consists of **TWO** Sections and comprises **TWELVE (12)** printed pages. Section **A** has **TWO (2)** questions, and Section **B** has **TWENTY-ONE (21)** questions. The maximum score is **THIRTY** (30).
3. Answer **ALL** questions. Write your answers within the **space provided** in this booklet.
4. This is an **OPEN BOOK** assessment.
5. You are allowed to use a calculator. You are allowed to bring your laptop/iPad. You are **NOT ALLOWED** to have network access during the assessment.

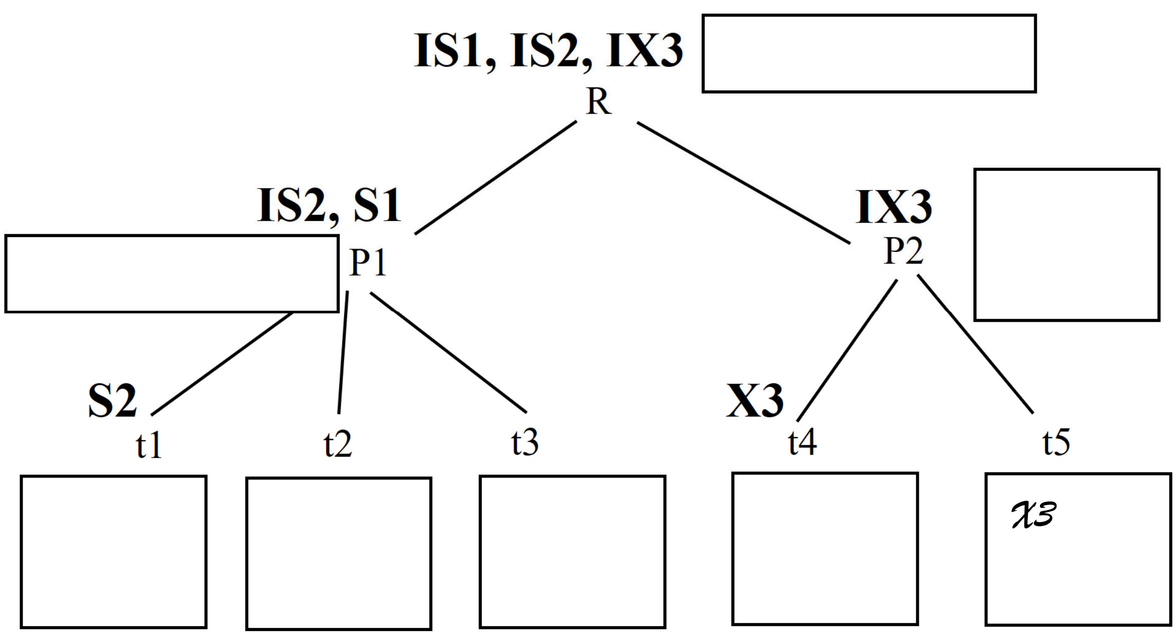
Student Number:												

EXAMINER'S USE ONLY	
Questions	**Marks**
Section A (9 marks)	
Q1 (3 marks)	
Q2 (6 marks)	
Section B (21 marks)	
Total (30 marks)	

Section A (9 marks)

1. **(3 marks)** A DBMS uses the multi-granularity locking scheme with three layers as shown in the figure below: R is a table, P_1 and P_2 are two pages, and t_1-t_5 are tuples stored in these pages as reflected in the hierarchy. The figure shows the state of the system at a particular time when FOUR transactions are active. For example, transactions T_1 and T_2 hold IS locks on table R, transaction T_3 holds an IX lock on R. **At this moment, transaction T_4 does not hold any locks yet.** Indicate in the boxes in the figure (besides the objects) the possible **NEXT** lock actions that each of the **FOUR** transactions can possibly take. For example, transaction T_3 could next lock t_5 on X mode, so you should write X_3 (shown as *X3*) in that box. Note that this same box could have another action, say X_n, if another transaction T_n could also get this lock in its next step. Note that transactions T_3 and T_n could not both hold the X lock on t_5; a box with two such transactions simply means that one or the other could get the lock next. Furthermore, note that if a transaction could possibly hold different kinds of locks on an object in its next step, you should list all of them. Finally, a transaction could have multiple next actions on different objects. Again, you should list all of them in the corresponding boxes.

   DO NOT show entries that are not useful even though they do not create a conflict. For example, it does not make sense for transaction T_1 to request for an S lock or IS lock on t_2, so there should be no IS_1 or S_1 in the box for t_2.



2. **(6 marks)** Consider a DBMS that uses **undo/redo logging** with **non-quiescent checkpoints** for recovery and uses the **basic two-phase locking protocol** (i.e., 2PL with the locking rules 1, 2 and 3 as described in our lecture) for concurrency control. Suppose the actions of three transactions T_1, T_2 and T_3 are interleaved resulting in the following sequence of log records just right before a system crash.

< T_1, Start >
< T_2, Start >
< T_1, X, 0, 10 >
< T_2, **?**, **?**, 20 > ⇐ **incomplete log record IC1**
< Checkpoint Start>
< **?**, Y, **?**, 30 > ⇐ **incomplete log record IC2**
< T_3, Start>
< T_1, Y, 30, 40 >
< Checkpoint End>
< T_1, commit >
< T_3, X, **?**, 100 > ⇐ **incomplete log record IC3**
SYSTEM CRASH AT THIS POINT !!!!!

In the sequence, the log entries follow the format <Transaction ID, Object, Old Value, New Value>. Suppose the database has only two objects X and Y, and the initial values of these objects are both 0 prior to the start of the log file shown above. Moreover, three of the log records are incomplete in the sense that some information are missing. For example, for incomplete log IC1, the object and its old value are missing.

A. Suppose it is possible for a transaction to write an object multiple times (each generating a log record). Identify the missing values of the incomplete log records. You should identify all possible combinations of IC1, IC2 and IC3 that result in a correct log file. Complete your answer in the table below. The first column shows an example (not necessarily a correct answer): suppose a correct sequence of logs should include IC1 being < T_2, **X**, **0**, 20 >, IC2 being < **T_1**, Y, **0**, 30 >, and IC3 being < T_3, X, **0**, 100 >, then fill in "**X, 0**" in the first row of the column, "**T_1, 0**" in the second row, and "**0**" in the third row. Note that while the table allows **up to** 4 possible answers, this **does not mean** that there are 4 answers.

	SAMPLE	**Answer I (Log I)**	**Answer II (Log II)**	**Answer III (Log III)**	**Answer IV (Log IV)**
IC1	**X, 0**				
IC2	**T_1, 0**				
IC3	**0**				

B. Based on your possible log files in the previous part, list all possible values of A and B on disk before recovery. Complete your answer in the table below. Again, the number of possible answers depend on your answer in (A) (though 4 are given below).

	X	Y
Log I		
Log II		
Log III		
Log IV		

C. Suppose each transaction always reads an object right before writing it. Did any of the possible log files lead to any problem? Hint: You should consider whether you can recover the database to a consistent state.

	Problem (Yes/No)	If YES, explain the problem
Log I		
Log II		
Log III		
Log IV		

Section B (21 marks)
This section contains TWENTY-ONE (21) questions. Fill in your answers in the following table.

Q1			Q8			Q15	
Q2			Q9			Q16	
Q3			Q10			Q17	
Q4			Q11			Q18	
Q5			Q12			Q19	
Q6			Q13			Q20	
Q7			Q14			Q21	

1. Consider a relation R(a, b, c) that may contain duplicate tuples. Which of the following equivalences is/are NOT true? Note that all operations should be assumed to be BAG operations.
   A. $\pi_{a,b}(R) \bowtie \pi_{b,c}(R) = R$
   B. $\pi_{a,b}(\delta(R)) = \delta(\pi_{a,b}(R))$
   C. $R \cap \delta(R) = \delta(R)$
   D. A & B E. A & C F. B & C G. A, B & C.

2. Consider the following statistics for three relations R, S, U.

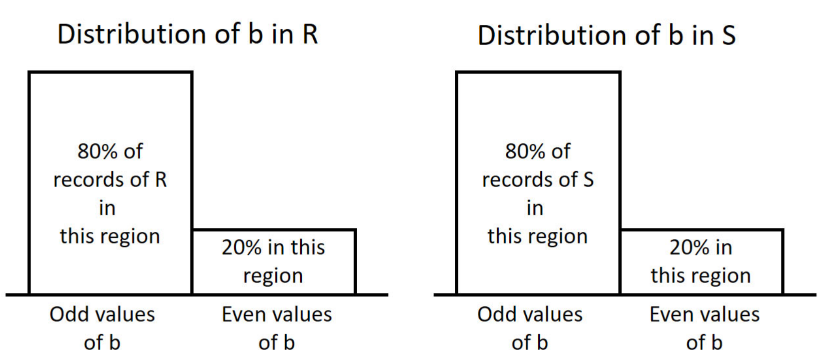
R(a, b, c)	S(b, c, d)	U(b, e)
T(R) = 1000	T(S) = 2000	T(U) = 5000
V(R, a) = 100		
V(R, b) = 20	V(S, b) = 50	V(U, b) = 200
V(R, c) = 200	V(S, c) = 100	
	V(S, d) = 400	
		V(U, e) = 500

Moreover, suppose that all attributes are of the same size, and that each page can contain 10 tuples of R. Records do not span across pages. Unless otherwise stated, all attributes should be included in a join output, for example, the join output of R and S (without any projection) should contain 6 attributes (a, b, c, b, c, d). Unless otherwise stated, we also made the standard assumptions that we have introduced in our lecture, e.g., preservation of values, uniform distribution of data, and so on.

What is the output size (**in number of tuples**) of the following query:

SELECT DISTINCT b FROM R;

3. Consider the same setting as Question 2. Now, suppose we have the following information on the distribution of the distinct values of b in R and S:



What is the output size (**in number of tuples**) of the following query:

SELECT * FROM R, S WHERE R.b = S.b and R.c = S.c;

4. Consider the same setting in Question 3 (**WITH the histogram distribution**). What is the size of the following query (**in number of tuples**)? Assume that U.e are integer values ranging from 1 to 500.

   SELECT R.a, S.d FROM R, S, U WHERE R.b = S.b and R.c = S.c

   and R.b = U.b and U.e > 250;

5. Consider the same setting in Question 2 (i.e,. **WITHOUT the histogram information**). Consider the query

   SELECT R.a, S.d FROM R, S, U WHERE R.b = S.b and R.c = S.c

   and R.b = U.b and U.e > 250;

   Suppose our DBMS employs the Dynamic Programming (System R) optimizer. As covered in the lecture, the optimizer avoids cross products, restricts the space to left-deep trees, and keeps only one optimal plan for a combination of tables. Suppose our DBMS supports only one join algorithm: the **page nested-loops join** (with 3 buffer available for the join). All intermediate results for joins are to be written out to disk. Suppose there are no indexes. For simplicity, you can assume U to be the intermediate table after the selection condition U.e has been applied; the cost to generate this intermediate table is ignored. Assume that U.e are integer values ranging from 1 to 500.

   Which of the following subplans/plans are pruned? Hint: You should only consider subplans/plans that are **explicitly** generated and considered by the algorithm, i.e., you should not consider a plan that is not generated at all, e.g., those involving cross products.

   A. RS   B. SR   C. RU
   D. (UR)S   E. (RS)U   F. (SR)U

6. Referring to the setting in Question 5. What is the final optimized plan? Write "(SU)R" if you think the final plan is the join of S and U first with S being the outer table, followed by joining the intermediate result with table R with R being the inner table.

7. Which of the following statement(s) is(are) TRUE?
   A. Randomized algorithms operate on full plans (states). The first state has to be randomly generated.
   B. Greedy algorithms are typically polynomial time complexity because they do not consider plans in the bushy tree space.
   C. The Dynamic Programming algorithm is typically not used for large number of joins (e.g. more than 100) because error propagation may result in sub-optimal plans.
   D. Greedy algorithms typically do not produce optimal solution because they generate the query plan bottom-up (starting from 2 tables, then 3 tables, etc).
   E. A & B   F. B & C   G. A & D   H. None of the above.

8. The cost of a query plan is error prone. Which of the following is not a valid reason?
   A. Cost model does not capture the actual cost (CPU or I/O) accurately.
   B. Error in intermediate result size estimation.
   C. Inaccurate base table size statistics.
   D. Assumptions like preservation of values are used.
   E. None of the above.

9. Which of the following statement(s) is (are) FALSE?
   A. Strict 2PL schedules will not have dirty reads.
   B. Strict 2PL schedules will not result in a deadlock.
   C. Strict 2PL schedules is not always view serializable.
   D. Strict 2PL schedules will not result in unrepeatable reads.
   E. B & C F. B & D G. C & D H. A & D

10. Which of the following statement(s) is/are TRUE?
    A. The validation-based protocol can never have dirty reads.
    B. The validation-based protocol can never have deadlocks.
    C. The validation-based protocol can never have unrepeatable reads.
    D. The validation-based protocol can never have cascading aborts.
    E. The validation-based protocol can never have starvation.
    F. A, B, C G. A, B, C, D H. A, B, D, E I. None of the above.

11. Consider the following two transactions:
    a. T_1: $r_1(X)$, $r_1(Y)$, $w_1(Y)$, $w_1(X)$
    b. T_2: $r_2(Y)$, $w_2(Y)$, $r_2(X)$, $w_2(X)$

    How many conflict-serializable schedules can be created by interleaving these two transactions (including serial schedules)?

12. Which of the following statement(s) is TRUE with regards to this schedule:

    S: $r_1(X)$ $w_2(Y)$ $w_2(Y)$ $w_1(Z)$ $r_2(X)$ c_1 $w_2(Z)$ c_2

    A. S is recoverable.
    B. S is cascadeless.
    C. S is strict.
    D. S is view serializable.
    E. A & B F. A, B & C G. A, B & D
    H. B & D I. B, C & D J. A, B, C & D

13. Consider a DBMS that employs a new lock model for the INCREMENT operation. The INCREMENT operation is "special" because it is commutative, e.g., transactions T_1 and T_2 both increment element A; and the final result of A is the same regardless of which transaction operates on A first. The compatibility matrix now becomes the following:

	S	**X**	**I**
S	T	F	F
X	F	F	F
I	F	F	I

Here, S, X and I denote shared (read), exclusive (write) and increment locks respectively. So, if a transaction holds an I lock, then no other transactions can hold an S or X lock. However, many transactions can hold an I lock concurrently. We are given the following schedule (incr represents an increment action):

$$S = r_1(A); r_2(B); r_3(C); incr_2(C); r_3(B); incr_1(C); incr_2(A);$$

Which of the following statements is TRUE?

A. S is serializable and is equivalent to the serial schedule T1, T2, T3
B. S is serializable and is equivalent to the serial schedule T2, T3, T1
C. S is serializable and is equivalent to the serial schedule T3, T1, T2
D. S is serializable and is equivalent to the serial schedule T1, T3, T2
E. S is serializable and is equivalent to the serial schedule T2, T1, T3
F. S is not serializable

14. Which of the following statement(s) is/are FALSE?

A. For a DBMS that correctly implements the failure recovery mechanisms akin to those discussed in class, the data stored in the relation files on the disk is always in a consistent state.
B. For a DBMS that correctly implements the concurrency control mechanisms akin to those discussed in class, the data stored in the relation files on the disk is always in a consistent state.
C. For a DBMS that correctly implements failure recovery and concurrency control mechanisms akin to those discussed in class, the data stored in the relation files on the disk is always in a consistent state.
D. A & C
E. B & C
F. A & B
G. All of the above.
H. None of the above.

15. Four transactions, T_1, T_2, T_3, T_4. running in a DBMS would have resulted in the following schedule for their actions (based on their time of arrivals if we had ignored locking):

    $r_1(X)$; $w_2(Y)$; $r_1(Y)$; $r_3(Z)$; $w_3(W)$; $w_4(Z)$; $w_3(X)$; $w_2(W)$

    However, for each action, the transaction has to acquire the necessary locks for its action to proceed. In addition, we assume that a transaction never releases a granted lock unless it is aborted, in which case, its locks may be granted to the transaction waiting in the queue. Aborted transactions are not restarted. Suppose the lock manager adopts the Wait-Die policy, and the priority of the transactions are: $T_1 > T_2 > T_3 > T_4$. In other words, T_1 is the oldest transaction while T_4 is the youngest. At the end of the sequence of lock requests, how many transactions are waiting, and how many transactions are aborted?

    A. #waiting = 0; #aborted = 1
    B. #waiting = 1; #aborted = 1
    C. #waiting = 2; #aborted = 1
    D. #waiting = 0; #aborted = 2
    E. #waiting = 1; #aborted = 2
    F. #waiting = 2; #aborted = 2

16. Consider the same setting in Question 15, except the lock manager now adopts the Wound-Wait policy. At the end of the sequence of lock requests, how many transactions are waiting, and how many transactions are aborted?

    A. #waiting = 0; #aborted = 1
    B. #waiting = 1; #aborted = 1
    C. #waiting = 2; #aborted = 1
    D. #waiting = 0; #aborted = 2
    E. #waiting = 1; #aborted = 2
    F. #waiting = 2; #aborted = 2

17. In practice, the NoUndo/NoRedo recovery scheme is not widely used. Which of the following statements is TRUE?

    A. Redo logs are still maintained on disk. However, as soon as a transaction commits, the log records are removed.
    B. Undo logs are still maintained on disks. However, the updated data are not written to the disks until commit time. As such, if there is a crash before a transaction commits, there is no need to perform undo since the data are not updated until commit time.
    C. Redo and Undo logs are still maintained but they are kept in the memory only, which consumes too much memory resource.
    D. No logs need to be maintained. But updates have to be written out to disk to ensure all dirty pages are on disk. This incurs random I/O. But, no recovery is needed since dirty pages are already on disk.
    E. None of the above.

18. Consider a DBMS that uses the Validation-based optimistic concurrency control mechanism (as taught in class). The following table lists two transactions with their read and write sets:

Transaction	Read Set	Write Set
T_1	{X, Y}	{X, W}
T_2	{Z}	{Y, Z}

Based on the Validation-based protocol, the 3 phases – Start-Phase (S), Validate Phase (V), Finish Phase (F) – of the transactions can be interleaved in different ways. For example, one possible interleaving (which we shall refer to as a schedule) is T_1.S, T_2.S, T_1.V, T_2.V, T_1.F, T_2.F. **Suppose T_1 starts first**. Which of the following statements is TRUE:

A. There is exactly one schedule where T_1 validates successfully, but T_2's validation fails.
B. There is exactly one schedule where T_2 validates successfully, but T_1's validation fails.
C. There is exactly one schedule where both transactions validate successfully.
D. There is exactly one schedule where both transactions validate unsuccessfully.

19. Consider the same setting in Question 17. Suppose we are allowed to change the ReadSet of T_1. What is the **minimum** ReadSet(T_1) for both transactions to validate successfully? If ReadSet(T_1) = {X}, then write "X"; if there are multiple possible answers, you only need to write any one of them (e.g., if both X and Y are correct answers, then you only need to write either "X" or "Y").

20. Which of the following statements is TRUE of non-quiescent checkpointing?

A. In UNDO logging, all the log records before the most recent <START CKPT> log can be discarded.
B. In REDO logging, all the log records before the most recent pair of <START CKPT> and <END CKPT> log can be discarded.
C. In UNDO/REDO logging, even if checkpointing is used, it is still possible the recovery process accesses logs at the beginning of the log file.
D. If a crash happens during checkpointing (i.e., there is a <START CKPT> log but no <END CKPT> log), we only need to examine logs upto the <START CKPT> log of the most recent successful checkpoint.
E. None of the above.

21. Consider the concurrency control mechanism for B^+-tree (as taught in class). Suppose we have a 3 level tree – root, internal node, leaves. Which of the following sequence(s) is(are) possible sequences of actions on the tree to insert a new entry to a leaf page? Here, L denote Lock, U denote Unlock, M denote modification. L(X) denote locking a node at level X (X = R for root; I for internal node; and L for leaf node). U(X) is similarly defined.
    A. L(R); L(I); L(L); M(L); U(R); U(I); U(L)
    B. L(R); L(I); U(R); L(L); U(I); M(L); U(L)
    C. L(R); L(I); L(L); U(R); U(I); M(L); U(L)
    D. L(R); L(I); U(R); L(L); M(L); U(I); U(L)
    E. A & B
    F. A, B & C   G. A, B, C & D   H. B & C   I. B, C & D.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ **END OF PAPER** ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~