HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY

COMP3311: Database Management Systems

Spring Semester, 2019

Final Examination

May 20th, 2018

4:30pm to 7:30pm

Name: ____________________ **Student Number: ______________________**

Email: ____________________

Instructions:

1. This is an open-note examination. You are allowed to bring and use the printed course slides from lectures, tutorials and labs in the examination. No electronic device but simple calculator is allowed throughout the exam.
2. This examination paper consists of **19** pages and **6** questions.
3. Please write your name, student ID and Email on this page.
4. For each subsequent page, please write your student ID at the top of the page in the space provided.
5. Please answer **all** the questions within the space provided on the examination paper. You may use the back of the pages for your rough work and afterwards drawn a diagonal line through it to show that it is not part of your answer.
6. Please read each question very carefully and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
7. Leave all pages stapled together.
8. The examination period will last for **3 hours**.
9. Stop writing immediately when the time is up.

Questions	**Marks**	**Scores**
1	**30**	
2	**16**	
3	**14**	
4	**12**	
5	**14**	
6	**14**	
Total	**100**	

Q1 Multiple-choice Questions (30 marks)

For each question, there is only one correct answer. Please put your answers in the following boxes. Only answers in the boxes will be considered and graded.

1	**A**	**6**	**C**	**11**	**D**	**16**	**D**
2	**B**	**7**	**B**	**12**	**B**	**17**	**A**
3	**C**	**8**	**C**	**13**	**C**	**18**	**D**
4	**A**	**9**	**D**	**14**	**B**	**19**	**C**
5	**B**	**10**	**A**	**15**	**D**	**20**	**D**

1. Which of the following operation is equivalent to the natural join of two relations r and s when the relational schemas of r and s have no common attributes?

   A. Cartesian product.
   B. Set difference.
   C. Set union.
   D. Set intersection.

2. Given a database having two tables R(A,B) and S(B,C) where the primary key is underlined with usual notation. Which of the following statements is *true* when running the following SQL statement successfully?

   ```
   CREATE VIEW T AS
   SELECT A, C
   FROM R, S
   WHERE R.B = S.B;
   ```

   A. A new table T(A,C) is created and then stored in the database.
   B. We can always use T with R or S in the FROM clause of an SQL statement.
   C. We can always insert a record in T.
   D. We can always modify the attribute values A or C in T.

3. Which is the correct explanation of *physical data independence* in the three levels of abstractions of a DBMS?

   A. Independence between the view level and the logical level.
   B. Independence between the view level and the physical level.
   C. Independence between the logical level and the physical level.
   D. Independence between the physical level and the network level.

4. Consider the following **Orders** table.

CID	BID	QTY
1	1	6
1	2	10
2	1	20
2	2	30
3	4	60

What is the result of running the following SQL statement?

```
SELECT CID, AVG(QTY)
FROM ORDERS
WHERE QTY ≤ 10
GROUP BY CID
HAVING COUNT(*) > 1;
```

A. One record is displayed, and it is (1, 8).
B. One record is displayed, and it is (2, 25).
C. Two records are displayed, and they are (1, 8) and (2, 25).
D. Two records are displayed, and they are (2, 25) and (3, 60).

5. Which is the best description of NOSQL databases?

A. A NOSQL database does not need to support selection on data.
B. A NOSQL database does not need to use relational tables to store data.
C. A NOSQL database does not need to obey key constraint.
D. A NOSQL database does not need to handle requests from multiple sites.

6. Which of the following statements about column-oriented database (COD) when comparing with relational database is true?

A. A COD is more efficient when we want to project on multiple columns of a table.
B. A COD is more efficient when we want to insert a new record in a table.
C. A COD is more efficient when we want to compute aggregates on a column.
D. A COD is more efficient when we want to join tables.

7. Which of the following statements about database API is true?

A. An embedded database API cannot use early binding.
B. An embedded database API cannot use late binding.
C. A call-level database API cannot use early binding.
D. A call-level database API cannot use late binding.

8. Consider the following statements about database APIs:

   I. The application code related to ODBC in an application needs to be modified whenever the database driver is changed.
   II. Different JDBC driver types provide tradeoffs in terms of portability and performance.
   III. SQLJ is able to perform SQL syntax checking in a source program before the runtime.

   A. Only I and II are true.
   B. Only I and III are true.
   C. Only II and III are true.
   D. All are true.

9. Which of the following statements about indexing creation and design are true?

   I. We could define only one primary index for each table.
   II. We could define more than one secondary index for each table.
   III. Clustered indexes need not have unique search key.

   A. II only.
   B. III only.
   C. II and III only.
   D. All of them.

10. Suppose we are computing $R \bowtie S$ with two large tables (e.g. R and S have more than 1 million rows). The estimated cost of a block nested join is approximated cut in half, if :

    A. Tuples per block of R is doubled but the number of tuples of R is unchanged.
    B. Number of tuples of R is halved but the number of blocks of R is unchanged.
    C. Number of tuples of S is halved but the number of block of S is unchanged.
    D. Tuples per block of R is halved but the number of tuples of R unchanged.

11. Which of the following statements about using index search in a query is *false*?

    A. It is efficient to use hashing if the query involves only a single search key value.
    B. Multicolumn indexes are not always superior to multiple indexes.
    C. If the query contains many predicates, it is better to build a multicolumn index over the attributes of all those predicates in the query.
    D. The less retrieved rows in the result are, the more efficient the query processes.

12. Consider the following schedule for transactions T1, T2 and T3 with usual notation:

    S: <R1(X), R2(Y), R3(Y), W2(Y), W1(X), W3(X), R2(X), W2(X)>

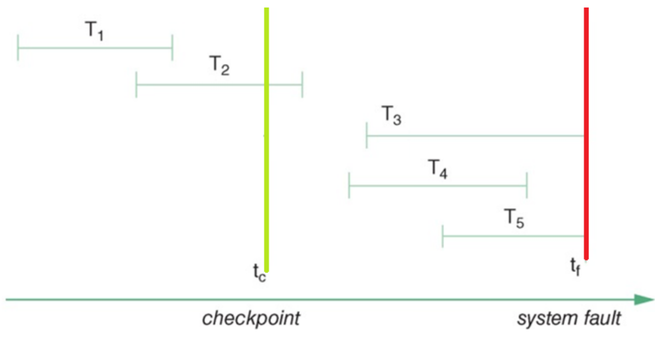
    Which one of the following is correct?

    A. S is serializable and an equivalent serial schedule is T1, T2, T3.
    B. S is serializable and an equivalent serial schedule is T1, T3, T2.
    C. S is serializable and an equivalent serial schedule is T3, T2, T1.
    D. S is not serializable.

13. Which one of the following statements is false?

    A. A cascadeless schedule is always recoverable.
    B. The commit order of a serializable schedule can always be set to make the schedule become recoverable.
    C. If a schedule is executed by a locking protocol, it is always serializable.
    D. Even if a serializable schedule is executed by a 2PL protocol, it may run into a deadlock.

14. The following figure presents five transactions (T1 to T5) in usual notation with checkpoint and system fault along the timeline as shown. Assuming we use deferred database modification pending all the updates until the checkpoint. Which one of the following statements is true in the recovery?



    A. We should do nothing for T1 and T5 only.
    B. We should do nothing for T1, T3 and T5 only.
    C. We should do nothing for T3 and T5 only.
    D. We should do nothing for all transactions.

15. Consider the following schedule for transactions T1, T2 and T3 with usual notation:

    S: <R1(X), W2(X), R3(X), W2(Y), W1(Y), R3(Y), R1(X), W2(X), c2, c3, c1>

    Which one of the following is correct?

    A. S is serializable
    B. S is recoverable
    C. S is cascadeless
    D. None of the above.

16. Which of the following sequence of operations in a mapReduce pipeline should be performed simultaneously?

    A. Map and Reduce
    B. Map and Sort
    C. Shuffle and Reduce
    D. Sort and Shuffle

17. Which of the following statements is false?

    A. RAID level 0 does not help improve the read performance.
    B. RAID level 0 is used if performance is more important than fault tolerance.
    C. RAID level 1 needs twice the amount of storage capacity compared to RAID 0 to store the same amount of data.
    D. RAID 5 strikes a balance between read/write performance, storage efficiency and fault tolerance.

18. Which one of the following regarding using B+ trees as index is correct?

    A. A taller B+ trees has better performance gain.
    B. The physical location of records is affected by updating a key in a B+ tree.
    C. If there are a total of n records in the database and the tree has fanout B, the height of the B+-tree is at most $\lceil \log(B)*n \rceil$.
    D. If the B+-tree has fanout B, all the nodes of the B+-tree always have at least $\lceil (B-1)/2 \rceil$ keys.

19. Consider the bitmap example shown in the following figure.

record number	*name*	*gender*	*address*	*income -level*
0	John	m	Perryridge	L1
1	Diana	f	Brooklyn	L2
2	Mary	f	Jonestown	L1
3	Peter	m	Brooklyn	L4
4	Kathy	f	Perryridge	L3

Bitmaps for *gender*

m	10010
f	01101

Bitmaps for *income-level*

L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

Which one is the correct bitmap operations for finding the records satisfying the conditions "gender = male" and "income level lower than L3"? The income levels are set to be L5 > L4 >… > L1?

I. (10100 OR 01000) AND 10010
II. (10100 AND 01000) AND 10010
III. (10100 OR 01000) AND NOT(01101)

A. I Only.
B. II and III Only.
C. I and III Only.
D. All of them.

20. Which one of the following statements is *false*?

I. B+-trees can be used as a file organization method.
II. B+-trees can be used as a primary index or a secondary index.
III. B+-trees have data pointers in the leaf level only.

A. I only.
B. II only.
C. III only.
D. None of the above.

Q2 Physical File Organization and Indexing (16 marks)

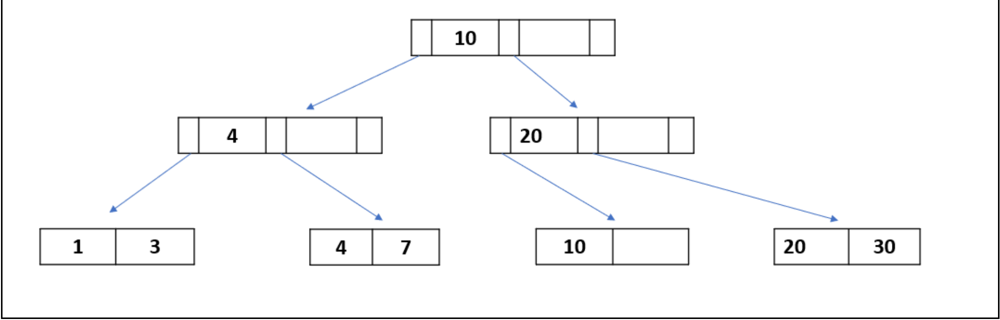
(a) (8 **marks**) You are given the following B$^+$-tree with fanout n=3. The tree is based on commonly used simplified notation.

i. Draw clearly the B$^+$-tree for each of the following insertion operations: first insert 15, then insert 2. (i.e. two updated B$^+$-trees in total)

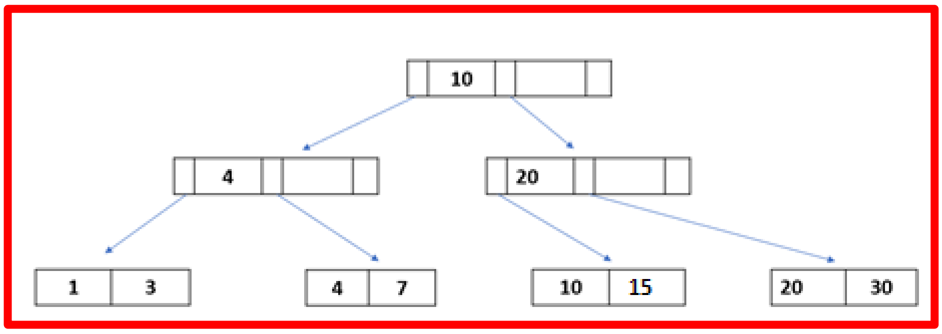
ii. Draw clearly the B$^+$-trees for each of the following deletion operations: first delete 3, then delete 20, finally delete 10. (i.e. three updated B$^+$-trees in total)

The insertions and deletions are performed in the described orders. Both parts (i and ii) should begin with the initial B$^+$-tree given below.

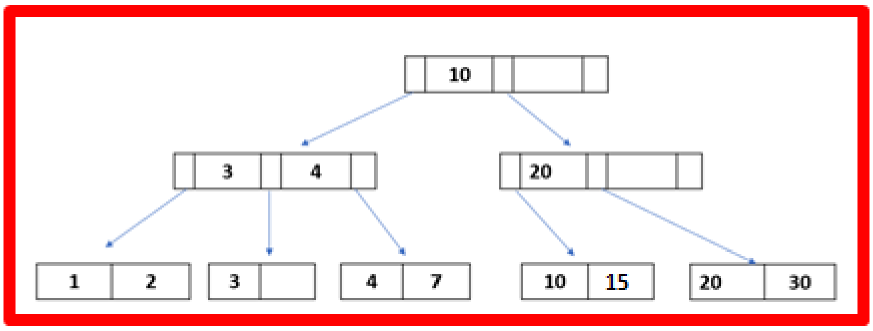
Initial B$^+$-tree



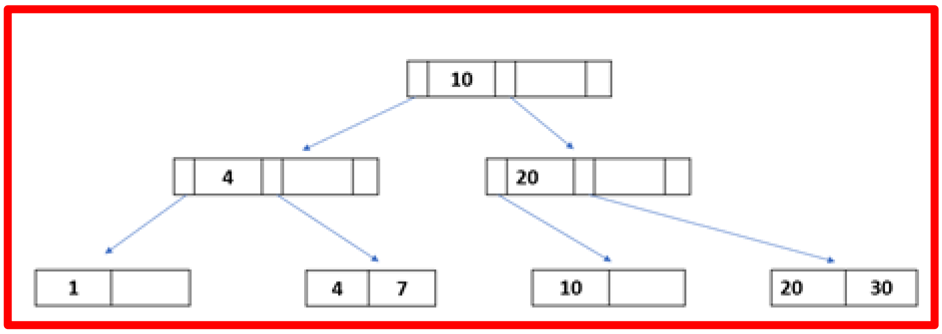
i. (**1 marks**) First insert 15:



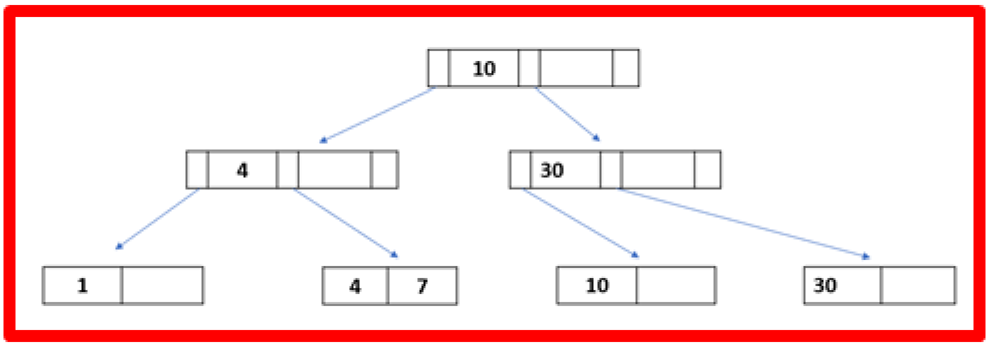
(**2 marks**) Then insert 2:



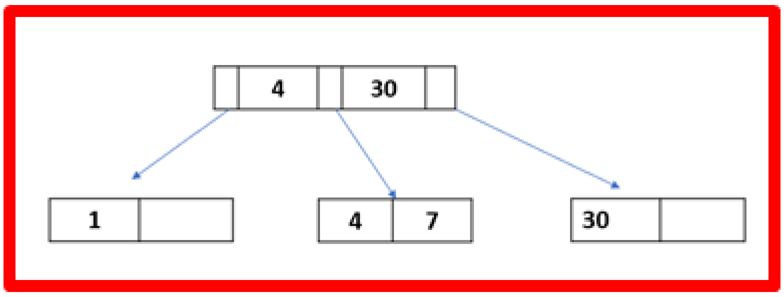
ii. (**1 marks**) Start from the initial B^+-tree again, first delete 3:



(**2 marks**) Then delete 20:



(**2 marks**) Finally delete 10:



(b) (**8 marks**) Consider a movie database with the following relations and sizes of each attribute:

- Film (<u>title</u>: 32 bytes, director: 32 bytes, release_year: 4 bytes, company: 32 bytes, type: 8 bytes)
- Actor (<u>id</u>: 4 bytes, name: 32 bytes, date_of_birth: 8 bytes)

There exist 10,000 films in the database and 200,000 actors. Each page is 512 bytes and each pointer is 8 bytes. Show your main computation and explanation clearly in the following parts.

i. (**2 marks**) Assume that the relation Film is sorted on title and there is no index, what is the cost (in terms of page reads) for finding the film with title "Avengers: Endgame" ?

(no penalty for sba or rba for all parts):
$\lfloor$512 bytes per page / 108 bytes per Film record$\rfloor$ = 4
$\lceil$(10,000 Film records / 4 Film records per page)$\rceil$ = 2500 pages
<u>$\lceil \log_2 2500 \rceil$ = 12 pages (via binary search)</u>

ii. (**2 marks**) Assume that the Actor relation is sorted on the name and we want to create an ordered index on id where each index entry has the form <id, *pointer*> and it is a single-level index. What is the cost of retrieval based on a single id using this index (e.g., "Find actor with id=50")?

bf_{Aindex} = $\lfloor$512 bytes per page / (4 + 8) bytes per index entry$\rfloor$ = 42
page need =$\lceil$200,000 Actor records / 42 index entries per page$\rceil$ = 4762
<u>$\lceil \log_2 4762 \rceil$ + 1 = 14 pages</u>

iii. (**2 marks**) Continue from Part ii, if we convert the single-level index into a multi-level index, how many levels of index are needed (assuming the index taking full pages)? Show your computation and explanation clearly.

Total levels needed: 4
We have 4762 index pages to store 200,000 Actor index entries. This is the last level of index.
4762 index entries need $\lceil$4762 / 42 index entries per page$\rceil$ = 114 index pages. This is the second last level of index.
114 index entries need $\lceil$114 / 42 index entries per page $\rceil$ = 3 index pages. This is the third last level of index.
Finally, 3 index entries need one additional index page. There are 4 levels of index.

iv. (**2 marks**) Using the multi-level index from Part iii, what is the cost of answering the query "Find the actor with id = 50"?

4 level of index search + 1 page of data (only 1 record) = 5 pages

Q3 Join Operation (14 marks)
Consider the following two relational tables with usual notation:

- Employee(<u>e_id</u>, e_name, e_address)
- Department(<u>d_id</u>, d_name, d_address, *e_id*), where e_id is the id of manager for the department (i.e. a foreign key to the employee file)

There are 100,000 employees and 1,000 departments. The size of each page (memory or disk) is 1000 bytes. Also the attribute sizes of e_id, e_name, d_id, d_name are 10 bytes each, while the attribute sizes of e_address and d_address are 20 bytes each. Assume that we only have 10 memory pages and that we run the following query as follows. Estimate the cost of the using two different join methods in Parts (a) and (b).

SELECT E.e_name, D.d_id, D.d_name
FROM Employee E, Department D
WHERE E.e_id = D.e_id

(a) **(6 marks)** Using Nested Loop Join:

i. We need ________4000 pages to store Employee and _____50 pages to store Department.

ii. The minimal memory space needed to run this join is ____________3 pages.

iii. If Department is used as the outer relation, the cost for using block-nested loop join is 28050__ pages. If Employee is used as the outer relation, the cost is______29000 pages.

iv. The minimal memory pages to achieve the most efficient block-nested join is ________52.

(b) **(8 marks)** Using Hash Join:

i. The maximum number of buckets we can use for partitioning is _____9 . Suppose we use a hash function to generate the maximum number of buckets, __Department (fill in 'Employee' or 'Department') must be used as build input to fit in the memory.

ii. The overall cost of hash join is ________________12150 pages.

iii. To further reduce the cost of hash join, we first partition the build input and write back to the disk. Before writing back to the disk and we first remove unnecessary attribute(s) ________d_address, e_address. (Leave blank if all attribute are necessary). The cost of completing the hashing step for both relations is ________6081 pages. Finally, we read each bucket of build input and match it against the probe input, the cost of this step is 2031___pages. So the overall cost of using hash join in this process is _______8112 pages.

Q4 NOSQL (12 marks)

We use a mongoDB database to generate a key-value store “wdb” for recording the quantity of products per type of wine. Some data samples are given as follows:

k1	**v1**
1	{WineType: Merlot, Quantity in bottle: 1000}
2	{WineType: Sparkling, Quantity in bottle: 4000}
3	{WineType: Barbera, Quantity in bottle: 3000}
...	...

The following code is to execute a MapReduce query with MongoDB wdb and the query is given as follows:

Query: Find the average quantity of products per wine type.

```
// We implement in our Java code by passing plain strings “map” and “reduce” to the mapReduce()
// to compute the average quantity of products per wine type

public static void reportAggregate(MongoDatabase wdb) {
        String map = "function() { " +
                        " var Quantity = this.Quantity; " +
                        " this.WineType.forEach(function(WineType) { " +
                        " emit(WineType, {average: Quantity, count: 1}); " +
                        " }); " +
                        "} ";
        String reduce = "function(WineType, Values) { " +
                        " var s = 0; var newc = 0; " +
                        " Values.forEach(function(curAvg) { " +
                        " s += curAvg.average * curAvg.count; " +
                        " newc += curAvg.count; " +
                        " }); " +
                        " return {average: (s / newc), count: newc}; " +
                        "} ";

MapReduceIterable<Document> result = wdb.getCollection("wines").mapReduce(map, reduce);
        for (Document r : result)
                System.out.println(r);}
```

(a) **(6 marks)** After running the program and invoke the mapReduce function, we represent the key-value pairs in different workers (machines) after the mapping process as follows:

Worker 1	
k2	**v2**
Merlot	1000, 1
Merlot	500, 1
Sparkling	4000, 1
Barbera	3000, 1
Barbera	1000, 1

Worker 2	
k2	**v2**
Sparkling	1000, 1
Merlot	1500, 1
Malbec	3000, 1
Malbec	3000, 1
Syrah	1000, 1

Worker 3	
k2	**v2**
Merlot	1000, 1
Sparkling	1500, 1
Sparkling	5500, 1
Syrah	3000, 1
Syrah	2000, 1

i. Suppose we first reduce the Worker 1 and Worker 2. Write down the result in the following table and assume that the sort and shuffle process is ignored (note: the number of rows may be more or less than necessary and you could add more rows or leave some rows blank).

k2	**v3**
Merlot	1000,3
Sparkling	2500,2
Barbera	2000,2
Malbec	3000,2
Syrah	1000,1

ii. Finally, we reduce the above result with Worker 3. Write down the final result in the following table.

k2	**v3'**
Merlot	1000,4
Sparkling	3000,4
Barbera	2000,2
Malbec	3000,2
Syrah	2000,3

(b) **(6 marks)** Fill in the two blank sections of the following code to execute the following query on wdb:

Query: Find the maximum quantity of products per wine type. You may assume the function Math.max(x, y) returns a maximum between x and y in JS code.

Note: Your variables should be consistent with all the parameters in the code.

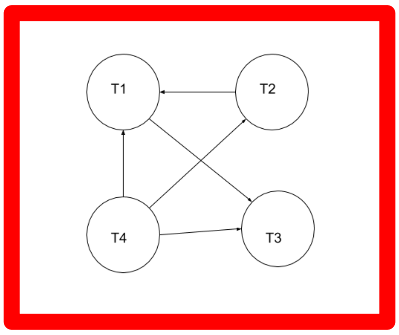
```
public static void reportAggregate(MongoDatabase wdb) {
        String map = "function() { " +
                        " var Quantity = this.Quantity; " +

                              " this.WineType.forEach(function(WineType) { " +
                        " emit(WineType, {max: Quantity}); " +

                             " }); " +
                        "} ";
        String reduce = "function(WineType, Values) { " +
                        " var newMax = 0; " +

                              " Values.forEach(function(curMax) { " +
                        " newMax = Math.max(newMax, curMax.max); " +

                              " }); " +
                        " return {max: newMax}; " +
                        "} ";

MapReduceIterable<Document> result = wdb.getCollection("wines").mapReduce(map, reduce);

        for (Document r : result)
                System.out.println(r);}
```

Q5 Transaction Management and Recovery Management (14 marks)

(a) **(8 marks)** Consider the following schedule S with four transactions (T1 to T4) with usual read and write operations (R and W) on items (X, Y, Z) and commit operations (C1 to C4) presented in time sequence as follows:

S: <R4(X),W1(X),W4(Z),R2(Z),R1(X),W3(X),W1(Z),R4(Y),W2(Y),W1(Y),C1,C2,C3,C4>

i. Draw the precedence graph for the schedule S. Explain why or why not the schedule serializable?



$$\begin{tikzcd}
T1 \arrow[dr] & T2 \arrow[l] \\
T4 \arrow[u] \arrow[ur] \arrow[r] & T3
\end{tikzcd}$$

Yes, no cycle in the PG.

ii. Consider the case when two-phase locking is used on Part i.

- Use the notation lock_s and lock_x with transaction id to show shared and exclusive locks. For example, lock_s1(A) and lock_x1(B) show that Transaction 1 acquires shared lock on item A and exclusive lock on item B.
- Use the notation unlock with transaction id to show releasing locks. For example, unlock1(A) to show Transaction 1 releasing its lock on item A.
- Insert all the lock and unlock. Write a new schedule S' in a sequence similar to S given above. You may use more than one line to show S'.

S':

<lock_s4(X),R4(X),lock_x1(X),W1(X),lock_x4(Z),W4(Z),lock_s2(Z),R2(Z),R1(X),
lock_x3(X),W3(X),unlock3(X),lock_x1(Z),W1(Z),lock_s4(Y),R4(Y), unlock4(X),
unlock4(Y),unlock4(Z),lock_x2(Y),W2(Y),unlock_2(Y),unlock_2(Z),lock_x1(Y),
W1(Y),unlock1(X),unlock1(Y),unlock1(Z), C1, C2,C3,C4>

iii. Will two-phase locking allow the schedule to be executed written above?

___________NO

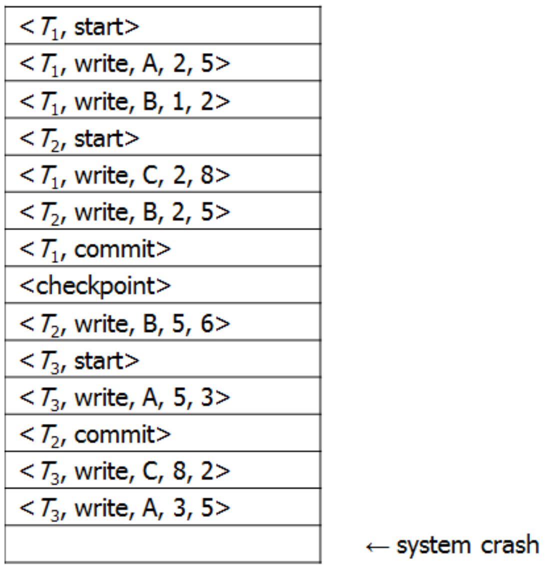
iv. Will there be deadlocks during the execution of the schedule written above?

__________NO

v. Is the schedule given in S recoverable? If not, write down a (one) new sequence of the commit times C1 to C4 such that the schedule becomes recoverable.

NO (not recoverable)
Schedule: Move C2 after C4 or C4 before C2.

(b) **(6 marks)** Consider the log corresponding to a schedule of three concurrently executing transactions T1, T2, and T3 where a system crash occurs at the point indicated in the given table. We use the log for running the database recovery. Assume that we use the immediate update protocol with checkpoint.



< T_1, start>
< T_1, write, A, 2, 5>
< T_1, write, B, 1, 2>
< T_2, start>
< T_1, write, C, 2, 8>
< T_2, write, B, 2, 5>
< T_1, commit>
<checkpoint>
< T_2, write, B, 5, 6>
< T_3, start>
< T_3, write, A, 5, 3>
< T_2, commit>
< T_3, write, C, 8, 2>
< T_3, write, A, 3, 5>

← system crash

i. Write down the item values just after checkpointing: A:__5___B:__5___C:__8___

ii. Which transactions should be undone in recovery? ___________________T3

iii. Which transactions should be redone in recovery? ___________________ T2

iv. Write down the item values after the recovery operations: A:_5__B:_6__C:_8__

Q6 SQL and PL/SQL (14 marks)

Consider the following relational schemas.

- Player (playerID, name, skill_level, international_ranking, debut_date, salary)
- Club(clubID, clubLevel, coach)
- Membership(*playerID*, *clubID*)

Notation: Primary keys are underlined and foreign keys are in italics. Membership has foreign keys from Club and Player tables. The datatypes are given in the following table:

Primary Key Attribute	number(10)	
name, coach, clubLevel	varchar(100).	clubLevel can only have three domain values: National, State, or City
skill_level, international_ranking, salary	number(10)	
debut_date	date	

(a) **(3 marks)** Write an SQL to show the club ID and number of players for the club that has the most number of players in it. You should name your columns exactly as "Club ID" and "Number of Players".

```
select Membership.clubID as "Club ID", count(Membership.playerID) as "Number of
Players"
from Membership
group by Membership.clubID
having count(Membership.playerID) = (select max(count(membership.playerID))
      from Membership
      group by Membership.clubID)
```

(b) **(4 marks)** Write an SQL query that gives the unique names of players who have played at "National" Level clubs and have also played in every club that is coached by "Bill"

```
select distinct p.name
from Player p, Club c, Membership m
where p.playerID = m. playerID and c.clubID = m.clubID and c.clubLevel = 'National' and
NOT EXISTS
(
select clubID from Club where coach = 'Bill'
MINUS
select m2. clubID from Membership m2 where m2.playerID = p.playerID),
```

(c) **(3 marks)** Suppose we have 4 records in the table **Player** as follows (irrelevant attributes are omitted):

playerID	international_ranking
1	1500
2	800
3	2000
4	1000

What will be the *international_ranking* of each record in the table **Player** after executing the PL/SQL script below? Your answer should be shown by filling the given table below the script (inserting or deleting rows if necessary).

```
DECLARE
            increase NUMBER(4):=10,
    BEGIN
        FOR p in (
                    SELECT playerID, international_ranking FROM Player)
            ) loop
                    UPDATE Player SET international_ranking=international_ranking-
                    increase
                    WHERE playerID=p.playerID,
                    increase:=increase*2,
        END LOOP
    END,
    /
```

playerID	international_ranking
1	1490
2	780
3	1960
4	920

(d) **(4 marks)** Complete the blanks in the following PL/SQL script, which finds the current number of clubs each player has played in, and adds the number into table **Player** as new attribute *no_of_clubs*, which we suppose it to be a two digit number.

```
/*Finish the statement below, add a new attribute to Player table*/

ALTER TABLE Player ADD (no_of_clubs NUMBER(2)),

DECLARE
   var_clubs NUMBER(2),
   var_playerID NUMBER(10),
   CURSOR club_cursor
/*Finish the statement below, find the playerID and number of clubs each player has
played in*/

   IS SELECT playerID, count(*) AS clubs FROM Membership GROUPBY playerID,

BEGIN
   FOR rec in club_cursor
/*Continue the statements below*/

   LOOP
      var_clubs:=rec.clubs,
      var_playerID:=rec.playerID,
      UPDATE Player SET no_of_clubs = var_clubs WHERE playerID=var_playerID,
   END LOOP,

END,
/
```

- End of the Exam Paper-