SWE3003 Database Systems - Final Exam Fall 2022

Student ID	Name

For Instructor/TA only,

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total [100 pts]

Academic Honor Pledge

I affirm that I will not at any time be involved with cheating or plagiarism while enrolled as a student Programming Language class at Sungkyunkwan University. I understand that violation of this code will result in penalties as severe as indefinite suspension from the university.

Your signature: ______________________

1. [Storage and File Structures] You have a database table that consists of 10,000 records. Each 4 KB disk page holds 100 records without free space. I.e., at least 100 disk pages are required for the database table. Suppose your DBMS does not use the buffer cache. For each of the following database table format, describe how many disk pages need to be accessed on average for each insert or point (exact match) query.

   (a) Heap file - insert

   1 page answer:

   (b) Heap file - search

   average, 50 pages answer:

   (c) Sequential file - insert

   $log_2 100 + 50$ pages answer:

   (d) Sequential file - search

   $log_2 100$ pages answer:

2. [External Sort-Merge] Suppose you are asked to sort a file of 20,000 pages on a computer with approximately 101 buffer pages available (100 pages to buffer input runs and 1 page to buffer output).

   (a) How many merge passes are required for external merge sort?

   answer:

   initial number of sorted runs: 200
   $(log_{100} 200) = 2$

   (b) How many disk pages are expected to be **written** by the external merge sort?

   answer:

   20000 x 2 = 40000

3. [Block Nested-Loop Join] Suppose there is a transaction the joins two relations (tables) using 'Block-Nested Loop Join' algorithm with 102 buffer pages. 100 pages are used to buffer the outer relation, one page is reserved for the inner relation, and the other page is reserved for the output. Both relations are not sorted, and the inner relation has 200 pages and the outer relation has 500 pages.

   (a) How many pages will be **read** (transferred) from disks?

   answer:
   200/100*500 + 200 = 1200
   partial credit for 500/100*200 + 500 = 1500

   (b) How many disk **seeks** are expected?

   answer:
   $2 \times 200/100 = 4$
   partial credit for $2 \times 200/100 = 4$

4. [Hash Join] Suppose there is a transaction the joins two relations (tables) using 'Hash-Join' algorithm using 401 buffer pages. One page is reserved for the output. Both inner and outer relations are not sorted, keys are uniformly distributed. The size of one relation is 400 pages, and the other is 1000 pages.

   (a) how many pages will be **read** from disks?

   answer:

   1400

   (b) And, how many disk **seeks** are expected?

   answer:

   2

5. [Query Optimization] Suppose you are have the following three tables - `Players`, `Clubs`, and `TopScorers` in a DB schema that models club soccer players.

Players (10,000 records)

player	national_team	club_team	back_no
Benzema	France	R. Madrid	9
De Bruyne	Belgium	Man City	17
Kane	England	Tottenham	10
Lewandowski	Poland	Barcelona	9
Mbappe	France	PSG	10
Messi	Argentina	PSG	10
Nalgangdo	Korea(?)	NULL	7
Neuer	Germany	B. Munich	1
Neymar	Brazil	PSG	10
Son	Korea	Tottenham	7
van Dijk	Netherland	Liverpool	4
...			
...			

Clubs (200 records)

club_team	league
Arsenal	Premier
PSG	Ligue 1
Barcelona	La Liga
Man City	Premier
Liverpool	Premier
B. Munich	Bundesliga
...	

TopScorers (1000 records)

rank	player	score
1	Mbappe	5
2	Messi	3
3	Morata	3
4	Richarlison	3
5	Lewandowski	2
6	Gue-sung Cho	2
...		
997	Benzema	0
998	De Bruyne	0
1000	Nalgangdo	0

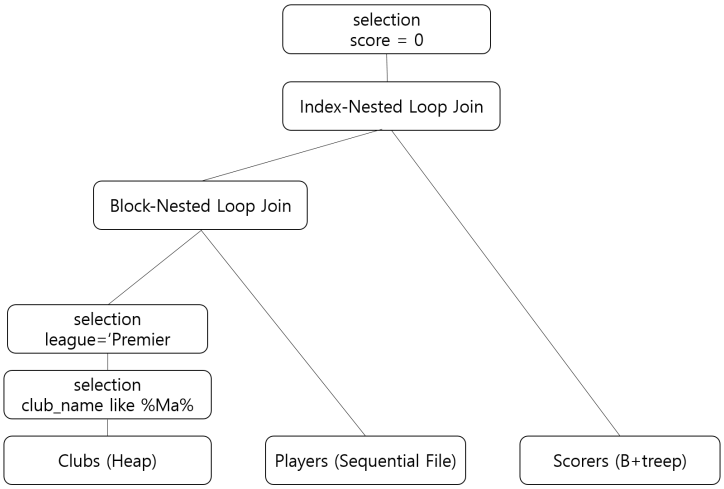
`Players` table has ten thousands of records stored in a sequential file sorted by `player` name.
`Clubs` table has hundreds of records and it is stored in a HeapFile.
`TopScorers` table has a thousand records stored in a B+tree file indexed by `player`.
Note that very few players have scored in this World Cup, so most score fields in the `TopScorers` table are zero.

Consider the following query.

```
SELECT national_team, back_no
FROM Players JOIN TopScorers
WHERE score = 0 AND
    club_team IN (
        SELECT club_team
        FROM Clubs
        WHERE club_team LIKE ‘%Ma%’ AND league == ‘Premier’ );
```

Draw an expression tree for a query execution plan that you think the most efficient. Explain why your query plan will be efficient.

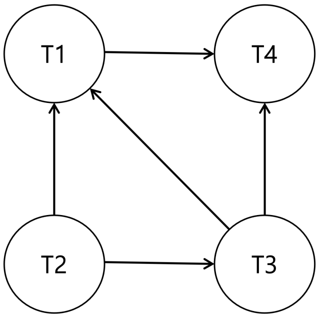
answer:



6. [Serializability] Consider the following concurrent schedule performed in a DBMS where snapshot isolation is NOT provided through the buffer cache.

T1	T2	T3	T4
start			
read(X)			start
write(X)			
			read(X)
		start	
		read(Y)	
		write(Y)	
	start		read(Y)
	read(Z)		commit
	write(Z)		
	commit	read(Z)	
write(Z)		commit	
commit			

(a) Draw a precedence graph:



(b) Is this conflict serializable?

answer:

Yes. No cycle in the graph.

7. [Isolation] Consider the following concurrent schedule performed in a DBMS where snapshot isolation IS provided through the buffer cache.

T1	T2	T3
start		
	start	
	W(Q) Q←1	
	commit	
R(Q): Q → 1		
		start
		W(Q): Q←2
R(Q): Q → 1		
		commit
R(Q): Q → 2		
commit		

In what transaction isolation level (SQL-92) is being used for the transactions?

answer:

Read Committed mode

8. [TSO] Consider the following concurrent schedule under Timestamp-Ordering protocol (TSO).

Time	T1	T2	T3
1	start		
2	read(X)	start	
3	write(X)		
4		read(Y)	start
5			read(Y)
6		read(X)	
7			read(X)
8			write(X)
9	read(X)		commit? abort?
10	commit? abort?	write(Y)	
11		commit? abort?	

Time	W-TS(X)	R-TS(X)	W-TS(Y)	R-TS(Y)
1	0	0	0	0
2		**1**		
3	**1**			
4				**2**
5				**3**
6		**2**		
7		**3**		
8	**3**			
9				
10				
11				

(a) Fill the right table for the corresponding read/write operation shown in the left table. Just fill the cells where a value changes.

(b) Will T1, T2, and T3 commit or abort?

answer:
T1: abort, T2: abort, T3: commit

9. [MVCC] Consider the following schedules under MV2PL and Snapshot Isolation. In each concurrency control protocol, indicate whether each transaction commits or aborts.

MV2PL

Time	T1	T2
1	start	
2	S-Lock(A)	start
3	read(A)	X-Lock(A) → blocked
4	X-Lock(B)	
5	write(B)	
6	Unlock(A)	X-Lock(A) → granted
7	Unlock(B)	S-Lock(B)
8	commit? abort?	write(A)
9		read(B)
11		Unlock(B)
		Unlock(A)
		commit? abort?

Snapshot Isolation

Time	T1	T2
1	start	
2		start
3	read(A)	
4	write(B)	write(A)
5	commit? abort?	
6		read(B)
		commit? abort?

(a) MV2PL T1: commit
(b) MV2PL T2: commit
(c) Snapshot Isolation T1: commit
(d) Snapshot Isolation T2: commit

10. [Recovery] Suppose the system crashed after Log Sequence No 13. The checkpoint file created by Log Sequence No 11 is valid.

```
LogSeqNo Log
  1     <T1 Start>
  2     <T1, A, 0, 10>
  3     <T2 Start>
  4     <T2, B, 0, 100>
  5     <checkpoint {T1,T2}>
  6     <T1 A, 10, 20>
  7     <T1 commit>
  8     <T3 start>
  9     <T3, A, 20, 30>
 10     <T3, C, 0, 1000>
 11     <checkpoint {T2,T3}>
 12     <T3, C, 1000, 2000>
 13     <T2, B, 100, 200>
 >>>    CRASH!!!!
```

What log entries will the recovery process append to recover from the failure?

answer:

```
14 <T2, B, 100>
15 <T3, C, 1000>
16 <T3, C, 0>
17 <T3, A, 20>
18 <T3, abort>
19 <T2, B, 0>
20 <T2, abort>
```

- This is the END -