SWE3003 Introduction to Database Systems - Final Fall 2023

Student ID	Name

For Instructor/TA only,

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Total

Academic Honor Pledge

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

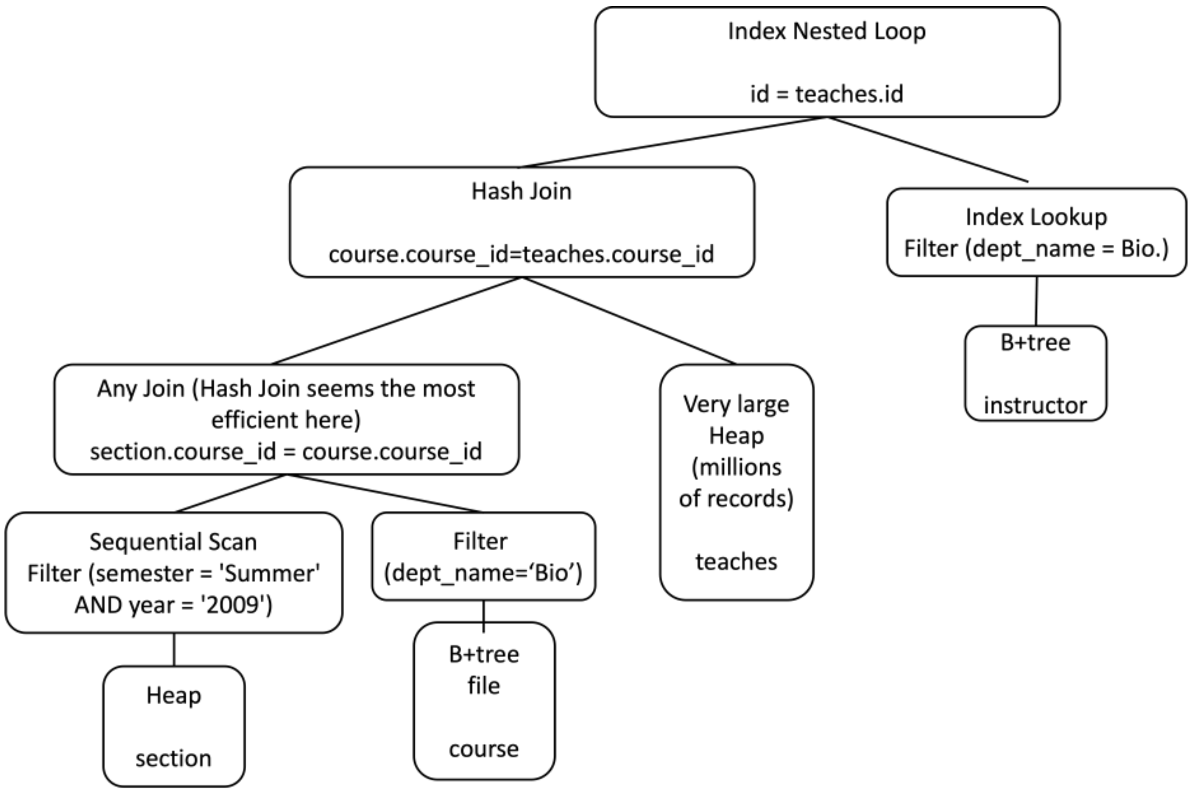
Your signature: ____________________

1. [Query Optimization (10 pts)] Suppose you are have `instructor`, `teaches`, `course`, and `section` tables. `instructor` table has hundreds of records and it is stored in a B+tree file indexed by `instructor_id`. `teaches` table has millions of records in a heap file without an index. The primary key of `teaches` table is `(instructor_id, course_id)`. `course` table has thousands of records and it is stored in a B+tree file indexed by `course_id`. `section` table has thousands of records stored in a heap file without an index. The primary key of `section` table is `(course_id, section_no)`. Consider the following query.

```
SELECT instructor.name, course.course_id, building, room_number
  FROM instructor NATURAL JOIN teaches NATURAL JOIN course NATURAL JOIN section
 WHERE course.dept_name = 'Biology'
   AND section.semester = 'Summer'
   AND section.year = '2009';
```

Draw an expression tree for a query execution plan that you think the most efficient. Describe the algorithm employed for non-trivial operations at each tree node. Explain why your query plan will be efficient. If necessary, you may make specific assumptions on the database.

answer:
+ Left-deep join tree
+ select operation before join operation
+ Hash join for teaches table



Index Nested Loop
id = teaches.id
Hash Join
course.course_id=teaches.course_id
Index Lookup
Filter (dept_name = Bio.)
B+tree
instructor
Any Join (Hash Join seems the most efficient here)
section.course_id = course.course_id
Very large Heap (millions of records)
teaches
Sequential Scan
Filter (semester = 'Summer' AND year = '2009')
Filter
(dept_name='Bio')
Heap
section
B+tree file
course

2. [Serializability (10pts)] Consider the following schedules where time increases from top to bottom.

time	T1	T2	T3
1			R(B)
2	R(A)		
3		R(B)	
4	W(B)		
5		R(A)	
6			W(A)
7	commit		
8		commit	
9			commit

Table 1: Schedule 1

time	T1	T2	T3
1			R(A)
2	R(A)		
3		R(A)	
4			W(A)
5			commit
6	W(B)		
7	commit		
8		R(B)	
9		commit	

Table 2: Schedule 2

(a) Is `Schedule 1` conflict-serializable? If so, provide a serial schedule equivalent to this schedule. If not, justify why this schedule is not conflict-serializable.

answer:
T3: R(B) − > T1: W(B)
T2: R(B) − > T1: W(B)
T1: R(A) − > T3: W(A)
T2: R(A) − > T1: W(A)
This schedule has a cyclic dependency. Therefore, this schedule cannot be transformed into a serial schedule by swapping non-conflicting operations.

(b) Is `Schedule 2` conflict-serializable? If so, provide a serial schedule equivalent to this schedule. If not, justify why this schedule is not conflict-serializable.

answer:
There's no cyclic dependency. So, this schedule is conflict-seerializable.

time	T1	T2	T3
1	R(A)		
2	W(B)		
3	commit		
4		R(A)	
5		R(B)	
6		commit	
7			R(A)
8			W(A)
9			commit

3. [2PL (10 pts)] For each of the following statements, indicate whether it is TRUE or FALSE. You will get 2 points for each correct answer, -2 points for each incorrect answer, and 0 point for each answer left blank or both answers marked.

	T	F
(a) Under two-phase locking, once a transaction releases a lock, it can no longer acquire any new locks. . . . . T	☐	☐
(b) Schedules that are conflict serializable have to be produced by two-phase locking. . . . . F	☐	☐
(c) Strict two-phase locking is both necessary and sufficient to guarantee conflict serializability. . . . . F	☐	☐
(d) Wound-wait and wait-die algorithms are pessimistic deadlock prevention algorithms and can cause more transaction aborts than needed. . . . . . . T	☐	☐
(e) Schedules produced by two-phase locking are guaranteed to prevent cascading aborts. . . . . F	☐	☐

4. [Timestamp-ordering (10pts)] Consider the following schedule under timestamp-ordering protocol. Initially, the write and read timestamps of data A, B, and C are all 0, and the timestamps of T1, T2, and T3 are 1, 2, and 3, respectively.

time	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	R(B)	
6		W(B)

Table 3: Schedule 3

time	T1	T2	T3
1	R(A)		
2		R(A)	
3			R(A)
4		R(B)	
5	R(C)		
6			W(A)
7		W(B)	
8	W(C)		
9		R(C)	
10			R(B)

Table 4: Schedule 4

(a) Does Schedule 3 abort any transaction under timestamp-ordering protocol? Justify your answer.

answer: No. This is a valid schedule under TSO.

time	T1	T2	R-ts(A)	W-ts(A)	R-ts(B)	W-ts(B)
1	R(A)		1	0	0	0
2		R(A)	2	0	0	0
3		W(A)	2	2	0	0
4		R(B)	2	2	2	0
5	R(B)		2	2	2	0
6		W(B)	2	2	2	2

(b) Does Schedule 4 abort any transaction under timestamp-ordering protocol? Justify your answer.

answer:
No. This is a valid schedule under TSO.

	T1	T2	T3	R-ts(A)	W-ts(A)	R-ts(B)	W-ts(B)	R-ts(C)	W-ts(C)
1	R(A)			1	0	0	0	0	0
2		R(A)		2	0	0	0	0	0
3			R(A)	3	0	0	0	0	0
4		R(B)		3	0	2	0	0	0
5	R(C)			3	0	2	0	1	0
6			W(A)	3	3	2	0	1	0
7		W(B)		3	3	2	2	1	0
8	W(C)			3	3	2	2	1	1
9		R(C)		3	3	2	2	2	1
10			R(B)	3	3	3	2	2	1

5. [Multi-version Timestamp Ordering (10 pts)] Consider the following schedule under multi-version timestamp-ordering (MVTSO) protocol. Initially, the write and read timestamps of data `A`, `B`, and `C` are all 0, and the timestamps of `T1`, `T2`, `T1`, and `T3` are 1, 2, and 3, respectively.

time	T1	T2
1	R(A)	
2		R(A)
3	W(A)	
4		W(A)

Table 5: Schedule 5

time	T1	T2	T3
1	R(A)		
2		R(A)	
3			R(A)
4		R(B)	
5	R(C)		
6			W(A)
7		W(B)	
8	W(C)		
9		R(C)	
10			R(B)

Table 6: Schedule 6

(a) Does Schedule 5 abort any transaction under MVTSO protocol? Justify your answer.

answer:
This is NOT a valid schedule under MVTSO.

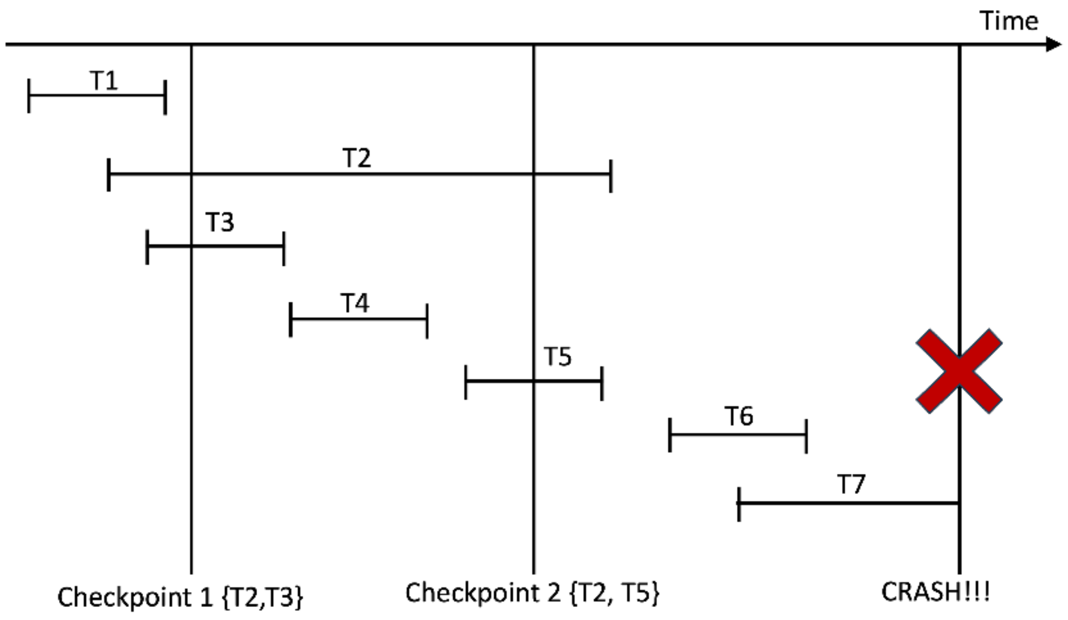
time	T1	T2	R-ts(A)	W-ts(A)
1	R(A)		1	0
2		R(A)	2	0
3	W(A)		2	1 (uh..oh)

T1's W(A) will find out that R-timestamp(A)=2. Therefore, T1 cannot perform W(A) because it's like creating an old version that should not exist from the perspective of T2. So, T1 will be aborted.

(b) Does Schedule 6 abort any transaction under MVTSO protocol? Justify your answer.

answer:
Same with 4(b). No transaction will be aborted under MVTSO.

6. [Recovery (10pts)] The following figure illustrates when concurrent transactions, checkpointing, and a crash occurred over time. Two checkpointing occurred before the system crashed. The DBMS runs under the snapshot isolation, and thus each transaction has its own private volatile workspace. The DBMS buffers I/Os, and dirty pages are not guaranteed to be flushed to disks even when its transaction commits.



(a) Which transactions should the recovery process redo and which transactions should it undo?

REDO:
T2, T5, T6 and T7

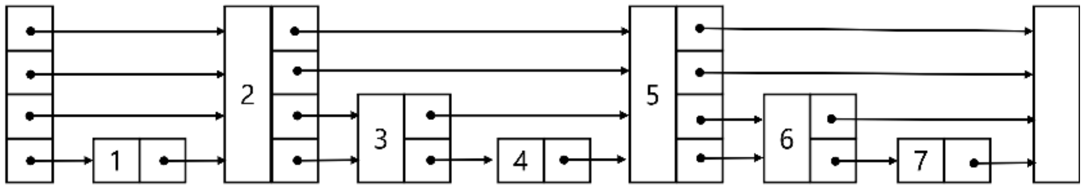
UNDO:
T7

(b) Explain why redo is necessary and why undo is necessary.

WHY REDO?:
As the problem describes, dirty pages written by committed transactions might not have been flushed to disks. So, redo is required for the transactions that were active during the checkpointing. Also, the transsactions executed after the last checkpointing must be replayed.

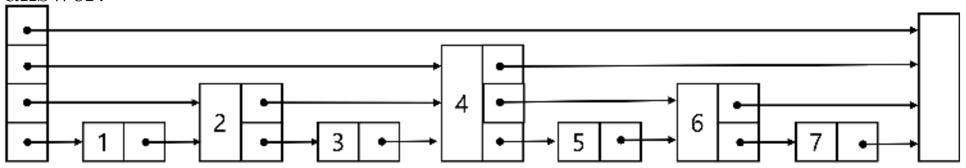
WHY UNDO?:
All operations of an aborted transaction are replayed during the redo phase. During the undo phase, aborted transactions are identified, and their operations need to be undone.

7. [Skip Lists (10 pts)] Consider the following Skip Lists. Although 7 keys are indexed, when searching for 7 in this Skip Lists, nodes 2, 5, 6, and 7 are visited sequentially, i.e., the search time is 4. Similarly, when searching for 4, node 2 is visited once, node 5 is visited twice, and nodes 3 and 4 are visited again for a total of 5 key comparisons. Therefore, the search complexity is not guaranteed to be $log_2(N)$ where N is the number of keys. Note that, in a binary search tree, you can also find a similar problem when keys are skewed. However, unlike binary search tree, the structure of Skip Lists is not determined by the insertion order, but by the height of nodes randomly selected.



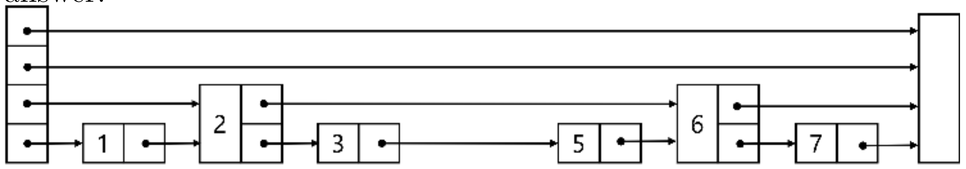
(a) Please modify the structure of the given Skip Lists (change the height of each node) to achieve a worst complexity of $log_2(N)$.

answer:



(b) How will your Skip Lists change after deleting the key '4'?

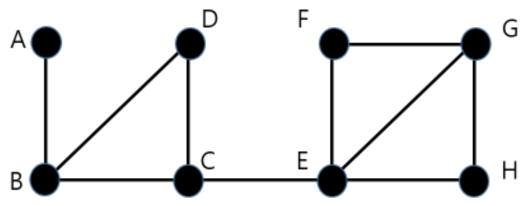
answer:



8. [NoSQL (10 pts)] For each of the following statements, indicate whether it is TRUE or FALSE. You will get 2 points for each correct answer, -2 points for each incorrect answer, and 0 point for each answer left blank or both answers marked.

		T	F
(f)	In LSM-trees, write operations are generally faster than read operations. T	☐	☐
(g)	Compaction in LSM-trees is a process that helps to manage and reduce the number of SSTables. . . . . T	☐	☐
(h)	LSM-trees are particularly well-suited for write-intensive workloads and scenarios with a high volume of insert/update operations. . . . . T	☐	☐
(i)	MongoDB allows for dynamic modification of the database schema through flexible DDL statements. . . . . F	☐	☐
(j)	MongoDB supports ACID transactions, making it suitable for applications with complex transactional requirements. . . . . F	☐	☐

9. [Graph Processing (5 pts)] Consider the following graph. We want to perform distributed parallel processing on this graph using two machines. How will you cut this graph? Explain why you think it is the most efficient partition.



answer:
C-E should be cut. This will minimize the number of ghost vertices.