Subject: **Database Management Systems** (**CO3021**)
Semester 1, 2018-2019

Chapter 5: Concurrency Control Techniques

Question 5.1. State the purposes of concurrency control. Give several examples.

Question 5.2. What is a lock? Give an example.

Question 5.3. What is two-phase locking? How does it ensure serializability? Give an example.

Question 5.4. Differentiate the variations of the two-phase locking technique: conservative, strict, rigorous.

Question 5.5. Describe deadlock and starvation problems and the approaches to dealing with them. Give an example.

Question 5.6. What is a wait-for graph? How can it detect deadlock?

Question 5.7. Given the following schedule. Draw the wait-for graph before and after the last action *write_lock*(*A*) of transaction T3. Will deadlock occur? Why?

T1	T2	T3	T4
read_lock(A);			
read_item(A);			
	write_lock(B);		
	write_item(B);		
read_lock(B);			
		read_lock(C);	
		read_item(C);	
	write_lock(C);		
			write_lock(B);
		write_lock(A);	

Question 5.8. Describe the two-phase locking technique using certify locks. Give an example.

Question 5.9. Describe the multiple granularity level two-phase locking technique. Give an example.

Question 5.10. Describe the lock compatibility matrix for multiple granularity locking. Why is IS compatible with SIX? Why is IX incompatible with SIX?

Question 5.11. Describe the optimistic concurrency control technique. How different is it from other concurrency control techniques?

Question 5.12. Describe the timestamp ordering technique. How does it ensure serializability? Give an example.

Question 5.13. Differentiate strict timestamp ordering from basic timestamp ordering.

Question 5.14. Describe the multiversion technique based on timestamp ordering. Give an example.

Question 5.15. What is the main difference between the multiversion techniques and their corresponding basic techniques? Give an example.

Question 5.16. Describe index locking. Give an example.

Question 5.17. What are latches? When are they used?

Question 5.18. What is an interactive transaction? Give an example.

Question 5.19. What is a phantom record? Describe its problem for concurrency control.

Question 5.20. Describe two-phase locking for insertions and deletions.

Question 5.21. Given two following transactions. Form their schedule with read_lock, write_lock, and unlock operations using two-phase locking. Will deadlock occur with their execution? Explain in detail.

T1
read_item(X);
read_item(Y);
X := X+Y;
write_item(X);

T2
read_item(X);
read_item(Y);
Y := Y-X;
write_item(Y);

Question 5.22. Consider the set of transactions accessing database element A. These transactions are operating under a basic timestamp-based scheduler. Explain why the transaction T3 has to be aborted. What happens if these transactions are operating under a multiversion timestamp-based scheduler?

T1	T2	T3	T4	A
150	200	175	225	RT=0 WT=0
$r_1(A)$				RT=150
$w_1(A)$				WT=150
	$r_2(A)$			RT=200
	$w_2(A)$			WT=200
		$r_3(A)$ ABORTED		
			$r_4(A)$	RT=225

Question 5.23. Consider the relation Movie(title, year, length, studioName) in the database Studio.

Transaction T1 consists of the query:

SELECT * FROM Movie

WHERE title = 'King Kong';

Transaction T2 consists of the query:

UPDATE Movie SET year = 1939

WHERE title = 'Gone with the wind';

Assume that there are two records in relation Movie with the title 'King Kong' and there is one record with the title 'Gone with the wind'.

Suggest the collection of locks for this situation using multiple granularity locking.

Question 5.24. Given two transactions and their schedules. Which schedule is serializable? Which schedule faces deadlock?

T1	T2
$r_1(X)$	$r_2(Z)$
$r_1(Z)$	$r_2(Y)$
$w_1(X)$	$w_2(Z)$
$r_1(Y)$	$r_2(X)$
$w_1(Z)$	

S1

T1	T2
	write_lock(Z)
	$r_2(Z)$
	read_lock(Y)
	$r_2(Y)$
	$w_2(Z)$
	read_lock(X)
	$r_2(X)$
	unlock(X)
read_lock(X)	
$r_1(X)$	
	unlock(Z)
write_lock(Z)	
$r_1(Z)$	
write_lock(X)	
$w_1(X)$	
	unlock(Y)
read_lock(Y)	
$r_1(Y)$	
$w_1(Z)$	
unlock(Z)	
unlock(Y)	
unlock(X)	

S2

T1	T2
	write_lock(Z)
	$r_2(Z)$
write_lock(X)	
$r_1(X)$	
	read_lock(Y)
	$r_2(Y)$
	$w_2(Z)$
write_lock(Z)	
	read_lock(X)
	$r_2(X)$
	unlock(X)
$r_1(Z)$	
$w_1(X)$	
	unlock(Z)
	unlock(Y)
read_lock(Y)	
$r_1(Y)$	
$w_1(Z)$	
unlock(Z)	
unlock(Y)	
unlock(X)	