CS 186 Introduction to Database Systems
Fall 2020 Alvin Cheung, Aditya Parameswaran

DIS 9

1 Conflict Serializability

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

(a) Draw the dependency graph (precedence graph) for the schedule.

(b) Is this schedule conflict serializable? If so, what are all the conflict equivalent serial schedules? If not, why not?

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

(c) Draw the dependency graph (precedence graph) for the schedule.

(d) Is this schedule conflict serializable? If so, what are all the conflict equivalent serial schedules? If not, why not?

2 Deadlock

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

(a) Draw a "waits-for" graph and state whether or not there is a deadlock.

(b) If we try to avoid deadlock by using the wait-die deadlock avoidance policy, would any transactions be aborted? Assume T1 priority > T2 > T3 > T4.

3 Locking

T1	T2
Lock_X(B)	
Read(B)	
B := B * 10	
Write(B)	
Lock_X(F)	
Unlock(B)	
	Lock_S(F)
F := B * 100	
Write(F)	
Commit	
Unlock(F)	
	Read(F)
	Unlock(F)
	Lock_S(B)
	Read(B)
	Print(F + B)
	Commit
	Unlock(B)

(a) What is printed, assuming we initially have B = 3 and F = 300?

(b) Does the execution use 2PL or strict 2PL?

(c) Would moving Unlock(F) in the second transaction to any point after Lock_S(B) change this (or keep it) in 2PL?

(d) Would moving Unlock(F) in the first transaction and Unlock(F) in the second transaction to the end of their respective transactions change this (or keep it) in strict 2PL?

(e) Would moving Unlock(B) in the first transaction and Unlock(F) in the second transaction to the end of their respective transactions change this (or keep it) in strict 2PL?

4 Multigranularity Locking

(a) Suppose a transaction T1 wants to scan a table R and update a few of its tuples. What kinds of locks should T1 have on R, the pages of R, and the updated tuples?

(b) Is an S lock compatible with an IX lock?

(c) Consider a table which contains two pages with three tuples each, with Page 1 containing Tuples 1, 2, and 3, and Page 2 containing Tuples 4, 5, and 6.

  (a) Given that a transaction T1 has an IX lock on the table, an IX lock on Page 1, and an X lock on Tuple 1, which locks could be granted to a second transaction T2 for Tuple 2?

  (b) Given that a transaction T1 has an IS lock on the table and an S lock on Page 1, what locks could be granted to a second transaction T2 for Page 1?

5 Project Prep

The LockManager in the project will have the following 4 functions:

- `acquire(transaction, resourceName, lockType)`: this method is the standard acquire method of a lock manager. It allows a transaction to request one lock, and grants the request if there is no queue and the request is compatible with existing locks. Otherwise, it should queue the request (at the back) and block the transaction
- `release(transaction, resourceName)`: this method is the standard release method of a lock manager. It allows a transaction to release one lock that it holds.
- `acquireAndRelease(transaction, resourceName, lockType, releaseLocks)`: this method atomically acquires one lock and releases zero or more locks. This method has priority over any queued requests (it should proceed even if there is a queue, and it is placed in the front of the queue if it cannot proceed).
- `promote(transaction, resourceName, lockType)`: this method allows a transaction to explicitly promote/upgrade a held lock. This method has priority over any queued requests (it should proceed even if there is a queue, and it is placed in the front of the queue if it cannot proceed).

In this problem we will have 3 transactions (T_1, T_2, T_3) and 2 resources that we will try to lock (A, B). After each lock request is processed, fill out the following two tables that store the information for what locks are held:

transactionLocks	Locks Held
T_1	
T_2	
T_3	

resourceEntries	Locks	Queue
A		
B		

The following lock requests are made in this order:

- acquire(T_1, A, X)
- acquire(T_2, B, S)
- acquire(T_3, B, X)
- acquireAndRelease(T_2, A, S, releaseLocks=<B>)
- release(T_1, A)
- acquire(T_1, A, S)
- acquire(T_3, A, X)
- promote(T_2, A, X)
- release(T_1, A)