Chapter 6 - Transaction Management: Concurrency Control

Zhe Li

December 2020

1 Serializability

Our database is not static. In fact, it is frequently updated by queries like:

- INSERT
- DELETE
- UPDATE

These queries trigger disk reads and writes. And we defined transaction as a sequence of disk reads and writes that are expected to be executed as a unit. We learnt that DBMS schedules the transactions by interleaving actions of different transactions.

1.1 Fundamentals

1.1.1 Where are the details?

Please briefly explain why we only care about the disk I/Os and their objects without paying attention to the details of the queries.

The details are stripped away to help reason about correctness of schedules more easily.

1.1.2 Why do we bother?

Please briefly explain why we ever bother with producing the schedules of how to interleaving the actions of transactions.

Disk accesses are frequent and relatively slow. It is wise to run several programs concurrently to prevent CPU from idling for most of the times.

1.1.3 Caution: Strong ACID

In transaction scheduling, even if the actions of the transactions are interleaved, we want the result to be the same as the transactions are executed in serial. Which transaction property among the ACID is being preserved by the serializability of the schedule? Please name it and briefly define that property.

Isolation. Concurrent transactions must not interfere with each other.

1.2 Conflict Serializable

Determining whether a schedule is serializable is hard. So we want to use a stronger notion of serializability so that it's easier for us to check. For this question, we will use conflict serializability.

1.2.1 The first schedule

Below is a schedule involving three transactions:

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

Please draw a precedence graph and determine if the above schedule is conflict serializable. If it is, provide its equivalent serial schedule.

$T3 > T1 > T2 > T3 > T2$. It is cyclic so that it is not conflict serializable.

1.2.2 The second schedule

Below is a schedule involving three transactions:

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

Please draw a precedence graph and determine if the above schedule is conflict serializable. If it is, provide its equivalent serial schedule.

$T3 > T1 > T2$. It is acyclic, and the serial schedule is $T3 > T1 > T2$.

1.3 Two-Phase Locking

The Two-Phase Locking protocol is a great tool to help generating conflict serializable schedules. It is easy to use and makes the guarantee.

1.3.1 Lock and unlock

There are two transactions listed below:

- T_1: R(X), W(X), R(Y), W(Y).
- T_2: R(X), R(T), W(X), W(Y).

Someone told us that all the locks in the following schedule are exclusive and the schedule is appropriate and produces conflict serializable schedule: $T_1 > T_2$. Please determine whether the statement is true and briefly explain.

T_1	T_2
L(X)	
R(X)	
W(X)	
L(Y)	
U(X)	
	L(X)
	R(X)
R(Y)	
W(Y)	
U(Y)	
	L(Y)
	R(Y)
	W(X)
	W(Y)
	U(X)
	U(Y)

The statement is true. There is no transaction requesting lock after its first release. Also, there is no deadlock. As T_1 enters the shrinking phase first, so it is the first in the equivalent serial schedule.

1.3.2 Deadlock

We are given the following schedule which uses 2PL. We know that all of them are in the growing phase. Determine if there is a deadlock and justify.

T1	T2	T3	T4	T5	T6
W(F)					
	R(E)				
		W(D)			
		R(F)			
			R(C)		
			W(C)		
			W(E)		
				R(B)	
				W(D)	
					R(A)
	R(F)				
W(A)					

$T_3 > T_1, T_4 > T_2, T_5 > T_3, T_2 > T_1, T_1 > T_6$. There is no cycle, so there is no deadlock.