CS4347 : Database Systems
Homework Assignment 6

Matthew McMillianmgm160130@utdallas.edu

November 11, 2018

1. Add the operation commit at the end of each of the transactions T1 and T2 in Figure 20.2, and then list all possible schedules for the modified transactions. Determine which of the schedules are recoverable, which are cascadeless, and which are strict.

Recoverable Schedules

T1	T2
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
y := y + n	
write_item(y)	
commit;	
	read_item(x)
	x := x + m
	write_item(x)
	commit;

T1	T2
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
	read_item(x)
y := y + n	
write_item(y)	
commit;	
	x := x + m
	write_item(x)
	commit;

Cascadless Schedules

T1	T2
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
y := y + n	
write_item(y)	
commit;	
	read_item(x)
	x := x + m
	write_item(x)
	commit;

T1	T2
	read_item(x)
	x := x + m
	write_item(x)
	commit;
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
y := y + n	
write_item(y)	
commit;	

Strict Schedules

T1	T2
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
y := y + n	
write_item(y)	
commit;	
	read_item(x)
	x := x + m
	write_item(x)
	commit;

T1	T2
	read_item(x)
	x := x + m
	write_item(x)
	commit;
read_item(x)	
x := x - n	
write_item(x)	
read_item(y)	
y := y + n	
write_item(y)	
commit;	

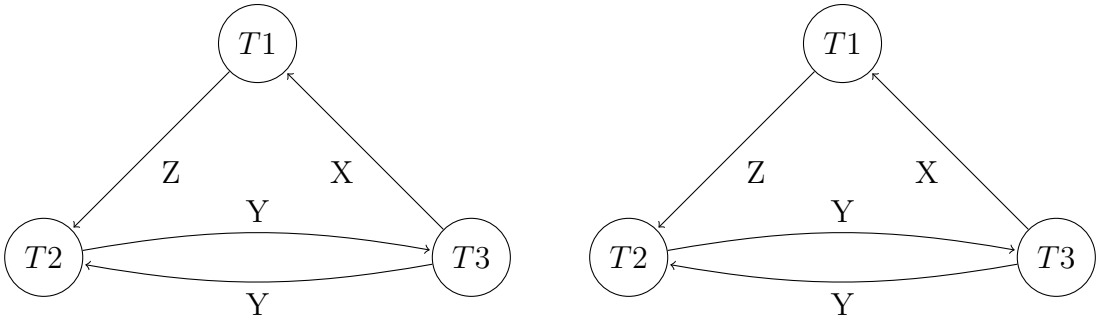
Interestingly enough, for this example we can use the same cascading schedule to define our strict schedule. We could vary if it we had a commit before the read_item(y) part.

2. List all possible schedules for transactions T1 and T2 in Figure 20.2, and determine which are conflict serializable (correct) and which are not.

   In the schedules listed above, the cascading and strict schedules are conflict equivalent. The recoverable schedules are conflict equivalent to themselves, however in general the recoverable and cascadeless solutions are not conflict equivalent to the recoverable schedules.

3. Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below. Draw the serializability (precedence) graphs for S1 and S2, and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).

Precedence Graphs for S1 and S2



Both of the precedence graphs are the same. We can conclude that since there is a cycle between two nodes that neither processes are serializable.

4. Consider schedules S3, S4, and S5 below. Determine whether each schedule is strict, cascadeless, recoverable, or nonrecoverable. (Determine the strictest recoverability condition that each schedule satisfies.)

   - A recoverable schedule is one where if a transaction reads an item previously written by another transaction, the commit operation must occur for the first transaction first. This is true of the first sequence. w3(y) happens before r2(y), however T3 commits before T2 so this makes this a recoverable solution.
   - This is a non-recoverable schedule. It is not cascadeless since no items are read only after committed transactions, items do not follow a strict schedule since a commit doesn't occur before other reads,, and it is not recoverable since nothing reads after write operations.
   - This is a non-recoverable schedule. It is not cascadeless since no items are read only after committed transactions, items do not follow a strict schedule since a commit doesn't occur before other reads, and it is not recoverable since nothing reads after write operations.