Tutorial 9: Transaction Processing Concepts and Theory

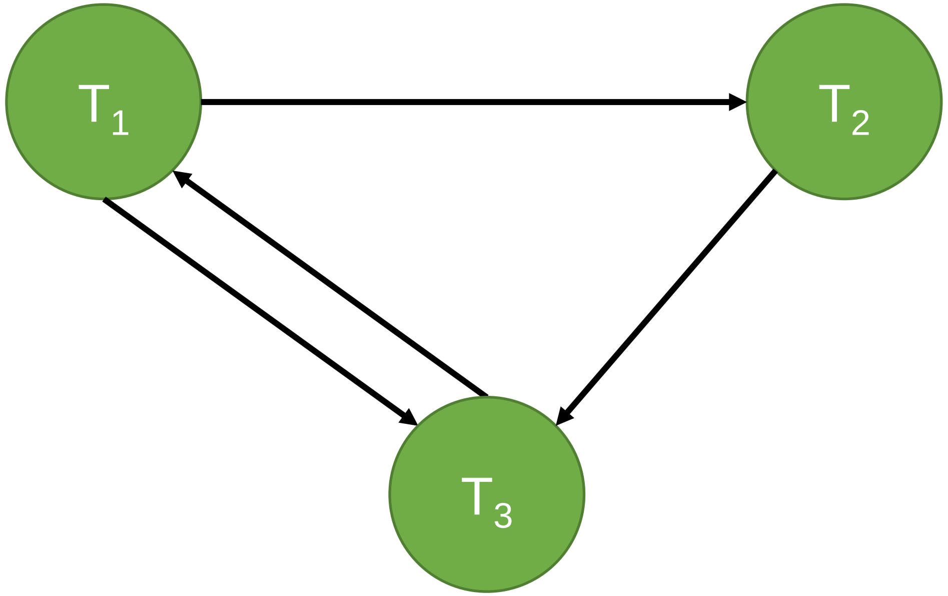
CS3402 Database Systems

Question 1

- Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.
  a) A: $r_1(X)$; $r_3(X)$; $w_1(X)$; $r_2(X)$; $w_3(X)$;
  b) B: $r_1(X)$; $r_3(X)$; $w_3(X)$; $w_1(X)$; $r_2(X)$;
  c) C: $r_3(X)$; $r_2(X)$; $w_3(X)$; $r_1(X)$; $w_1(X)$;

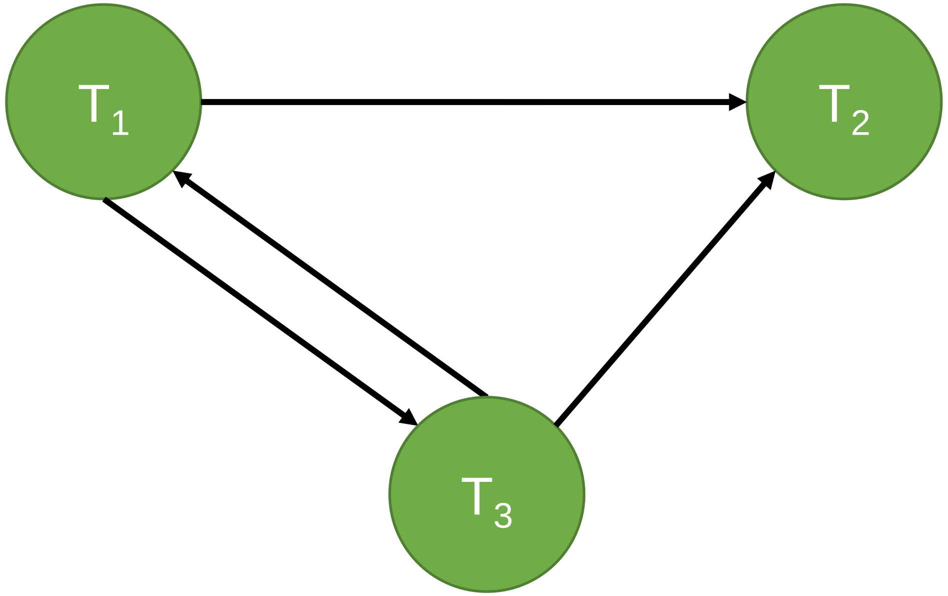
Question 1a (Answer)

- A: $r_1(X)$; $r_3(X)$; $w_1(X)$; $r_2(X)$; $w_3(X)$;
- Step 1: Add three nodes T_1, T_2 and T_3
- Step 2: Add edge for read-write conflict
  - Add an edge from T_1 to T_3 for $r_1(X)$; $w_3(X)$;
  - Add an edge from T_3 to T_1 for $r_3(X)$; $w_1(X)$;
  - Add an edge from T_2 to T_3 for $r_2(X)$; $w_3(X)$;
- Step 3: Add edge for write-read conflict
  - Add an edge from T_1 to T_2 for $w_1(X)$; $r_2(X)$;
- Step 4: Add edge for write-write conflict
  - Add an edge from T_1 to T_3 for $w_1(X)$; $w_3(X)$;
- Step 5: A is NOT serializable as the precedence graph has two cycles $T_1 \rightarrow T_3 \rightarrow T_1$ and $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_1$.



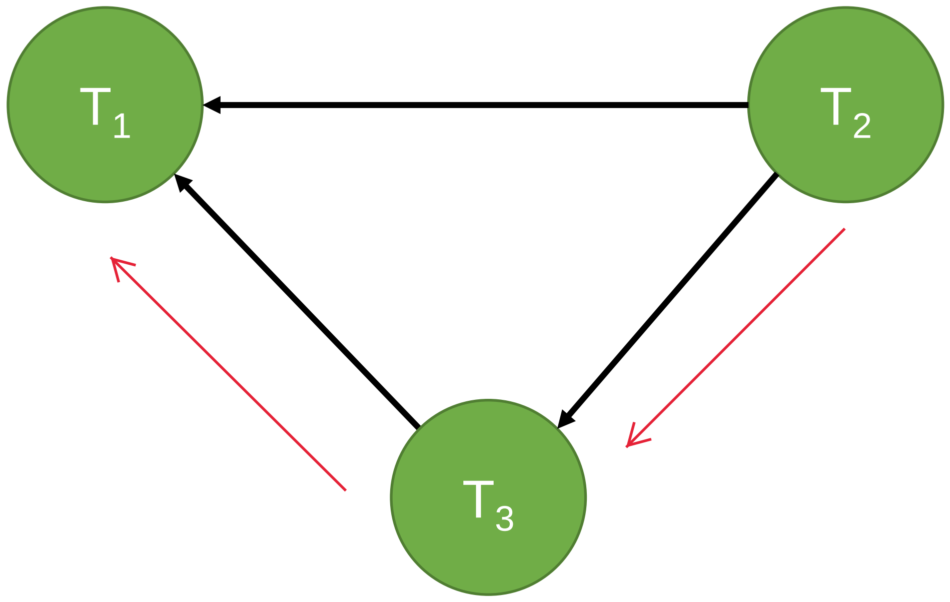
Question 1b (Answer)

- B: $r_1(X)$; $r_3(X)$; $w_3(X)$; $w_1(X)$; $r_2(X)$;
- Step 1: Add three nodes T_1, T_2 and T_3
- Step 2: Add edge for read-write conflict
  - Add an edge from T_1 and T_3 for $r_1(X)$; $w_3(X)$;
  - Add an edge from T_3 and T_1 for $r_3(X)$; $w_1(X)$;
- Step 3: Add edge for write-read conflict
  - Add an edge from T_3 and T_2 for $w_3(X)$; $r_2(X)$;
  - Add an edge from T_1 and T_2 for $w_1(X)$; $r_2(X)$;
- Step 4: Add edge for write-write conflict
  - Add an edge from T_3 and T_1 for $w_3(X)$; $w_1(X)$;
- Step 5: B is NOT serializable as the precedence graph has a cycle $T_1 \rightarrow T_3 \rightarrow T_1$.



Question 1c (Answer)

- C: $r_3(X)$; $r_2(X)$; $w_3(X)$; $r_1(X)$; $w_1(X)$;
- Step 1: Add three nodes T_1, T_2 and T_3
- Step 2: Add edge for read-write conflict
  - Add an edge from T_3 to T_1 for $r_3(X)$; $w_1(X)$;
  - Add an edge from T_2 to T_3 for $r_2(X)$; $w_3(X)$;
  - Add an edge from T_2 to T_1 for $r_2(X)$; $w_1(X)$;
- Step 3: Add edge for write-read conflict
  - Add an edge from T_3 to T_1 for $w_3(X)$; $r_1(X)$;
- Step 4: Add edge for write-write conflict
  - Add an edge from T_3 to T_1 for $w_3(X)$; $w_1(X)$;
- Step 5: C is serializable as the precedence graph has no cycles. C is equivalent to this serial schedule: T_2, T_3, T_1.



Question 2

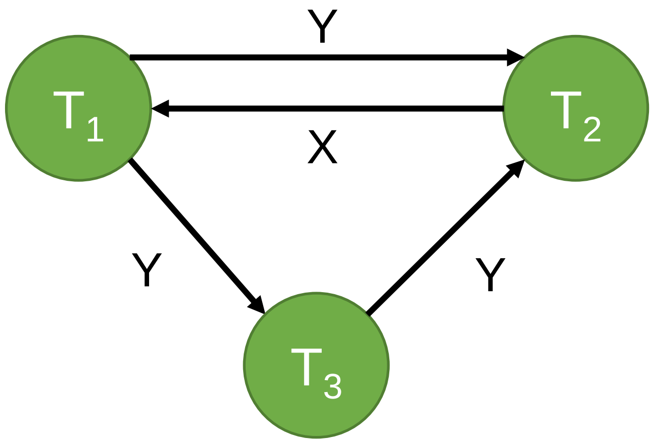
- Consider the following concurrent schedule S. Draw the serialization graph for the schedule. Is it conflict serializable?

T_1	T_2	T_3
	Read(X)	
Write(Y)		
		Read(Y)
	Write(Y)	
Write(X)		
	Commit	
		Write(Z)
Commit		
		Commit

Question 2 (Answer)

- S: $R_2(X)$; $W_1(Y)$; $R_3(Y)$; $W_2(Y)$; $W_1(X)$; C_2; $W_3(Z)$; C_1; C_3;
- Step 1: Add three nodes T_1, T_2 and T_3
- Step 2: Add edge for read-write conflict
  - Add an edge from T_2 to T_1 for $R_2(X)$; $W_1(X)$;
  - Add an edge from T_3 to T_2 for $R_3(Y)$; $W_2(Y)$;
- Step 3: Add edge for write-read conflict
  - Add an edge from T_1 to T_3 for $W_1(Y)$; $R_3(Y)$;
- Step 4: Add edge for write-write conflict
  - Add an edge from T_1 to T_2 for $W_1(Y)$; $W_2(Y)$;
- Step 5: S is NOT serializable as the precedence graph has two cycles: $T_1 \to T_2 \to T_1$ and $T_1 \to T_3 \to T_2 \to T_1$.

T_1	T_2	T_3
	Read(X)	
Write(Y)		
		Read(Y)
	Write(Y)	
Write(X)		
	Commit	
		Write(Z)
Commit		
		Commit



Question 3

- Consider schedules S_1, S_2 and S_3 below. Determine whether each schedule is strict, cascadeless, recoverable, or nonrecoverable. Determine the strictest recoverability condition that each schedule satisfies.
  a) S_1: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; c_1;
  b) S_2: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;
  c) S_3: $r_1(X)$; $w_1(X)$; $w_2(X)$; $w_1(Y)$; c_1; $r_2(X)$; c_2;

  Can you change schedule S_3 into a strict schedule?

Types of Schedules

- An **unrecoverable** schedule is one where, a dirty read takes place.
- A schedule S is **recoverable** if no transaction T in S commits until all transactions T′ that have written some item X that T reads have committed. T cannot commit until all T’ have committed because T reads items that have been written by T’.
- A schedule is said to be **cascadeless**, if every transaction in the schedule reads only items that were written by committed transactions.
- There is a more restrictive type of schedule, called a **strict schedule**, in which transactions can neither read nor write an item X until the last transaction that wrote X has committed (or aborted).

Question 3a (Answer)

- S_1: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; c_1;
- S_1 is a non-recoverable schedule
- T_2 (i.e., $r_2(X)$) read data item X written by T_1 and T_2 committed before T_1 (i.e., a dirty read)

Question 3b (Answer)

- S_2: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;
- S_2 is a recoverable schedule because T_1 committed fore T_2 (i.e., no dirty read).

2 reads X has been wtritten by 1 and 1 has not yet committed -> not cascadeless

Question 3c (Answer)

not strict

- S_3: $r_1(X)$; $w_1(X)$; $w_2(X)$; $w_1(Y)$; c_1; $r_2(X)$; c_2;
- S_3 is a cascadeless schedule because T_2 reads data item X previously written by T_1 and the commit operation of T_1 appears before the read operation of T_2.
- Strict schedule: $r_1(X)$; $w_1(X)$; $w_1(Y)$; c_1; $w_2(X)$; $r_2(X)$; c_2;
  - $w_2(X)$; $w_1(Y)$; are non-conflicting operations, so we can swap them.
  - T_1 can be committed before $w_2(X)$;
  - T_2 only writes and reads X and Y written by committed transaction T_1.