

# **Tutorial 9: Transaction**

**CS3402 Database Systems**

# Question 1

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

## Question 2

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

T <sub>a</sub>	T <sub>b</sub>	T <sub>c</sub>
	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)  $r_1(X); w_1(X); r_2(X); r_1(Y); w_2(X); c_2; c_1;$
- b)  $r_1(X); w_1(X); r_2(X); r_1(Y); w_2(X); w_1(Y); c_1; c_2;$
- c)  $r_1(X); w_1(X); w_2(X); w_1(Y); c_1; r_2(X); c_2;$

Can you change c) into a strict schedule?

# Types of Schedules (1/2)

- A unrecoverable schedule is one where, a dirty read takes place.
- A recoverable schedule is one where, if some transaction  $T_j$  is reading value updated or written by some other transaction  $T_i$ , then the commit operation of  $T_j$  must appear after the commit operation of  $T_i$ .
- A cascadeless schedule is one where, for each pair of transactions  $T_i$  and  $T_j$  such that  $T_j$  reads data items previously written by  $T_i$ , the commit operation of  $T_i$  appears before the read operation of  $T_j$ . Cascadeless schedules are desirable because the failure of a transaction does not lead to the aborting of any other transaction.

# Types of Schedules (2/2)

- A strict schedule is one where for any two transactions  $T_i$  and  $T_j$ , if a write operation of  $T_i$  appears before a conflicting operation of  $T_j$  (either read or write), then the commit or abort operation of  $T_i$  also appears before that conflicting operation of  $T_j$ .