DATABASE SYSTEMS

CS – 355/CE – 373

Instructor: Maria N. Samad

October 30th, 2024

VIEW SERIALIZABILITY

- Sometimes a schedule may not be conflict serializable, however it may produce the same “view” as a serial schedule.
- In simpler words, a schedule’s precedence graph may have cycles, however it may give a consistent result.
- A schedule is said to be ***View-Serializable*** if it is view equivalent to a Serial Schedule (where no interleaving of transactions is possible).

VIEW SERIALIZABILITY

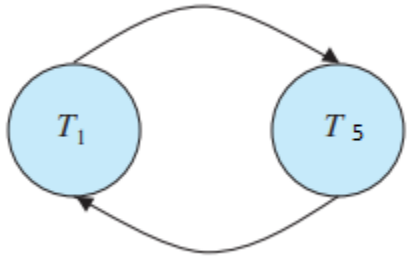
- Example: For the given schedule, check if it's conflict serializable or not. Then check whether it is view serializable or not.

T_1	T_5
read(A)	
$A := A - 50$	
write(A)	
	read(B)
	$B := B - 10$
	write(B)
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$A := A + 10$
	write(A)

Figure 17.13 Schedule 8.

VIEW SERIALIZABILITY

- Solution: To check for conflict serializability, we can use precedence graph to get the following:



- We can see there is a cycle here, so it is ***not conflict serializable***

VIEW SERIALIZABILITY

- Solution: To check for view serializability, we need to assume serial transaction, and test the code for some sample case, and compare it with the output of the given concurrent schedule. Assume A = 1000 and B = 2000 initially.
- There can be two possibilities:
  - T1 happens before T5
  - T5 happens before T1
- ***If T1 happens before T5:***
  - T1 results in: A = 950, B = 2050
  - T5 results in: B = 2040, A = 960
- ***If T5 happens before T1:***
  - T5 results in: B = 1990, A = 1010
  - T1 results in: A = 960, B = 2040
- In either case, serial execution will give these values

VIEW SERIALIZABILITY

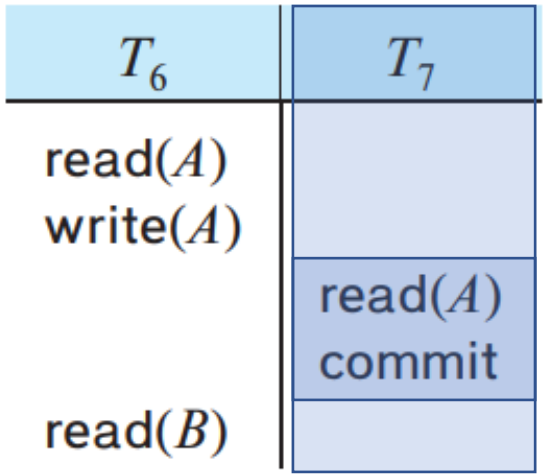
- In order for it to be view-serializable, the given concurrent schedule should also result in same values for A and B
- T1: A = 950
- T2: B = 1990
- T1: B = 2040
- T2: A = 960
- Here, we can see we are getting the same result as that expected of a serial schedule, so the answer is correct.
- View Serializability is a much more complex algorithm then simply checking for computations, but for basic arithmetic operations, we can dry run it to deduce if it is view serializable or not

TRANSACTION ISOLATION AND ATOMICITY

- In this section we consider the effect of transaction failures on concurrency.
- If a transaction T_i fails for any reason, we need to undo the effect of this transaction to ensure the atomicity property.
- To achieve this, we need to place restrictions on the types of schedules permitted in the system.

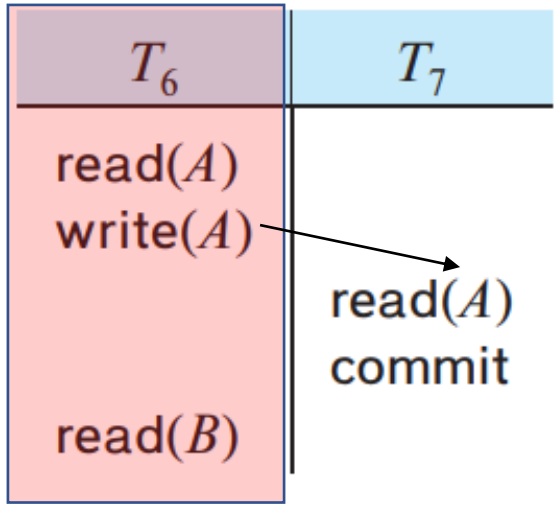
RECOVERABLE SCHEDULES

- Consider the partial schedule as shown in the following figure →
- Here, T_7 is a transaction that performs only one instruction: *read*(*A*).
- We call this a partial schedule because we have not included a commit or abort operation for T_6.
- Notice that T_7 commits immediately after executing the *read*(*A*) instruction.
- Thus, T_7 commits while T_6 is still in the active state.



RECOVERABLE SCHEDULES

- Suppose that T_6 fails before it commits. T_7 has read the value of data item A written by T_6.
- Therefore, we say that T_7 is **dependent** on T_6. Because of this, we must abort T_7 to ensure atomicity.
- However, T_7 has already committed and cannot be aborted.
- Thus, we have a situation where it is impossible to recover correctly from the failure of T_6.
- This is an example of a **nonrecoverable** schedule.



RECOVERABLE SCHEDULES

- A **recoverable schedule** is one where,
  - For each pair of transactions T_i and T_j (such that T_j reads a data item previously written by T_i)
  - the commit operation of T_i appears before the commit operation of T_j .
- For the example shown here to be recoverable, T_7 would have to delay committing until after T_6 commits.

T_6	T_7
read(A)	
write(A)	
	read(A)
	commit
read(B)	

EXAMPLE

- Consider the following schedule and deduce the following:
  1. Is it recoverable or not?
  2. If not, how to reschedule it to make it recoverable?
- Solution:
  - Write – Read dependencies:
    - T1 → T2 (Y)
    - T1 → T3 (A)
  1. To check whether recoverable or not, shift commit operations based on the dependencies
  2. On board

T1	T2	T3
read(X)		
Read(Y)		
Y=X+5		
Write(Y)		
		Read(X)
		X = X-5
		Write(X)
	Read(Y)	
Read(A)		
A=A+5		
Write(A)		
		Read(A)
		A = A+5
	Y=Y+5	
		Write(A)
		commit
	Write(Y)	
	commit	
Commit		

EXAMPLE

- Consider the following schedule and deduce the following:
  1. Is it recoverable or not?
  2. If not, how to reschedule it to make it recoverable?
- Solution:
  - Write – Read dependencies:
    - T1 → T2 (A)
    - T1 → T3 (A)
    - T2 → T3 (A)
  1. To check whether recoverable or not, shift commit operations based on the dependencies
     - Solved on board
  2. On board

T1	T2	T3
w(A)		
	r(A)	
r(B)		
		r(B)
		r(A)
	w(A)	
r(B)		
		r(A)
		W(A)
	r(B)	
Commit		
		r(B)
	w(B)	
		Commit
	Commit	

RECOVERABLE SCHEDULE

- Activity Sheet

RECOVERABLE SCHEDULE

- Activity Sheet Solution:
  - Recoverable Schedules Solution