DATABASE SYSTEMS

CS – 355/CE – 373

Instructor: Maria N. Samad

November 4th, 2024

RECOVERABLE SCHEDULE

- Even if the schedule is recoverable, there can be schedules that do not have single dependencies, or even have cyclic dependencies
- A schedule that has transitive dependencies is known to be ***cascaded***, i.e. series of dependencies that rely on the previous ones.
- Cascaded dependencies are not single dependencies, and must be handled in depth rather than only checking for the order of commit operations
  - Example: $T_1 \rightarrow T_2$ and $T_2 \rightarrow T_3$ and $T_3 \rightarrow T_4$
- A schedule has cyclic dependency when the transactions depend on each other
  - Example: $T_1 \rightarrow T_2$ and $T_2 \rightarrow T_1$

RECOVERABLE SCHEDULE

- For cascaded transactions, we deduce what is commonly known as the ***cascadeless schedule***.
- This method goes beyond just moving the commit operations to deduce it is “recoverable” or not
- Instead it converts the schedule in a form that prevents inconsistencies occurring due to failure in between, i.e. the temporary update problem

RECOVERABLE SCHEDULE

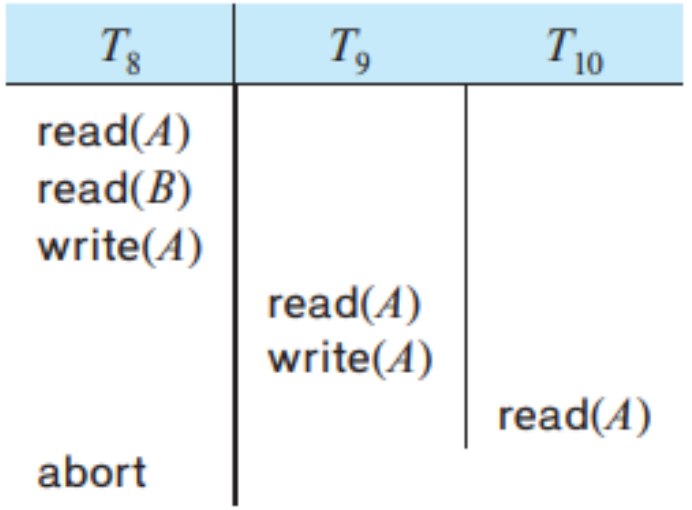
- However, schedules containing cyclic write – read dependencies are ALWAYS ***non-recoverable***, and cannot be converted into recoverable schedules either
- Thus, for cyclic dependencies, there is no need to move the commit operations or deduce cascadeless schedule, and we can directly say that if a failure occurs in such schedules, they can NEVER be converted into a recoverable state.
- The only possible option can be rewriting the schedule as a compensating transaction by removing the cycle

CASCADING ROLLBACK

- Even if a schedule is recoverable, to recover correctly from the failure of a transaction T_i , we may have to roll back several transactions.

- As an illustration, consider the partial schedule shown here →

- Transaction T_8 writes a value of A that is read by transaction T_9. Transaction T_9 writes a value of A that is read by transaction T_{10}.



CASCADING ROLLBACK

- Suppose that, at this point, T_8 fails. T_8 must be rolled back.
- Since T_9 is dependent on T_8, T_9 must be rolled back.
- Since T_{10} is dependent on T_9, T_{10} must be rolled back.
- This phenomenon, in which a single transaction failure leads to a series of transaction rollbacks, is called **cascading rollback**

T_8	T_9	T_{10}
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)
abort		

CASCADELESS SCHEDULES

- Cascading rollback is undesirable, since it leads to the undoing of a significant amount of work.
- It is desirable to restrict the schedules to those where cascading rollbacks cannot occur.
- Such schedules are called cascadeless schedules.
- Formally, a cascadeless 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 read operation of T_j .
- **Every cascadeless schedule is also recoverable.**

EXAMPLE

- Consider the following schedule and deduce the following:
  1. Are there are any cascading rollbacks?
  2. Is the schedule in recoverable state or not?
  3. If not, what is the cascadeless schedule for it?
- Solution:
  - Write – Read dependencies:
    - T1 → T2 (A)
    - T2 → T3 (A)
    - Check T1 → T3 for possibility of overall cycles?
      - T1 → T3 exists for A
    - T3 → T1 (B)
    - T3 → T2 (B)
  1. Cascading rollback: T1 → T2 → T3 (A)
  2. No, because cycles present:
     - T1 → T3 (A) & T3 → T1 (A)
     - T2 → T3 (B) & T3 → T2 (B)
  3. Can't deduce cascadeless schedule because it cannot be recovered due to cyclic dependencies

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

EXAMPLE

- Consider the following schedule and deduce the following:
  1. Are there are any cascading rollbacks?
  2. Is the schedule in recoverable state or not?
  3. If not, what is the cascadeless schedule for it?
- Solution:
  - Remember, a cascadeless schedule is by default recoverable schedule
  - Write – Read dependencies:
    - T1 → T2 (A)
    - T2 → T3 (A)
    - Check T1 → T3 for possibility of overall cycles?
      - T1 → T3 exists for A
    - T2 → T3 (A)
    - Check T1 → T3 for possibility of overall cycles?
      - T1 → T3 exists for A
  1. Cascading rollback: T1 → T2 → T3
  2. No cycles, so it might possibly be recoverable
     - On board
  3. Recoverable, so get cascadeless schedule
     - On board

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

CASCADELESS SCHEDULE

- Activity Sheet

CASCADELESS SCHEDULE

- Activity Sheet Solution:
  - Cascadeless Schedules Solution