Database Management System – 41 Transaction Processing (Testing for Serializability of a Schedule)

Ajay James
Asst. Prof in CSE
Government Engineering College Thrissur

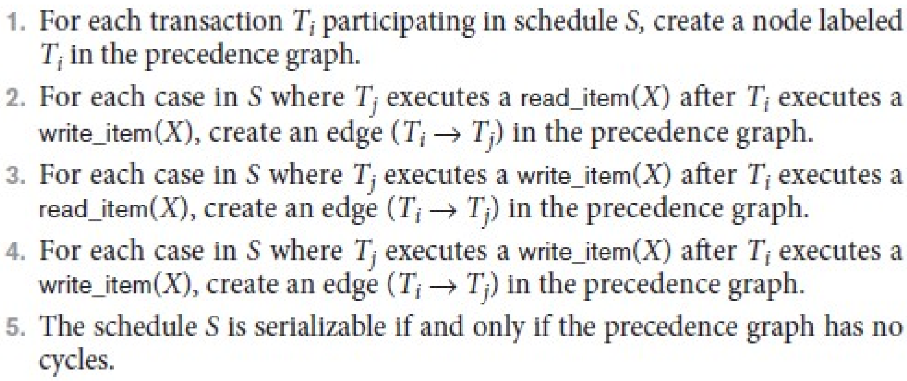
Outline

- Introduction
- Algorithm

Introduction

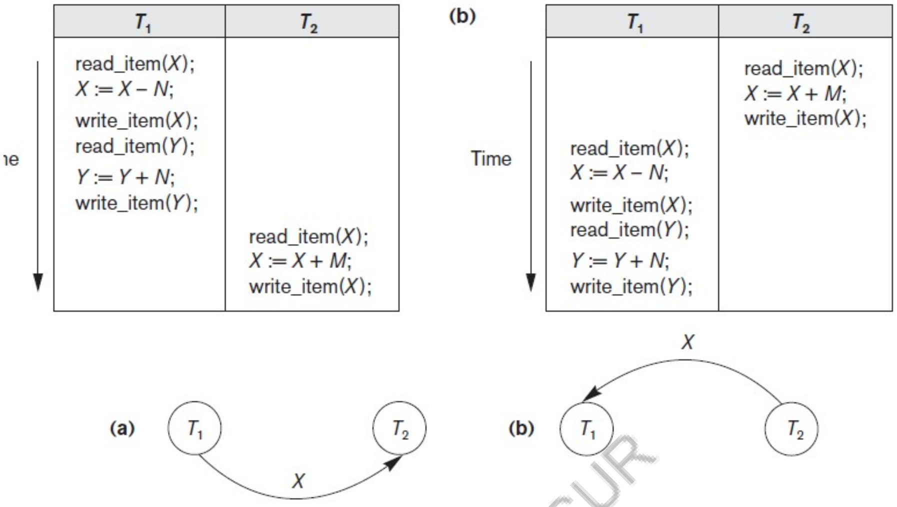
- **Precedence graph (or serialization graph)**
  - **directed graph** G = (N, E)
  - Consists of a set of nodes N = {T_1, T_2, … , T_n }
  - and a set of directed edges E = {e_1, e_2, … , e_m }
- One node in the graph for each transaction T_i in the schedule
- Each edge e_i in the graph is of the form (T_j → T_k)
  - T_j is the starting node of ei
  - T_k is the ending node of e_i.
- An edge from node T_j to node T_k is created by the algorithm if a pair of conflicting operations exist in T_j and T_k and the conflicting operation in T_j appears in the schedule before the conflicting operation in T_k.

Testing for serializability of a schedule



1. For each transaction T_i participating in schedule S, create a node labeled T_i in the precedence graph.
2. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
3. For each case in S where T_j executes a write_item(X) after T_i executes a read_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
4. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
5. The schedule S is serializable if and only if the precedence graph has no cycles.

Example



T_1	T_2

read_item(X);
X := X − N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);	

	read_item(X);
	X := X + M;
	write_item(X);

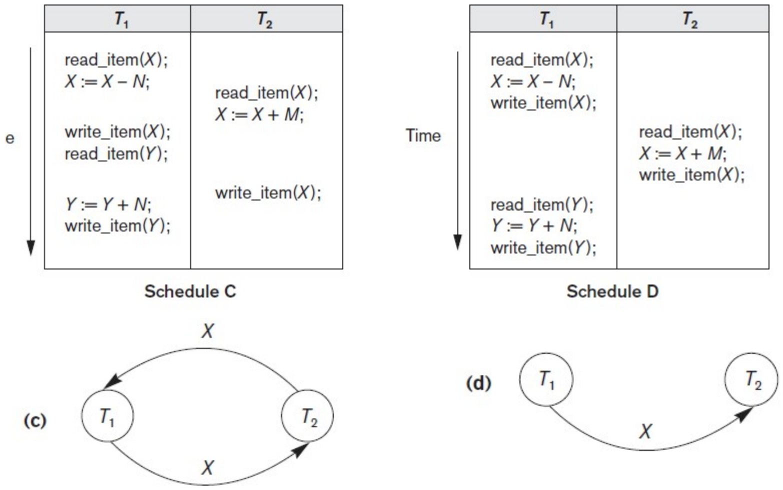
(b)

T_1	T_2

	read_item(X);
	X := X + M;
	write_item(X);

read_item(X);
X := X − N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);	

Example



T_1	T_2

read_item(X);
X := X − N;	

	read_item(X);
	X := X + M;

write_item(X);
read_item(Y);	

	write_item(X);

Y := Y + N;
write_item(Y);	

Schedule C

T_1	T_2

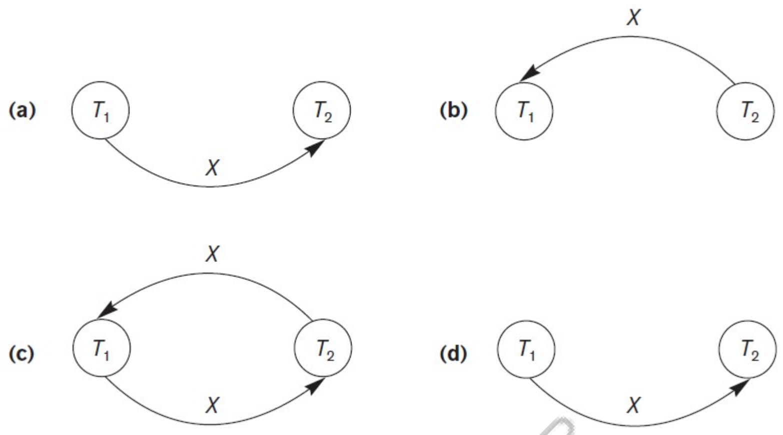
read_item(X);
X := X − N;
write_item(X);	

	read_item(X);
	X := X + M;
	write_item(X);

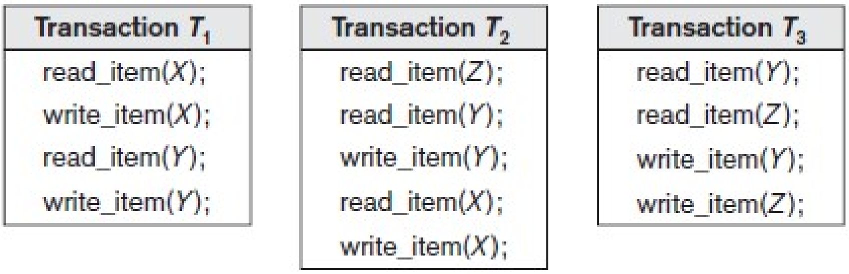
read_item(Y);
Y := Y + N;
write_item(Y);	

Schedule D

Example



Example 1

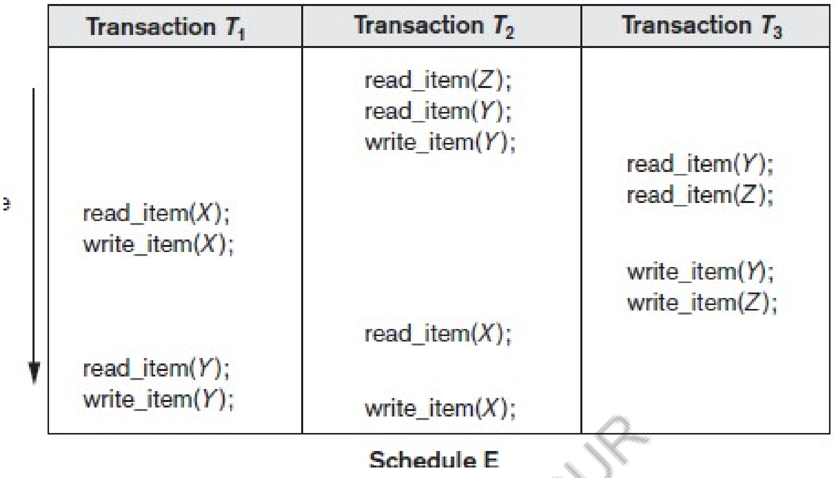


Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

Example 1



Transaction T_1	Transaction T_2	Transaction T_3

	read_item(Z);
	read_item(Y);
	write_item(Y);	

		read_item(Y);
		read_item(Z);

read_item(X);
write_item(X);		

		write_item(Y);
		write_item(Z);

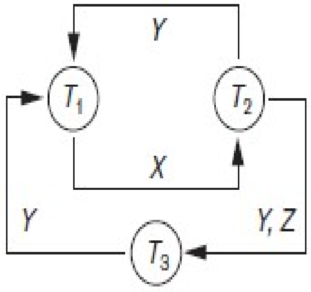
	read_item(X);	

read_item(Y);
write_item(Y);		

	write_item(X);	

Schedule E

Example1 contd.



Equivalent serial schedules

None

Reason

Cycle $X(T_1 \rightarrow T_2), Y(T_2 \rightarrow T_1)$

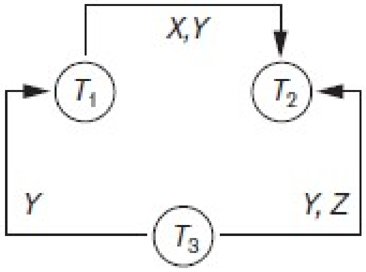
Cycle $X(T_1 \rightarrow T_2), YZ\ (T_2 \rightarrow T_3), Y(T_3 \rightarrow T_1)$

Example 2

Transaction T_1	Transaction T_2	Transaction T_3
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(Z);	
read_item(Y);		
write_item(Y);		
	read_item(Y);	
	write_item(Y);	
	read_item(X);	
	write_item(X);	

Schedule F

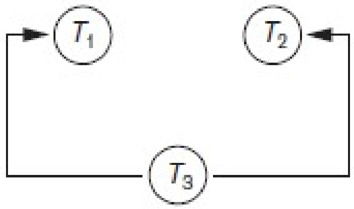
Example 2 contd..



Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

Example 3



Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

$T_3 \rightarrow T_2 \rightarrow T_1$

Reference

- Elmasri R. and S. Navathe, Database Systems: Models, Languages, Design and Application Programming, Pearson Education 6th edition and 7th edition

Thank you