Transaction Management

SCS2209 - Database II

Ms. Hiruni Kegalle

Serializability

- When several concurrent transactions are trying to access the same data item, the instructions within these concurrent transactions must be ordered in some way so as there are no problem in accessing and releasing the shared data item.
- Serializable schedule: A schedule S of n transactions is serializable if it is equivalent to some serial schedule of the same n transactions.
- Two types
  - Conflict Serializability
  - View Serializability

Result Equivalent Schedule

- If they produce the same final state of the database

Ex: X=100

S1
Read_item (X); X = X + 10; Write_item (X)

S2
Read_item (X); X = X * 1.1; Write_item (X)

- Result equivalent is not a good measure but conflict and view equivalent.

Conflict Equivalent

- The schedules S1 and S2 are said to be conflict-equivalent if the following two conditions are satisfied:
  - Both schedules S1 and S2 involve the same set of transactions (including ordering of actions within each transaction).
  - Both schedules have same set of conflicting operations. The order of any two conflicting operations is the same in both schedules.

Conflicting Actions

- Two operations in a schedule are said to conflict if they satisfy all three of the following conditions:

  (1) They belong to different transactions;

  (2) They access the same item X; and

  (3) At least one of the operations is a write_item(X).

$R_1(X);W_2(X);$

$R_1(X); R_2(X);$

$R_1(X); W_2(Y);$

Example

S1

T1	T2
R(x)	
W(x)	
	R(x)
	W(x)
R(y)	
W(y)	
	R(y)
	W(y)

S2

T1	T2
R(x)	
W(x)	
R(y)	
W(y)	
	R(x)
	W(x)
	R(y)
	W(y)

Conflict Serializable

- A conflict arises if at least one (or both) of the instructions is a write operation.
- The following rules are important in Conflict Serializability:
  1. If two instructions of the two concurrent transactions are both for read operation, then they are not in conflict, and can be allowed to take place in any order.
  2. If one of the instructions wants to perform a read operation and the other instruction wants to perform a write operation, then they are in conflict, hence their ordering is important.
     - Read;Write
     - Write;Read

Conflict Serializable

3. If both the transactions are for write operation, then they are in conflict but can be allowed to take place in any order, because the transaction do not read the value updated by each other.

➢ However, the value that persists in the data item after the schedule is over, is the one written by the instruction that performed the last write.

A schedule S is conflict serializable if it is (conflict) equivalent to some serial schedule S'.

Check for Conflict Serializability

- Check for the conflicting actions.
- Check for a cycle in the Precedence Graph.

Precedence Graph

- A simple and efficient method for determining conflict serializability of a schedule is construction of a directed graph called a precedence/conflict/ serializability graph.
- A precedence graph is a directed graph G = (N, E) that consists of a set of nodes N= {T1,T2,...,Tn } and a set of directed edges E= {e1, e2,...,em }.
- There is one node in the graph for each transaction T_i in the schedule.
- Each edge e_i in the graph is of the form (Tj→Tk),1 ≤j≤n,1 ≤k ≤ n,where Tj is the starting node of ei and Tk is the ending node of ei.

Algorithm

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.

Important

- Several serial schedules can be equivalent to S if the precedence graph for S has no cycle.

Example 1- Precedence graph

Draw precedence graph for each schedule. Two colors represent 2 transactions; T_1 in Blue and T_2 in orange.

1

READ(X)
X = X - N
WRITE(X)
READ(Y)
Y = Y + N
WRITE(Y)
READ(X)
X = X + M
WRITE(X)

2

READ(X)
X = X + M
WRITE(X)
READ(X)
X = X - N
WRITE(X)
READ(Y)
Y = Y + N
WRITE(Y)

3

READ(X)
X = X - N
READ(X)
X = X + M
WRITE(X)
READ(Y)
WRITE(X)
Y = Y + N
WRITE(Y)

4

READ(X)
X = X - N
WRITE(X)
READ(X)
X = X + M
WRITE(X)
READ(Y)
Y = Y + N
WRITE(Y)

Example 1 - Precedence graph

3

READ(X)
X = X - N
READ(X)
X = X + M
WRITE(X)
READ(Y)
WRITE(X)
Y = Y + N
WRITE(Y)

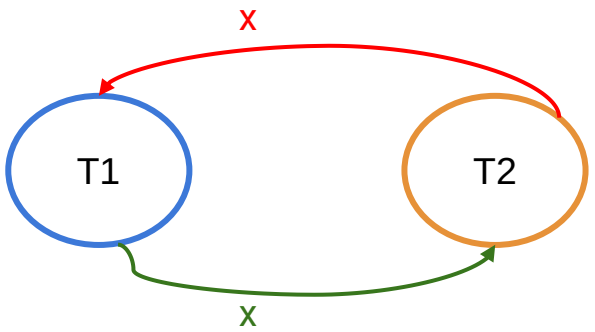
T1	T2
r(x)	
	r(x)
w(x)	
r(y)	
	w(x)
w(y)	

Example 1 - Precedence graph

3

READ(X)
X = X - N
READ(X)
X = X + M
WRITE(X)
READ(Y)
WRITE(X)
Y = Y + N
WRITE(Y)

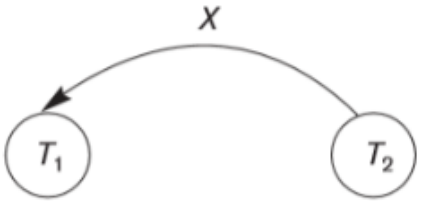
T1	T2
r(x)	
	r(x)
w(x)	
r(y)	
	w(x)
w(y)	



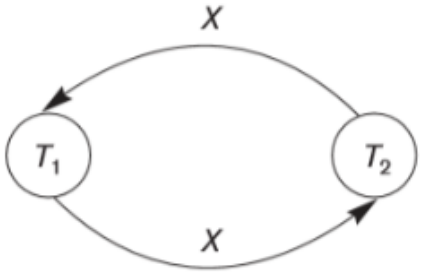
Example 1 - importance of properties

1

2



3



4

Example 2 - Precedence graph

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 2 - Precedence graph

Draw the serializable graph for the schedule E. Write the equivalent serial schedules if exist.

Time

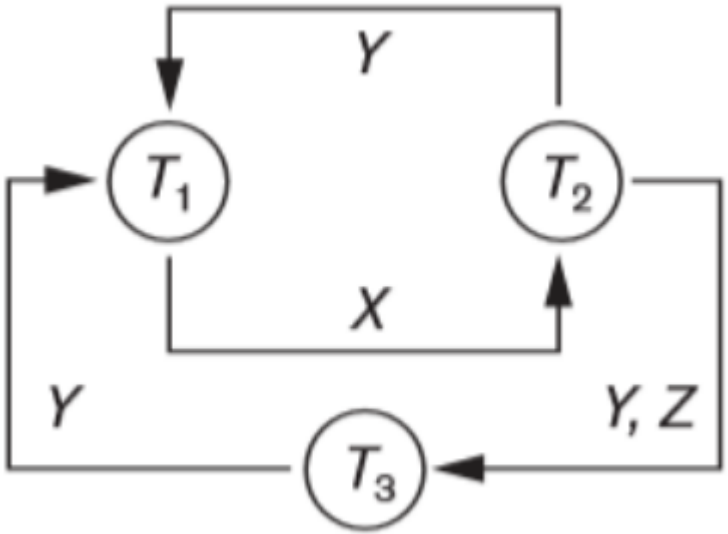
Transaction T_1	Transaction T_2	Transaction T_3

	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

Example 2 - Precedence graph

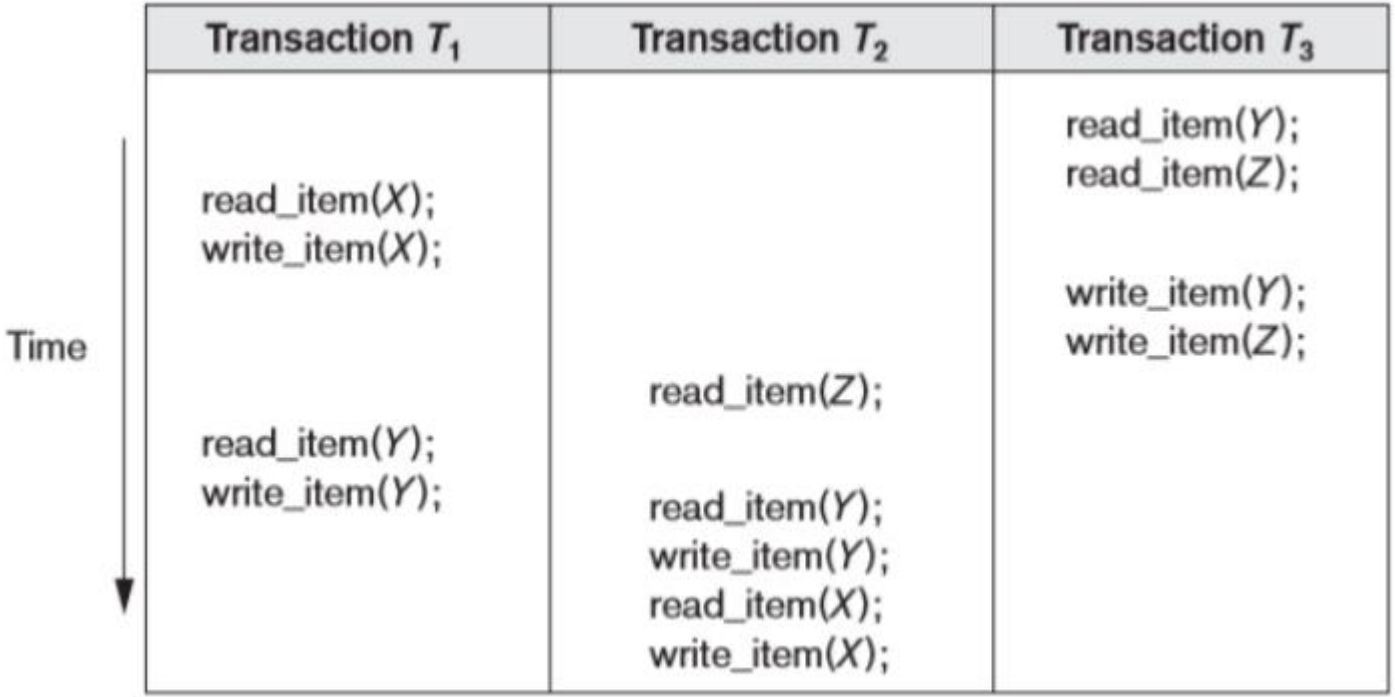


2 cycles

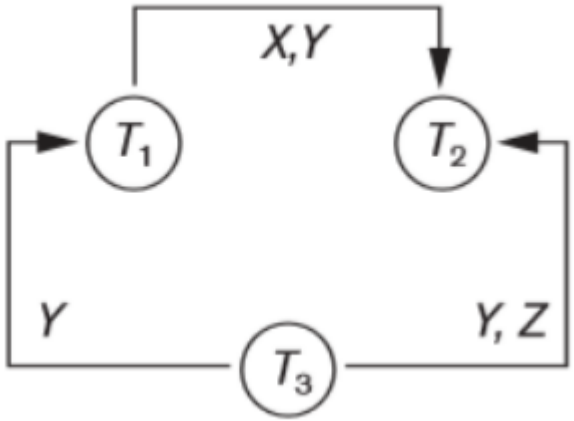
- Cycle X(T1 T2),Y(T2 T1)
- Cycle X(T1 T2),YZ (T2 T3),Y(T3 T1)

No Equivalent serial schedules.

Example 2 - Precedence graph



Example 2 - Precedence graph



No cycles

Equivalent serial schedule is **T3 -> T1 ->T2**

Practical Usage

- Usage of precedence graph is not practical in real DB
  - Predicting the schedule is hard
- Protocols used to enforce serializable schedules: 2PL and time stamping

View Serializability

- Two schedules are view serializable, If the following rules are followed while creating the second schedule out of the first.
- Transactions T1 and T2 are being serialized to create two different schedules. S1 and S2 which we want to be View Equivalent and both T1 and T2 wants to access the same data item.

View Serializability

1. If in S1, T1 reads the initial value of the data item, then in S2 also, T1 should read the initial value of that same data item.

1. If in S1, T1 writes a value in the data item which is read by T2, then in S2 also, T1 should write the value in the data item before T2 reads it.

1. If in S1, T1 performs the final write operation on that data item, then in S2 also, T1 should perform the final write operation on that data item.

Summary

Two schedules to become view serializable , the following 3 conditions should be satisfied.

1. Initial Reads
2. W-R Conflicts
3. Final Writes

A schedule S is said to be view serializable if it is view equivalent to a serial schedule.

Example 1- View Serializability

T1	T2	T3
		R(x)
	R(x)	
		W(x)
R(x)		
W(x)		

T2	T3	T1
R(x)		
	R(x)	
	W(x)	
		R(x)
		W(x)

Example 2- View Serializability

S1

T1	T2	T3
	R(b)	
	W(a)	
R(a)		
		R(a)
W(b)		
	W(b)	
		W(b)

S2

T1	T2	T3
	R(b)	
	W(a)	
	W(b)	
R(a)		
W(b)		
		R(a)
		W(b)

Question

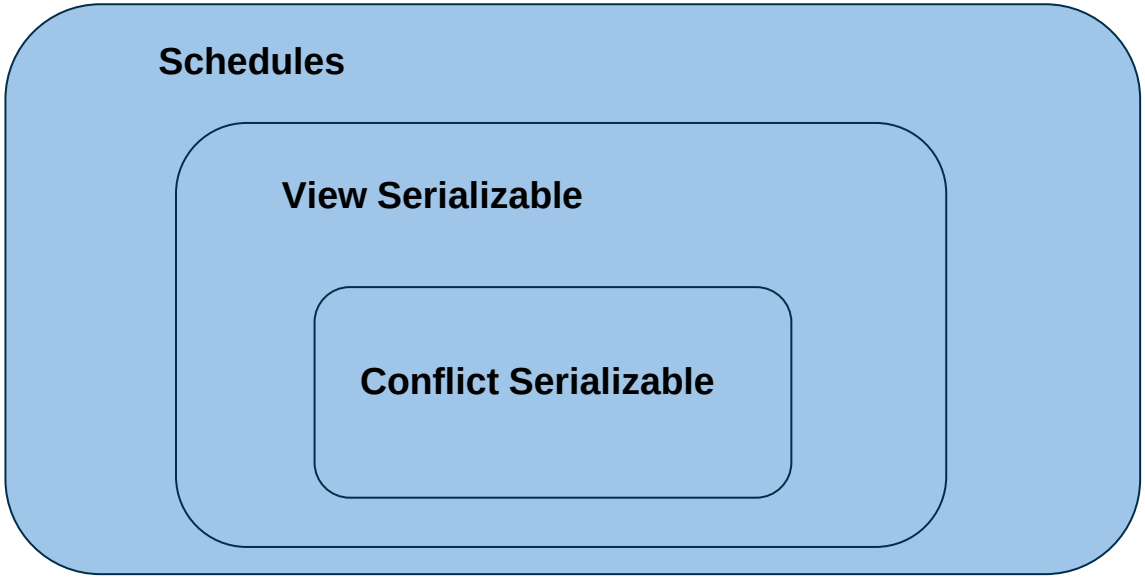
S: R1(X), W2(X), W1(X), W3(X), C1, C2, C3

Is S View serializable?

Is S Conflict serializable?

Summary

Any schedule that is conflict-serializable is also view-serializable, but not necessarily the opposite.



Assignment 01

Will be uploaded into VLE : Date?
Lecture: Next week?

Thank you...