COMP207
Database Development

Tutorial 1c

Precedence Graphs

Precedence graphs

Precedence Graph

time	T1	T2	T3
1			
2			
3			
4			

1. Scheduler produces schedule Sa:

Sa: r1(X);w2(X);w1(X);w3(X);

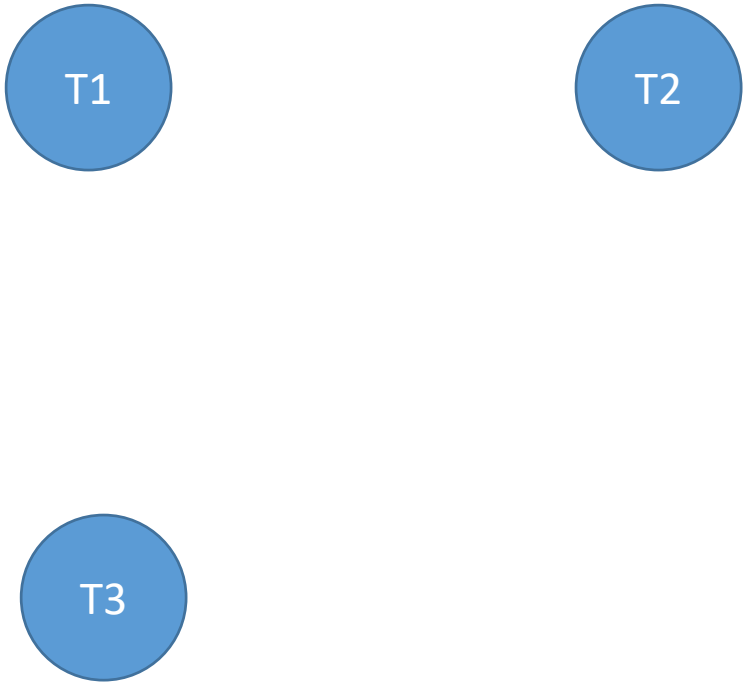
2. Set up a table

Precedence Graph

time	T1	T2	T3
1			
2			
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

3. Node for each transaction

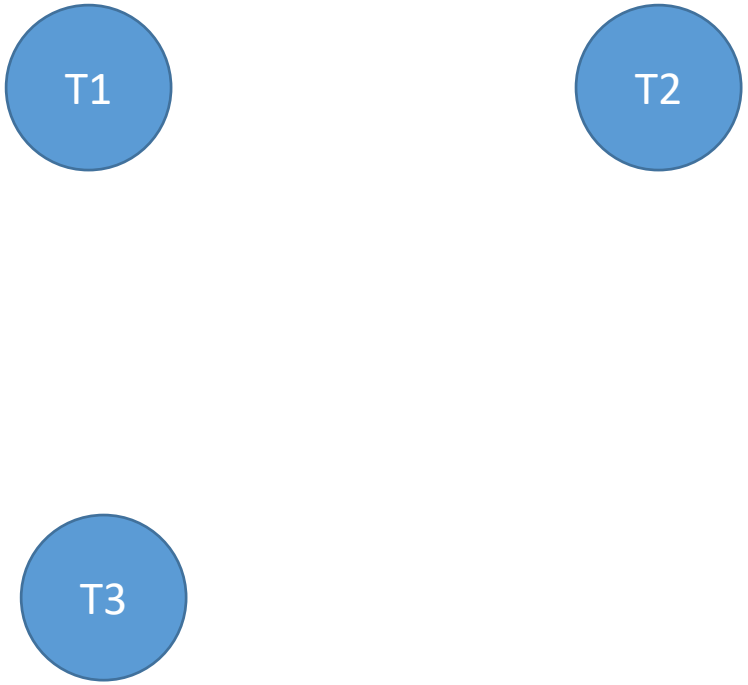


Precedence Graph

time	T1	T2	T3
1	r(X)		
2			
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

4. Process Sa:

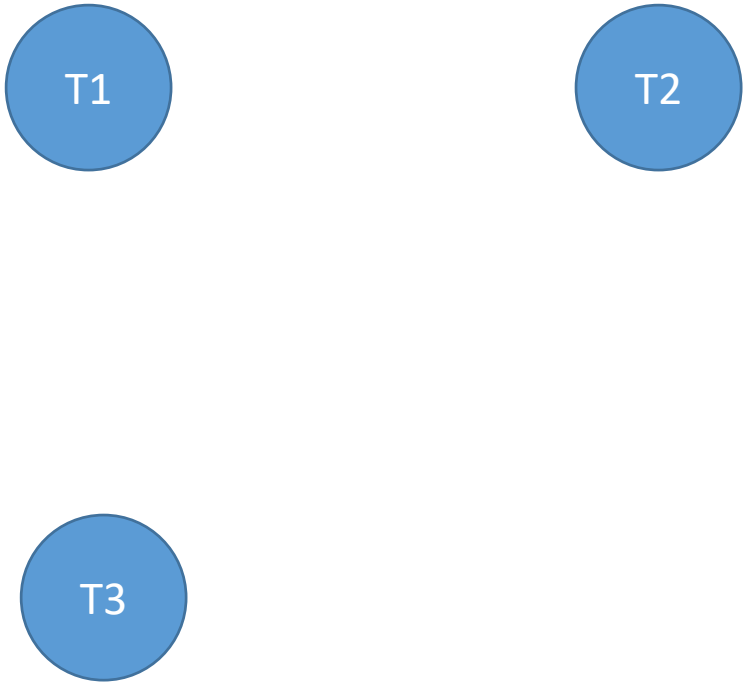


Read in the first operation of the schedule. No action can be taken yet – there is nothing to compare the operation to

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);



Read in the next operation of Sa

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

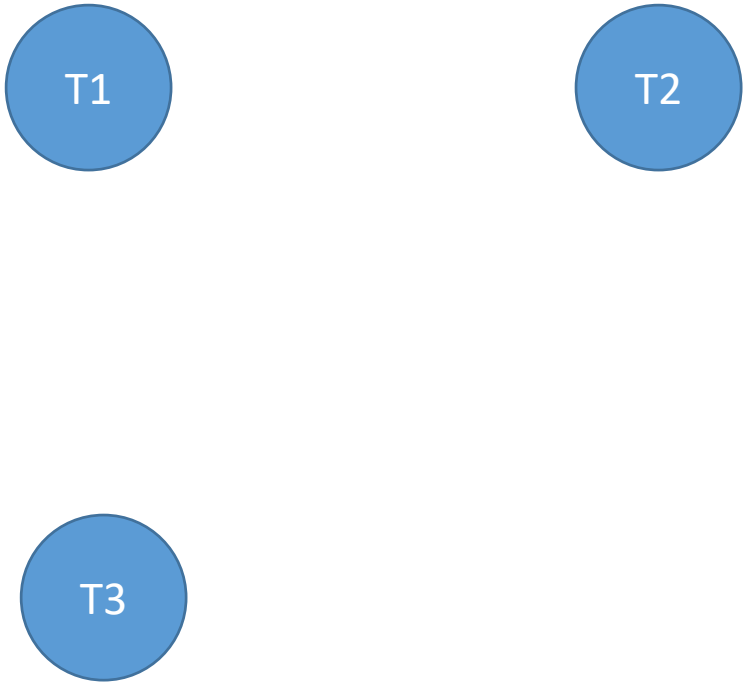
Begin to test for serialisability by looking for conflicting operations

Q1: Same item?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

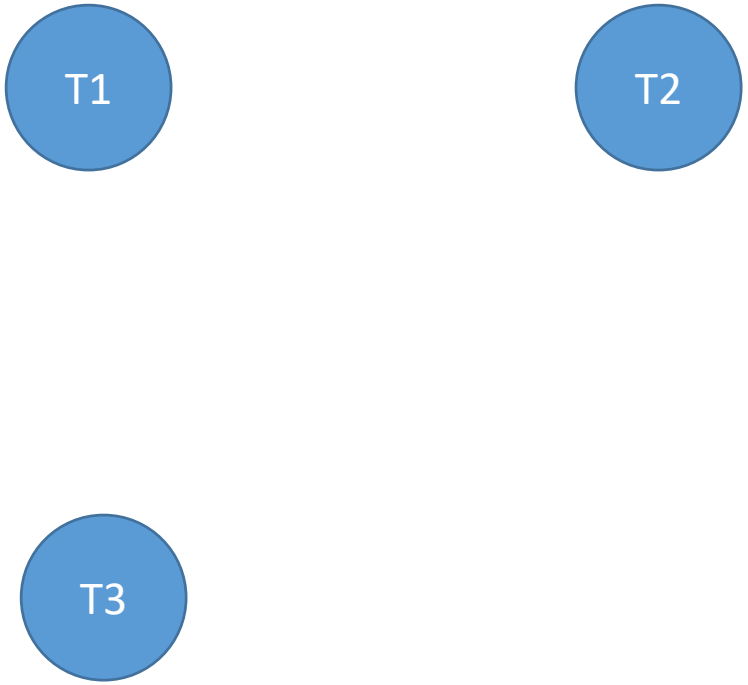


Q1: Same item? YES - both use X

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);



Q1: Same item? YES
Q2: Different Transactions?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

Q1: Same item? YES
Q2: Different Transactions? YES (T1 and T2)

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);

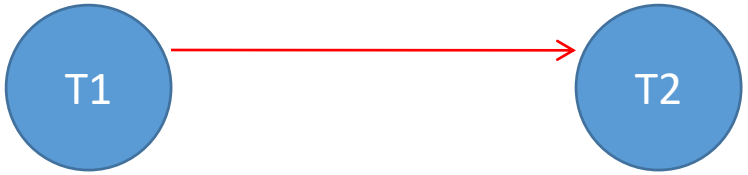
Same item? YES
Different Transactions? YES
Is one of the operations a write? YES

These are conflicting operations:
Produce an edge (arrow) from the earlier operation to the later one

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3			
4			

Sa: r1(X);w2(X);w1(X);w3(X);



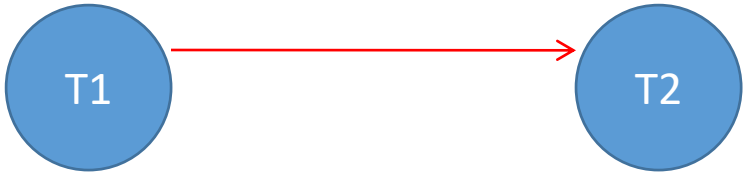
Same item? YES
Different Transactions? YES
Is one of the operations a write? YES

These are conflicting operations
Create an edge on the graph

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);

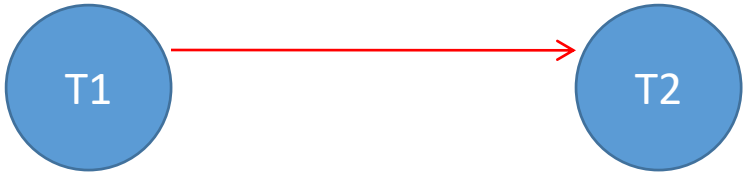


Q1: Same item?
Q2: Different Transactions?
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);

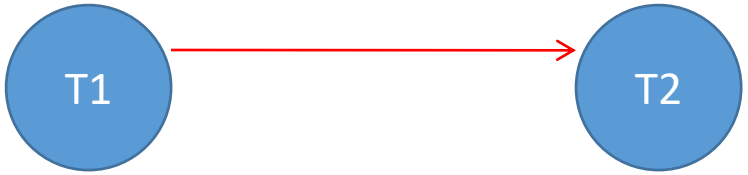


Q1: Same item? YES
Q2: Different Transactions?
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);

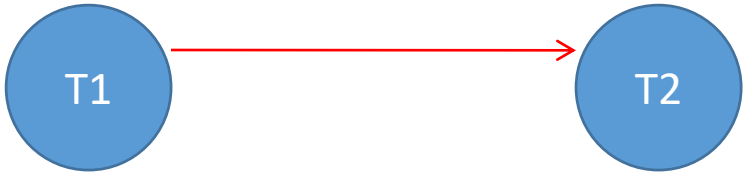


Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);



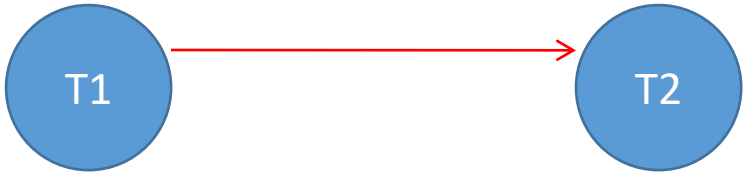
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);



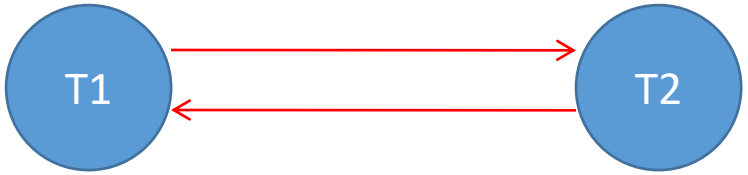
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

These are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			

Sa: r1(X);w2(X);w1(X);w3(X);



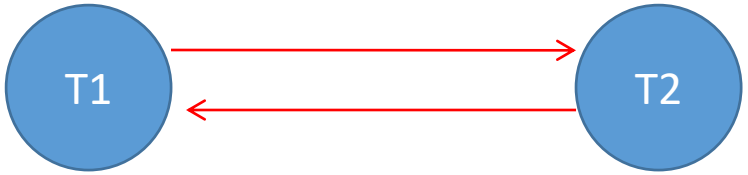
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

These are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);

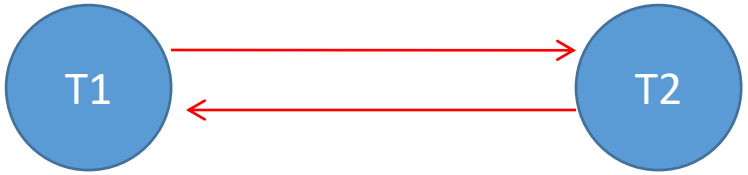


Q1: Same item?
Q2: Different Transactions?
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);

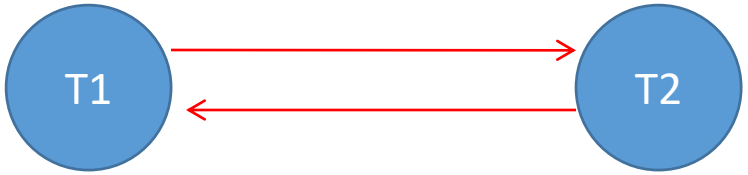


Q1: Same item? YES
Q2: Different Transactions?
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);

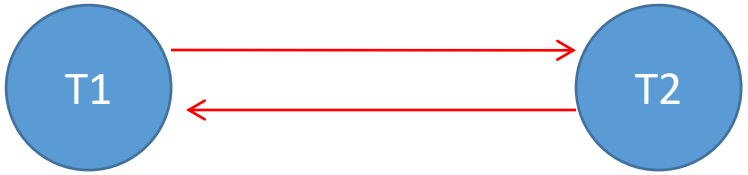


Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write?

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);

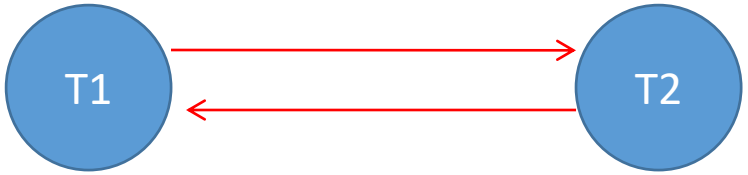


Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



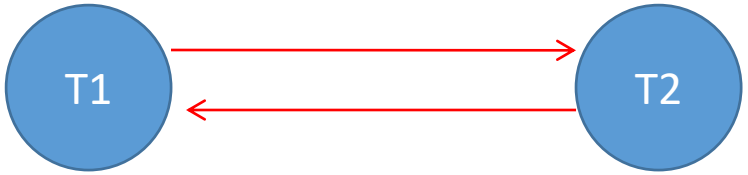
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



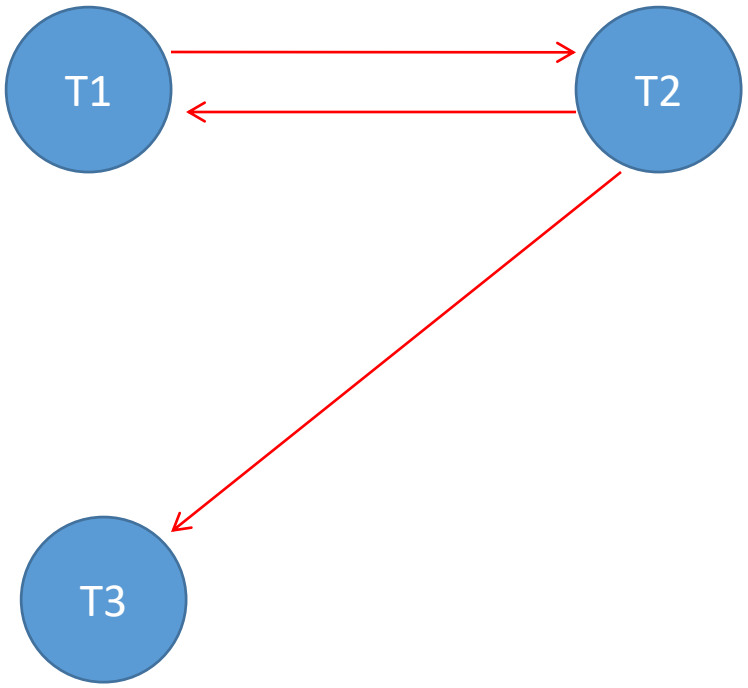
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

These are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



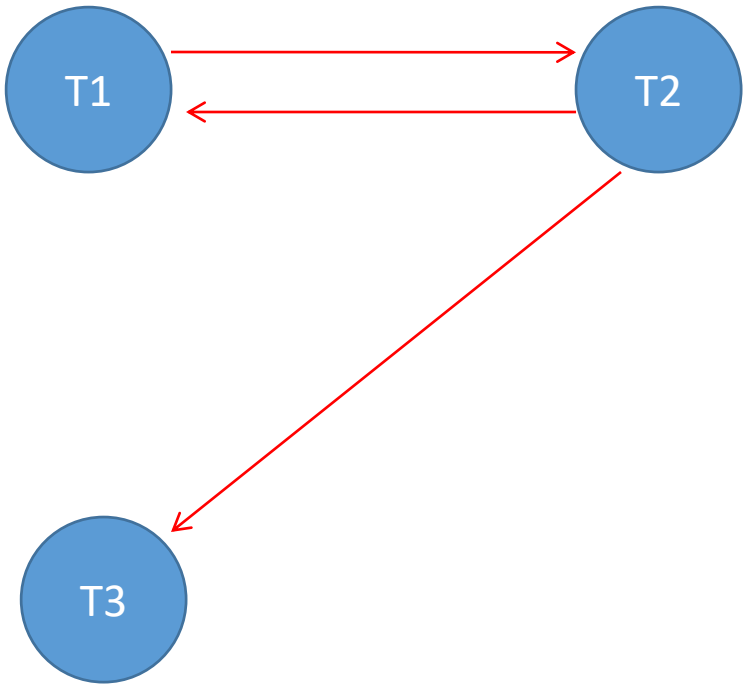
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

These are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



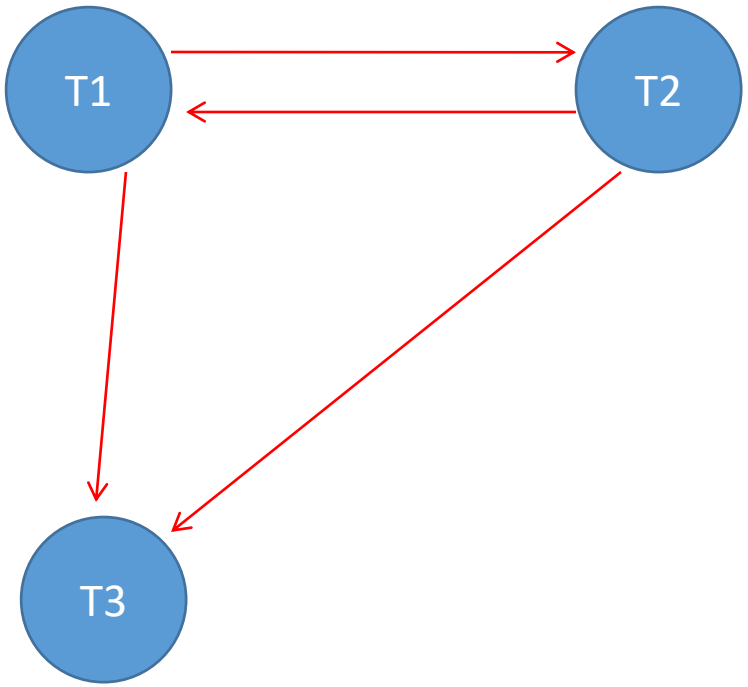
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



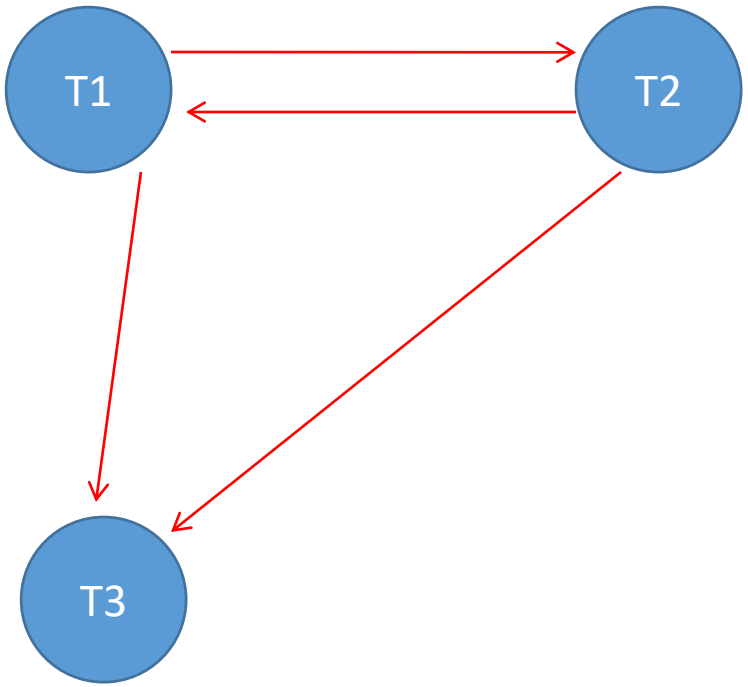
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



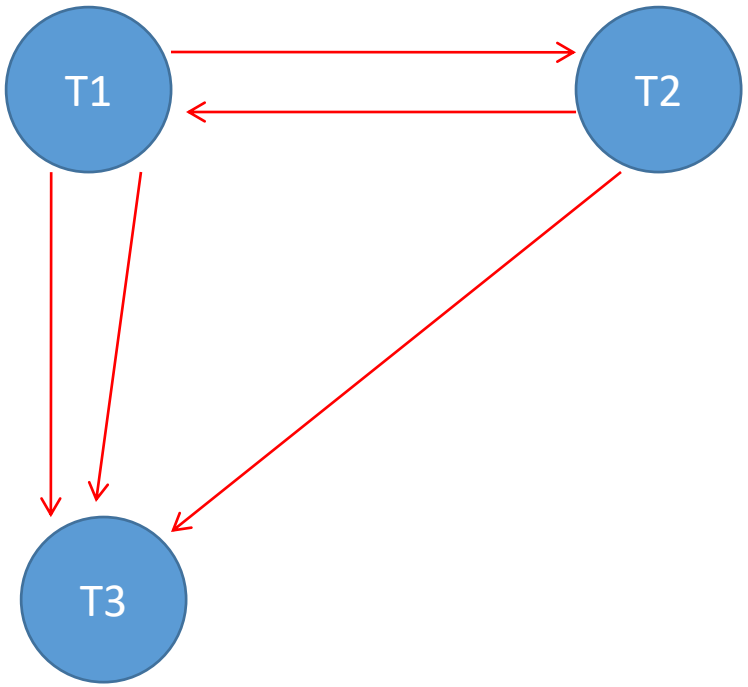
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



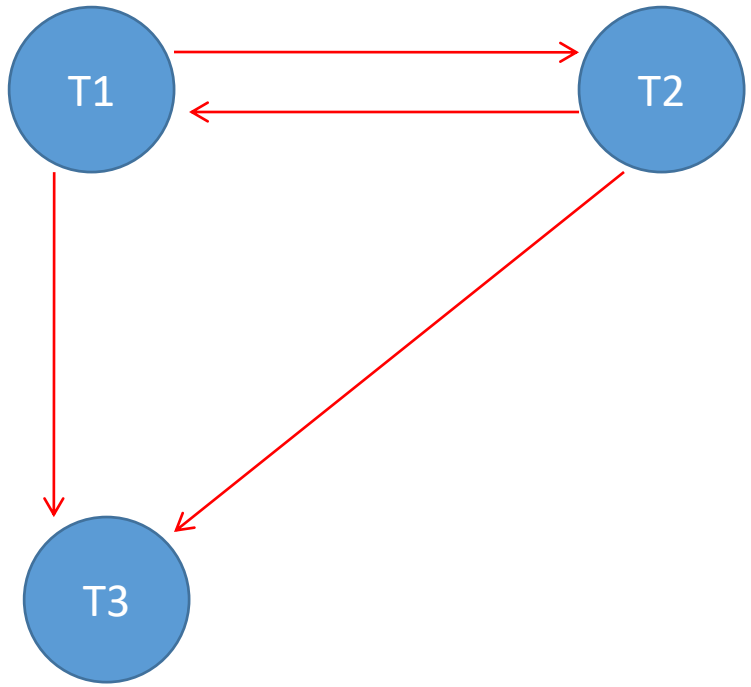
Q1: Same item? YES
Q2: Different Transactions? YES
Q3: Is one of the operations a write? YES

Three YES means these are conflicting operations
Create an edge

Precedence Graph

time	T1	T2	T3
1	r(X)		
2		w(X)	
3	w(X)		
4			w(X)

Sa: r1(X);w2(X);w1(X);w3(X);



If needed, duplicate edges can be reduced to one only for clarity as in T1 to T3.
We cannot reduce the edges between T1 and T2 as they are different edges – note the direction

Precedence Graph Analysis

- Schedule Sa produced by the scheduler is obviously NOT a serial schedule as different transactions have been interleaved with each other – hence it is a concurrent schedule
- This means there may be conflict between operations in the schedule that could give ‘wrong’ results
- However, there are combinations of conflicting operations that will still give the ‘correct’ result, i.e. they will work just as if they had been executed serially (so Sa is ‘Conflict Serialisable’)

Precedence Graph Analysis

- To see if Sa is conflict serialisable, produce a Precedence Graph and analyse the output
- For Sa there is a cycle in the graph
  - T1➔T2 and T2➔T1
  - So the schedule is not conflict serialisable
  - It may give the ‘wrong’ result
- If NO cycle is present
  - The graph is ‘acyclic’
    - The schedule is conflict serialisable (to a serial schedule)

Precedence Graph Analysis

- If a schedule is conflict serialisable, we are saying it is equivalent to executing the transactions in it as if they were done in a serial manner
- But what serial schedule is this equivalent to?
  - If we have a schedule with three transactions in it (T1, T2 and T3) then is the conflict equivalent to running T1-T2-T3 or T2-T3-T1 or T1-T3-T2 etc?

View-Equivalence - Complexity

Algorithms are available to test for a schedule being view-serializable or not.

Algorithmic Complexity is characterised by ‘Computational Complexity’ – the total amount of elementary computation required by the algorithm as a function of the size of the data set to which the algorithm applies.

Many algorithms have Polynomial Complexity – complexity expressed as a polynomial function of data set size – ‘tractable problem’

View-Equivalence - Complexity

- The problem of view-serialisability is NP-hard
  - **N**on-deterministic **P**olynomial-time hard
    - Polynomial time algorithms are reasonable to compute
    - But... The time to run the algorithm grows too fast to compute exact solutions in all cases
  - Unlikely to find an efficient polynomial time algorithm
  - NP-Complete problems have no known solution algorithm with a polynomial complexity
- Linear Complexity is a special case of Polynomial Complexity. NP-Complete problems are a class of problems that are ‘intractable problems’

View Equivalence

- Less stringent definition of schedule equivalence
- Schedules consisting of the same operations are view equivalent when:
  - 1. In Sa, if T1 reads initial value of item then in Sb, T1 must also read the initial value of the same item
  - 2. In Sa, if T1 reads an item written by T2, then in Sb, T1 must also read the same item written by T2
  - 3. In Sa, if T1 performs the final write on an item, then in Sb, T1 must also perform the final write on the same item

View Equivalence

- A schedule is view serialisable when it is view equivalent to a serial schedule
- Every conflict serialisable schedule is view serialisable
- A view serialisable schedule may not be conflict serialisable
  - These break the constrained write rule
    - Can’t write an item unless it’s been read first
      - “Blind Write”
    - There is an unconstrained write assumption that does allow blind writes

View Equivalence

- Creates all possible combinations of the transactions
- Given: Sx = r1(X); r2(Y);r2(Y); w2(X); w3(Y); r1(X);
  - Sa = T1 T2 T3
  - Sb = T1 T3 T2
  - Sc = T2 T1 T3
  - Sd = T2 T3 T1
  - Se = T3 T1 T2
  - Sf = T3 T2 T1

View Equivalence

- Given: Sx = r1(x); r2(Y);r2(Y); w2(X); w3(Y); r1(X);
- A schedule is view serialisable if:
  - 1. A transaction that reads the initial data in a schedule is the same transaction that reads the initial data in another schedule
    - T1 reads data before T2 – so TI comes before T2
      - Remove Sc, Sd and Sf

View Equivalence

- Given: r1(x); r2(Y);r2(Y); w2(X); w3(Y); r1(X);
- A schedule is view serialisable if:
  - 2. A transaction that reads an item after another transaction writes it in a schedule is the same transaction that reads an item after another transaction writes it in another schedule
    - T1 reads data after T2 writes – so T2 comes before T1
      - Remove Sa, Sb, Se
    - But ... In test1, we show T1 must come before T2
    - This produces a cycle so the schedule Sx is NOT view serialisable

View Equivalence

- Given: Sy = w3(Z); r2(X); w2(Y); r1(Z); w3(Y); w1(Y);
  - Note this is a ‘blind write’ example
- A schedule is view serialisable if:
  - 3. A transaction that writes the final value for an item in a schedule is the same transaction that writes the final value for that item in another schedule
    - T1 writes the final value – so T1 comes after T2 and T3
      - If it doesn’t, then T2 or T3 will overwrite the value of Y
    - Remove Sa, Sb, Sc and Se
    - Sd = T2, T3, T1 or Sf = T3, T2, T1 are **view equivalent** schedules