CS 186 Discussion #8

Transactions, Concurrency Control

Logistics

- Homework 4 due 11/04
- Midterm 2 on 11/09
  - Up to concurrency control (lock granularity)
- Midterm 3 partially cumulative

System R…

```
SELECT *
FROM R, S, T
WHERE R.a = S.a
AND S.b = T.b;
```

These are the two-table join costs:
1) R join S = 6,000
2) S join R = 2,000
3) R join T = 5,000
4) T join R = 1,000
5) S join T = 4,000
6) T join S = 3,000

We now add the third table and have the following join costs:
1) (R join S) join T = 10,000
2) T join (R join S) = 6,000
3) (S join R) join T = 15,000
4) T join (S join R) = 11,000
5) (R join T) join S = 10,000
6) S join (R join T) = 7,000
7) (T join R) join S = 14,000
8) S join (T join R) = 16,000
9) (S join T) join R = 13,000
10) R join (S join T) = 12,000
11) (T join S) join R = 20,000
12) R join (T join S) = 9,000

Which of these will the optimizer select as your final query plan? *

Which of the following two-table join plans will be selected by a System R-style query optimizer? *

System R…

```
SELECT *
FROM R, S, T
WHERE R.a = S.a
AND S.b = T.b;
```

We now add the third table and have the following join costs:

1) (R join S) join T = 10,000
2) T join (R join S) = 6,000
3) (S join R) join T = 15,000 ←
4) T join (S join R) = 11,000
5) (R join T) join S = 10,000
6) S join (R join T) = 7,000
7) (T join R) join S = 14,000
8) S join (T join R) = 16,000
9) (S join T) join R = 13,000
10) R join (S join T) = 12,000
11) (T join S) join R = 20,000 ←
12) R join (T join S) = 9,000

These are the two-table join costs:

1) R join S = 6,000
2) S join R = 2,000 ←
3) R join T = 5,000
4) T join R = 1,000
5) S join T = 4,000
6) T join S = 3,000 ←

Which of these will the optimizer select as your final query plan? *

Which of the following two-table join plans will be selected by a System R-style query optimizer? *

Transactions

- **A**tomicity: All actions in the xact happen, or none
  - Logging
- **C**onsistency: xact will not break DB consistency
  - Integrity constraints
- **I**solation: Execution of the xact is isolated from other xacts
  - Serial ordering
- **D**urability: Committed xacts have persistent effects
  - Logging

Transactions

- **A**tomicity: All actions in the xact happen, or none
  - Logging
- **C**onsistency: xact will not break DB consistency
  - Integrity constraints
- **I**solation: Execution of the xact is isolated from other xacts
  - Serial ordering
- **D**urability: Committed xacts have persistent effects
  - Logging

Transactions

- **A**tomicity: All actions in the xact happen, or none
  - Logging
- **C**onsistency: xact will not break DB consistency
  - Integrity constraints
- **I**solation: Execution of the xact is isolated from other xacts
  - Serial ordering
- **D**urability: Committed xacts have persistent effects
  - Logging

Serializability

- <u>Serial Schedule</u> - no intermittent transactions

R(A) W(A) R(B) W(B)

R(A) W(A) R(B) W(B)

R(A) W(A) R(B) W(B)

R(A) W(A) R(B) W(B)

Serializability

- <u>Serializable</u>: equivalent to any serial schedule
  - <u>Equivalent</u>: same xacts/actions and final state

R(A) W(A)

R(B) W(B)

R(A) W(A)

R(B) W(B)

Equivalent?

Serializable?

Serializability

- <u>Serializable</u>: equivalent to any serial schedule
  - <u>Equivalent</u>: same xacts/actions and final state

R(A) W(A)

R(B) W(B)

R(A) W(A)

R(B) W(B)

Equivalent? ✓

Serializable? ✓

Conflict Serializability

- What is a <u>conflict</u>?
  - Different xacts on one object that is *written to*

T1:	R(A)	R(B)		W(A)
T2:			R(B) W(B)	

Conflict Serializability

- What is a <u>conflict</u>?
  - Different xacts on one object that is *written to*

T1: R(A) R(B) W(A)

T2: R(B) W(B)

Conflict Serializability

T1: R(A) R(B) W(A)

T2: R(B) W(B)

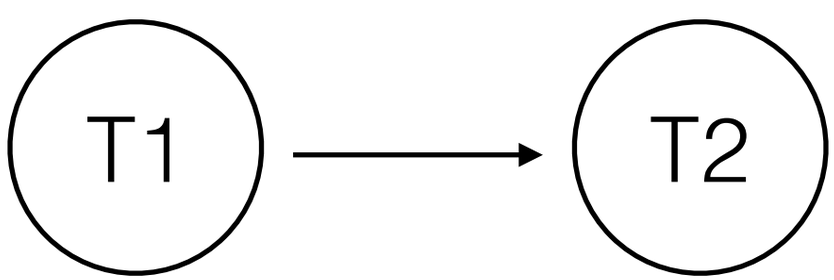
- <u>Dependency Graph</u>:
  - One node per xact
  - Edge from Ti to Tj if some operation O_i is earlier than and conflicts with O_j

Conflict Serializability

T1: R(A) R(B) W(A)

T2: R(B) W(B)

- <u>Dependency Graph</u>:
  - One node per xact
  - Edge from Ti to Tj if some operation O_i is earlier than and conflicts with O_j



Conflict Serializability

- Conflict Serializable: conflict equivalent to some serial schedule
  - Conflict Equivalent: same xacts/actions and *every conflict pair is ordered the same way*
  - Serializable vs. Conflict Serializable?

Conflict Serializability

- Conflict Serializable: conflict equivalent to some serial schedule
  - Conflict Equivalent: same xacts/actions and *every conflict pair is ordered the same way*
  - Serializable vs. Conflict Serializable?

- Theorem: Schedule is conflict serializable *if and only if* dependency graph is acyclic

Worksheet - 2a

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

- Dependency Graph
  - Conflicting writes? Edges?

Worksheet - 2a

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

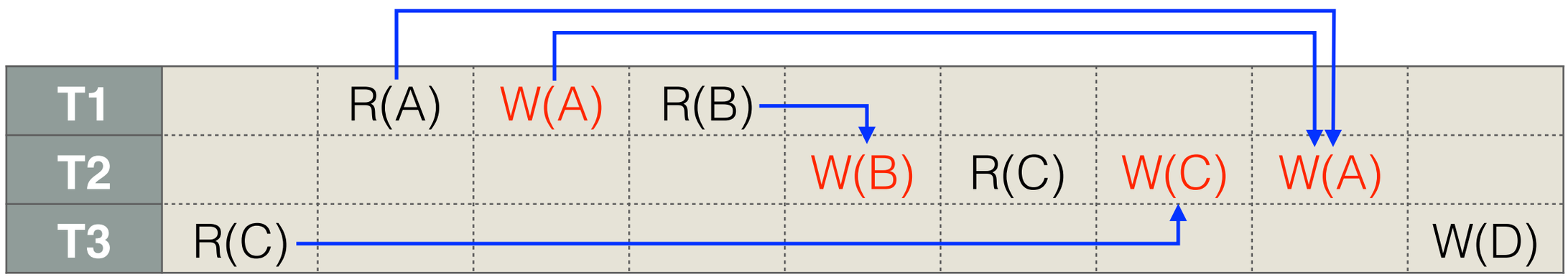
- Dependency Graph
  - Conflicting writes? Edges?

Worksheet - 2a

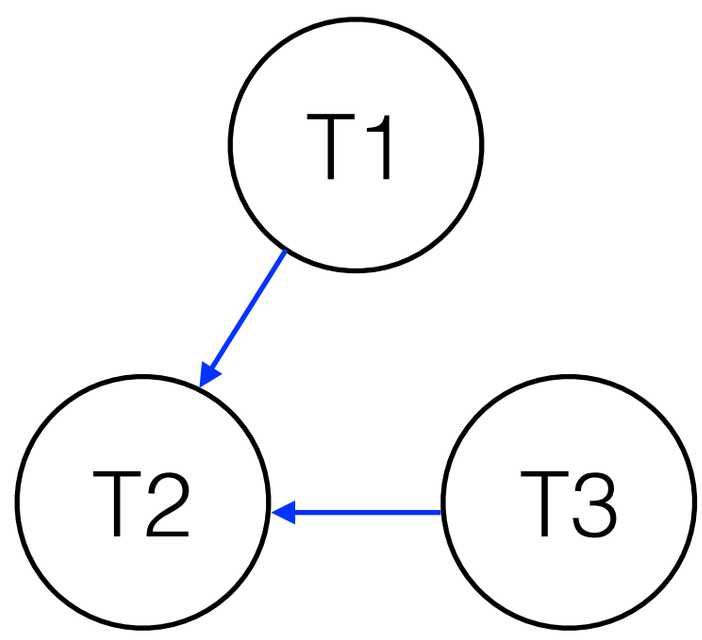
T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

- Dependency Graph
  - Conflicting writes? Edges?

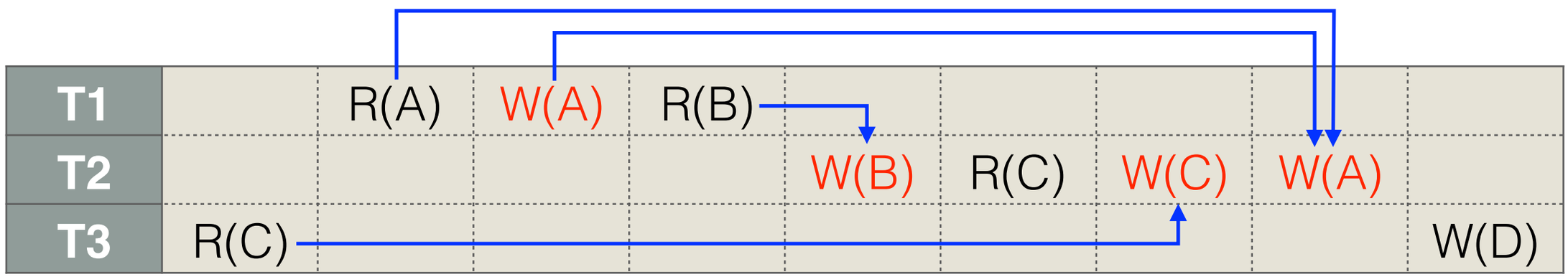
Worksheet - 2a



- Dependency Graph
  - Conflicting writes? Edges?



Worksheet - 2b



- Conflict Serializability
  - Dependency Graph?
  - What actions would we move?

Worksheet - 2b

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

- Conflict Serializability
  - Dependency Graph? Acyclic!
  - What actions would we move?

Worksheet - 2b

T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

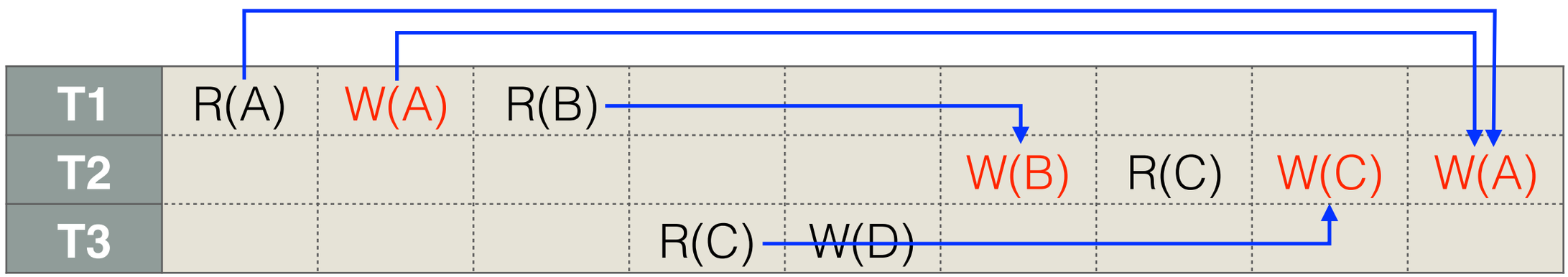
- Conflict Serializability
  - Dependency Graph? Acyclic!
  - What actions would we move? T3

Worksheet - 2b

T1			R(A)	W(A)	R(B)				
T2						W(B)	R(C)	W(C)	W(A)
T3	R(C)	W(D)							

- Conflict Serializability
  - Dependency Graph? Acyclic!
  - What actions would we move? T3

Worksheet - 2b



- Conflict Serializability
  - Dependency Graph? Acyclic!
  - What actions would we move? T3

Worksheet - 2c

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

Worksheet - 2c

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

Worksheet - 2c

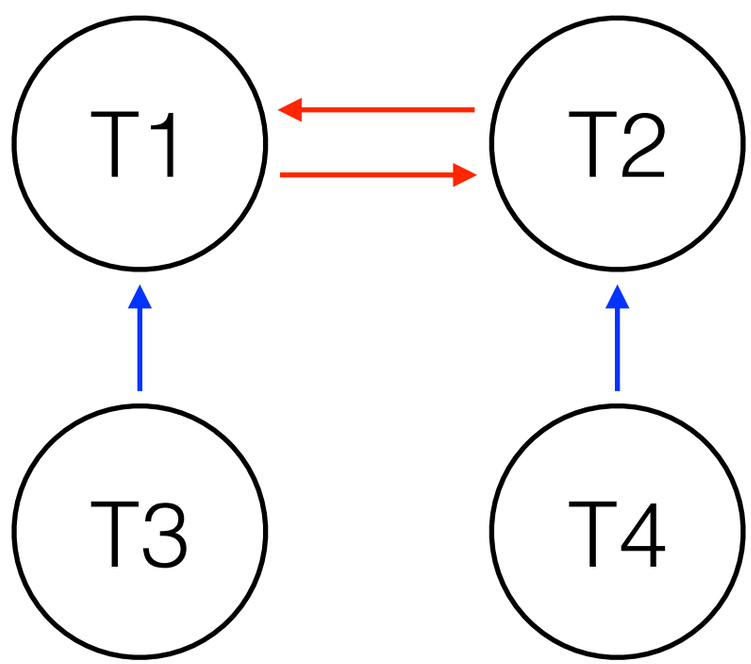
T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

- Is this conflict serializable?

Worksheet - 2c

T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

- Is this conflict serializable?



2PL

- S = shared lock
  - reads
- X = exclusive lock
  - writes (and reads)

	S	X
S	*	
X		

- For each xact: Once a lock has been released, cannot acquire any more

What does 2PL guarantee?

- Conflict Serializability?
- Cascading aborts?

What does 2PL guarantee?

- Conflict Serializability? ✓
- Cascading aborts? ✗

Strict 2PL

- Like 2PL, but addresses cascading aborts
  - Release locks <u>only when a xact completes</u>
    - Completion = commit, or abort + rollback

Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
B = B*10	Read(F)
Write(B)	Unlock(F)
Lock_X(F)	Lock_S(B)
F = B*100	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- Which locks have to wait?
- Is this conflict serializable?

Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
B = B*10	Read(F)
Write(B)	Unlock(F)
Lock_X(F)	Lock_S(B)
F = B*100	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- Which locks have to wait?
- Is this conflict serializable?

Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
B = B*10	Read(F)
Write(B)	Unlock(F)
Lock_X(F)	Lock_S(B)
F = B*100	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- 2PL?
- Strict 2PL?
- No deadlock?

Worksheet - 3

T1	T2
Lock_X(B)	
Read(B)	Lock_S(F)
B = B*10	Read(F)
Write(B)	Unlock(F)
Lock_X(F)	Lock_S(B)
F = B*100	
Write(F)	
Unlock(F)	
Unlock(B)	
	Read(B)
	Print(F+B)
	Unlock(B)

- 2PL? ✗
- Strict 2PL? ✗
- No deadlock? ✓

- There were some slides here that were incorrect. See “Discussion 8 Corrections” for more.

Deadlock Detection

- "Waits-for" Graph:
  - edge from Ti to Tj if *i* is waiting for *j*

- Deadlock if cycle in graph

Deadlock Detection

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

- What needs to wait?

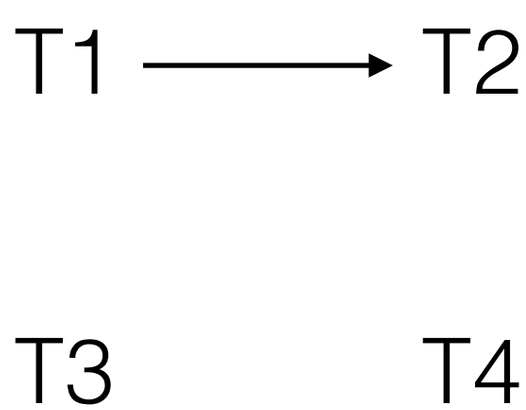
T1 T2

T3 T4

Deadlock Detection

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

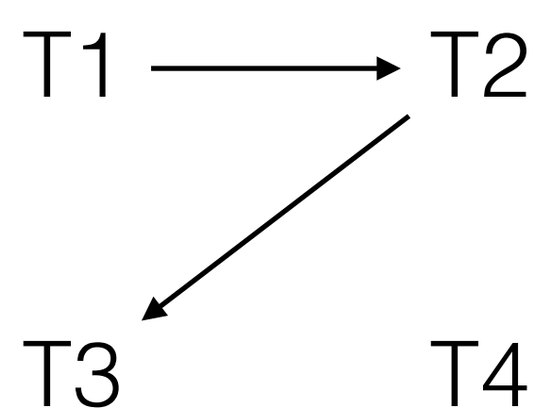
- What needs to wait?



Deadlock Detection

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

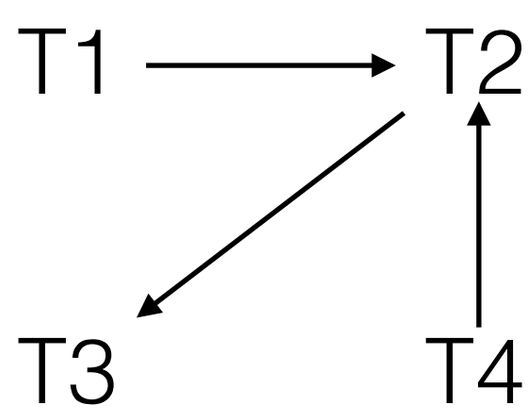
- What needs to wait?



Deadlock Detection

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

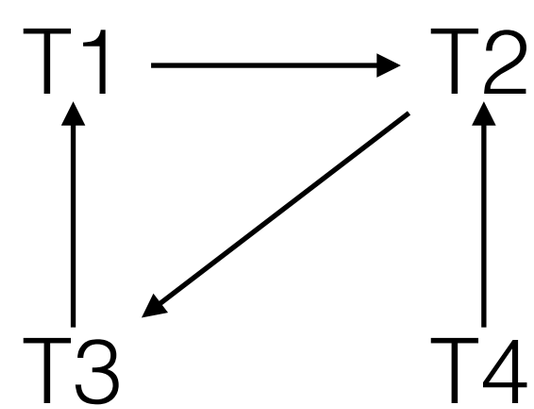
- What needs to wait?



Deadlock Detection

T1	S(A)	S(D)		S(B)					
T2			X(B)				X(C)		
T3					S(D)	S(C)			X(A)
T4								X(B)	

- What needs to wait?



Deadlock Avoidance

- Priorities based on timestamp
- Wait-Die: if Ti higher, wait for Tj, else *abort Ti*
- Wound-Wait: if Ti higher, *Tj aborts*, else wait for Tj