CS 186 Discussion 7

Transactions, Concurrency Control
Locking Granularity

Logistics

- Nothing new…
  - Have you checked Homework 3 pushes?
  - Homework 4 due this Friday
  - Midterm 1 regrade requests due this Friday
  - Midterm 2 Thursday after break
    - Review session (probably?) Tu/Wed week of
  - Fill out Mid-Semester Feedback

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

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)

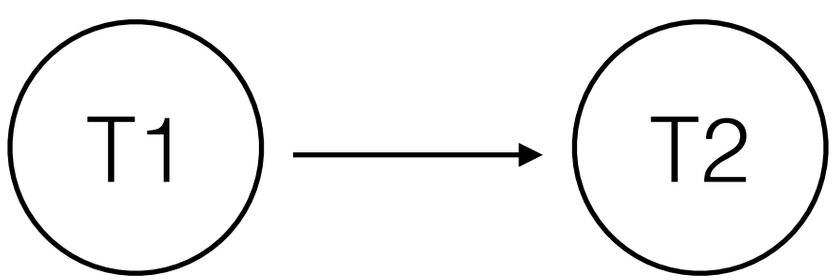
- <u>Dependency Graph</u> (aka <u>Precedence Graph</u>):
  - One node per xact
  - Edge from Ti to Tj if some operation O_i is earlier than and conflicts with O_j

Conflict Serializability

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

T2: R(B) W(B)

- <u>Dependency Graph</u> (aka <u>Precedence 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?
    - What about view 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)

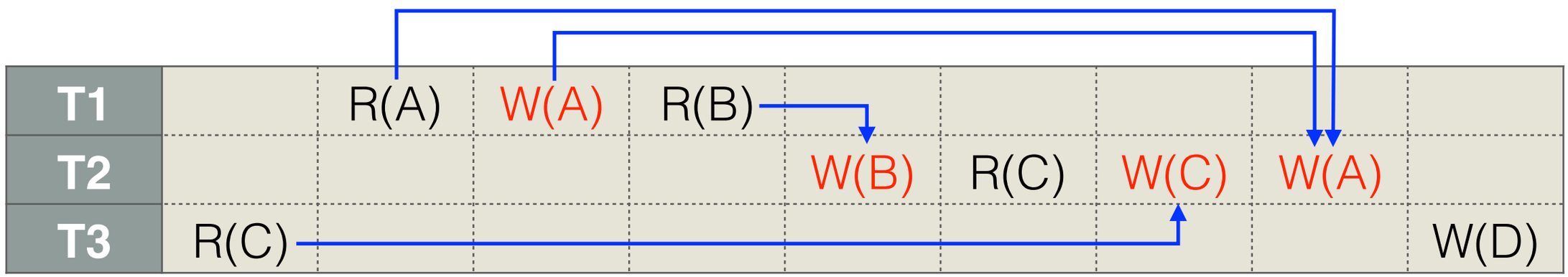
- Dependency Graph
  - Conflicting writes? Edges?

Worksheet - 2a

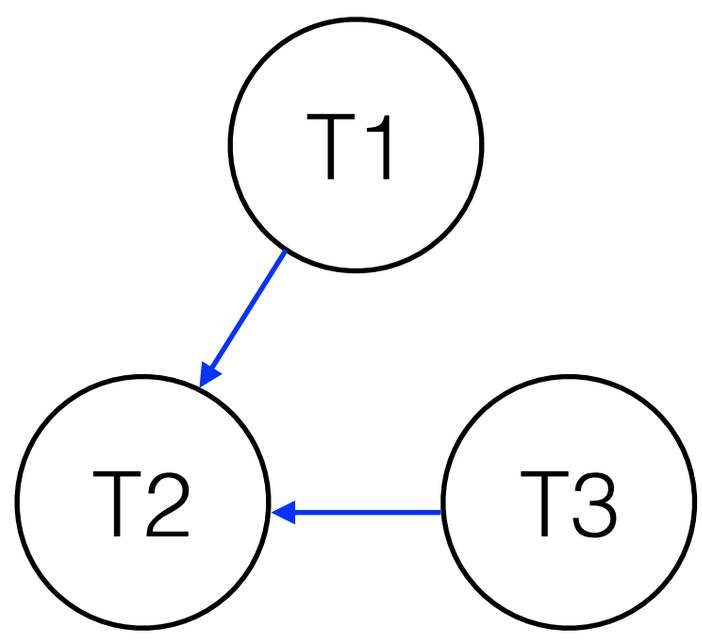
T1		R(A)	W(A)	R(B)					
T2					W(B)	R(C)	W(C)	W(A)	
T3	R(C)								W(D)

- Dependency Graph
  - Conflicting writes? Edges?

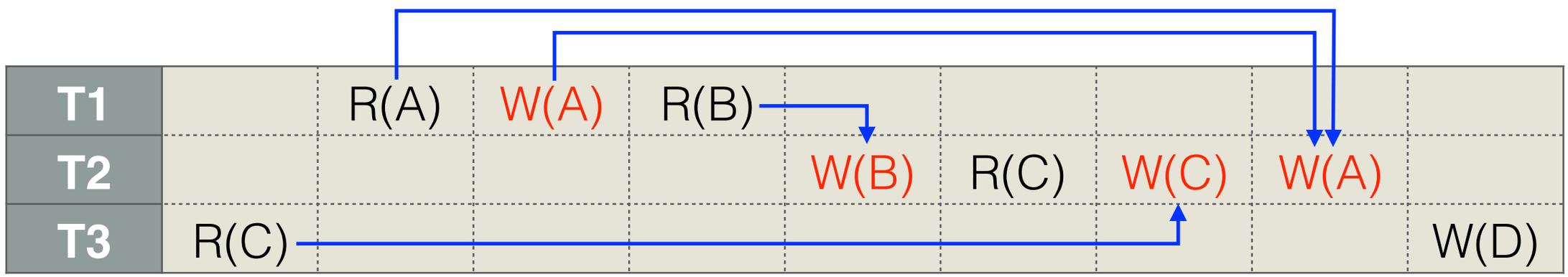
Worksheet - 2a



- 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)

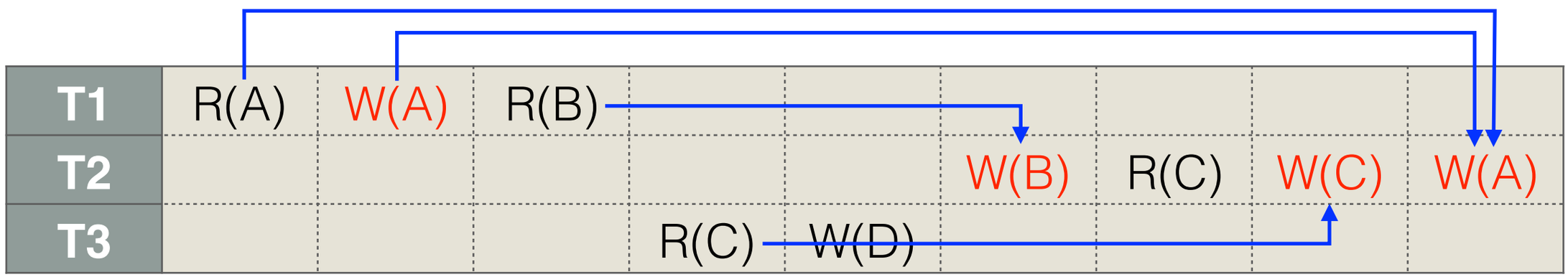
- Conflict Serializability
  - Dependency Graph? Acyclic!
  - What actions would we move? T3

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

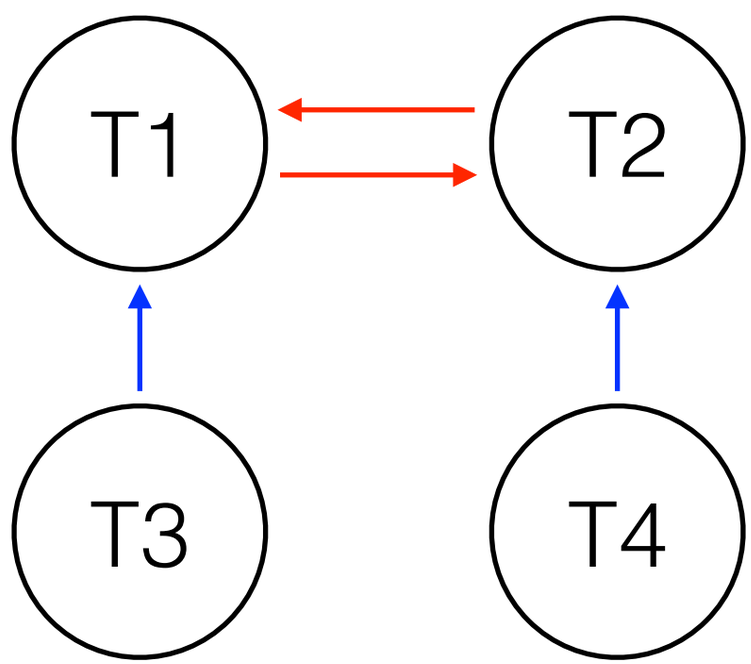
T1	R(A)		R(B)				W(A)	
T2		R(A)		R(B)				W(B)
T3					R(A)			
T4						R(B)		

- 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?

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

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

Worksheet - 3

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

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

Worksheet - 3

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

- Conflict serializable?
- View Serializable?
- Serializable?

Worksheet - 3

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

- Conflict serializable? ✓
- View Serializable? ✓
- Serializable? ✓

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?

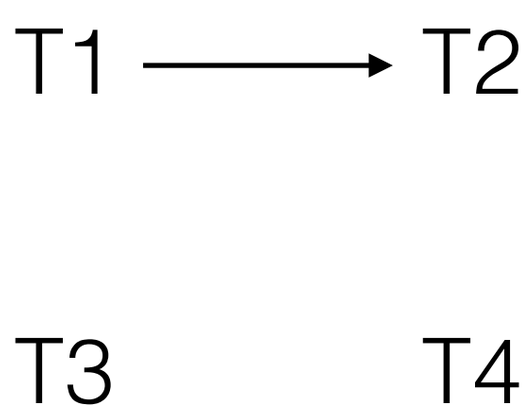
T1 T2

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)	

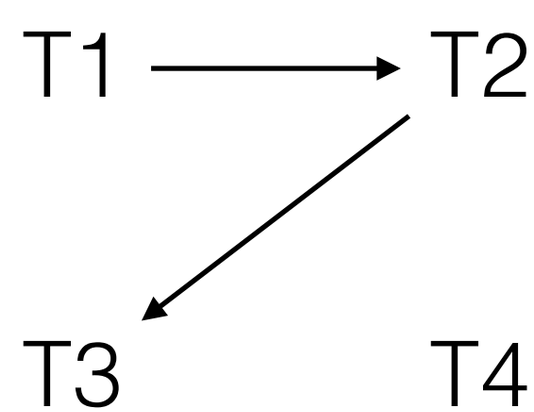
- What needs to wait?



Deadlock Detection

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

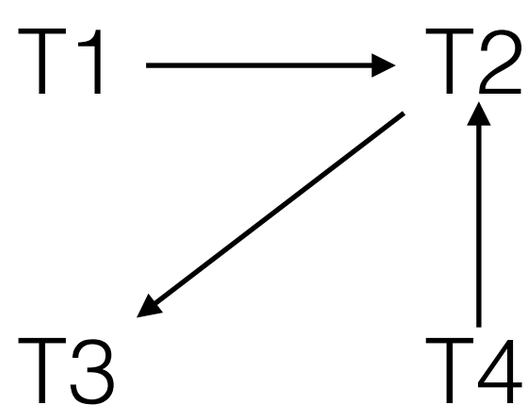
- What needs to wait?



Deadlock Detection

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

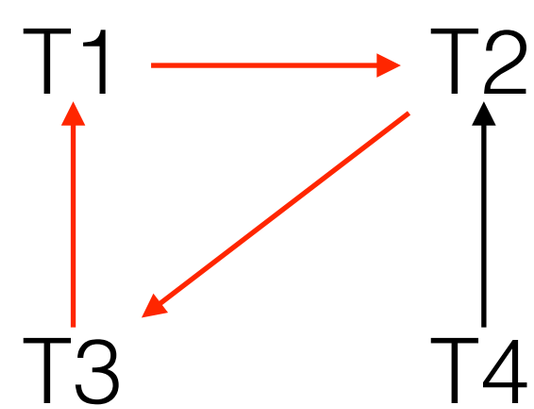
- What needs to wait?



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

Locking Granularity

- Databases have <u>hierarchies</u>
  - (Top) Databases > tables > pages > tuples
  - Assign locks top down, release bottom up
- Use intent locks!
  - IS - intent to get S
  - IX - intent to get X
  - SIX - S and IX

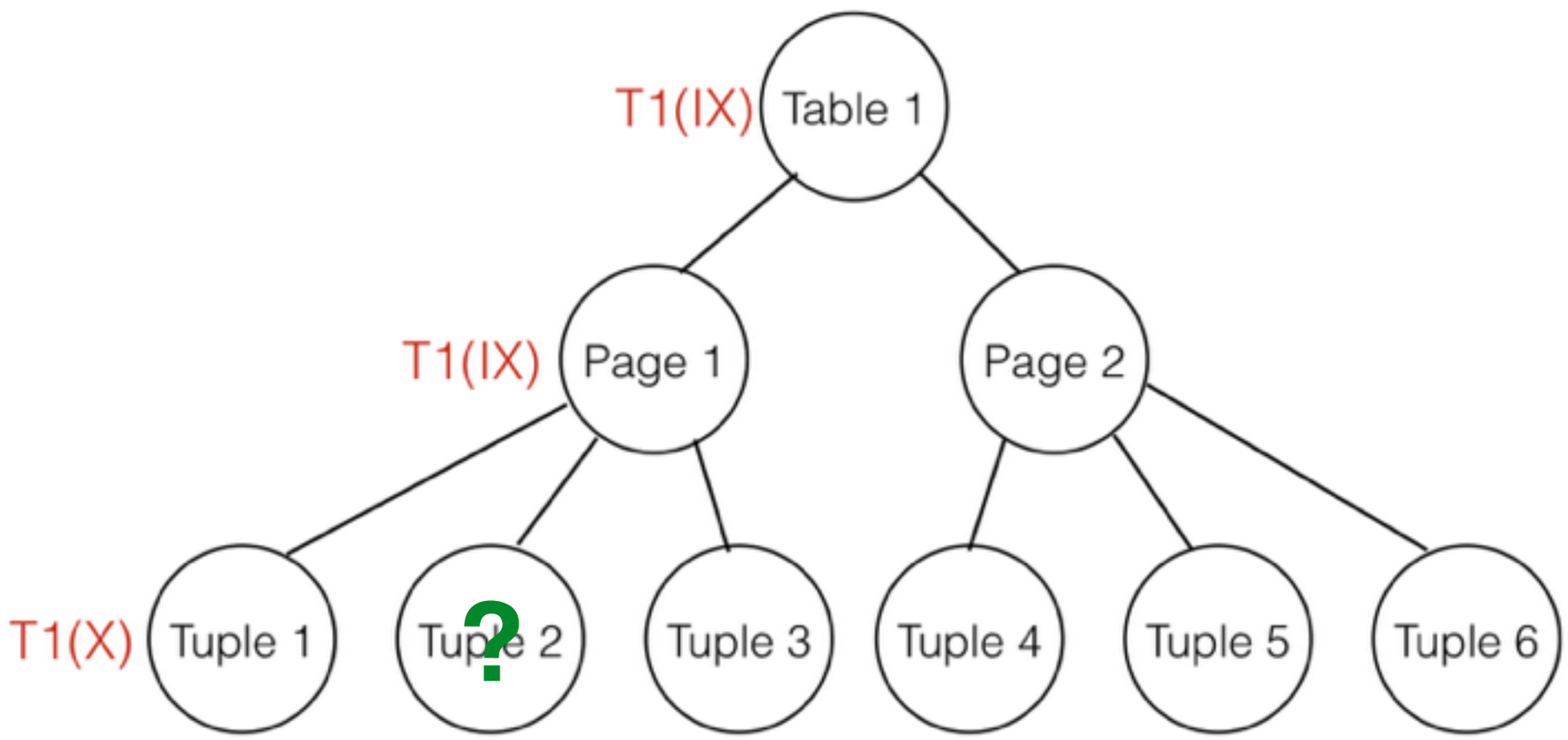
Locking Granularity

	IS	IX	SIX	S	X
IS	√	√	√	√	–
IX	√	√	–	–	–
SIX	√	–	–	–	–
S	√	–	–	√	–
X	–	–	–	–	–

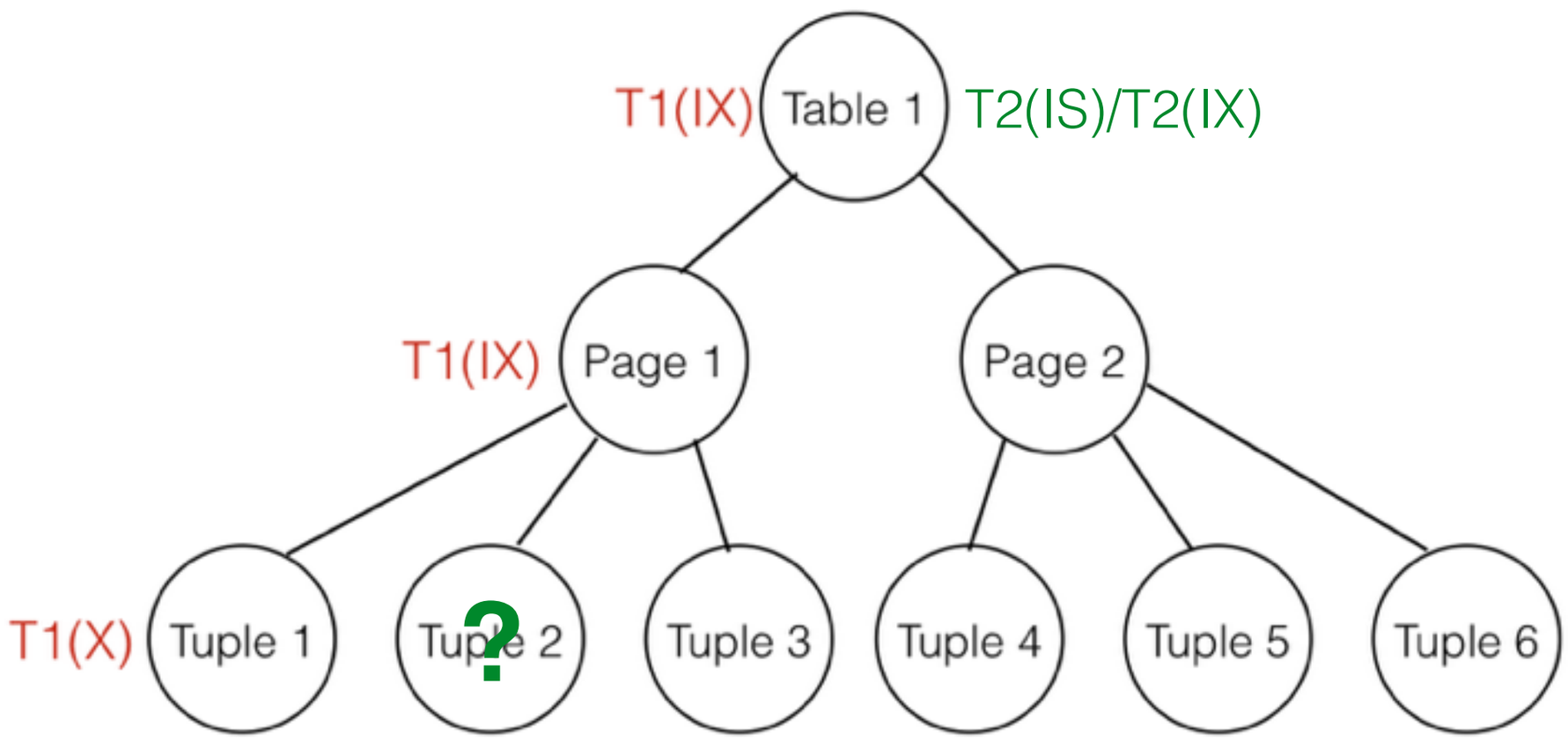
From Wikipedia:

To Get	Must Have on all Ancestors
IS or S	IS or IX
IX, SIX or X	IX or SIX

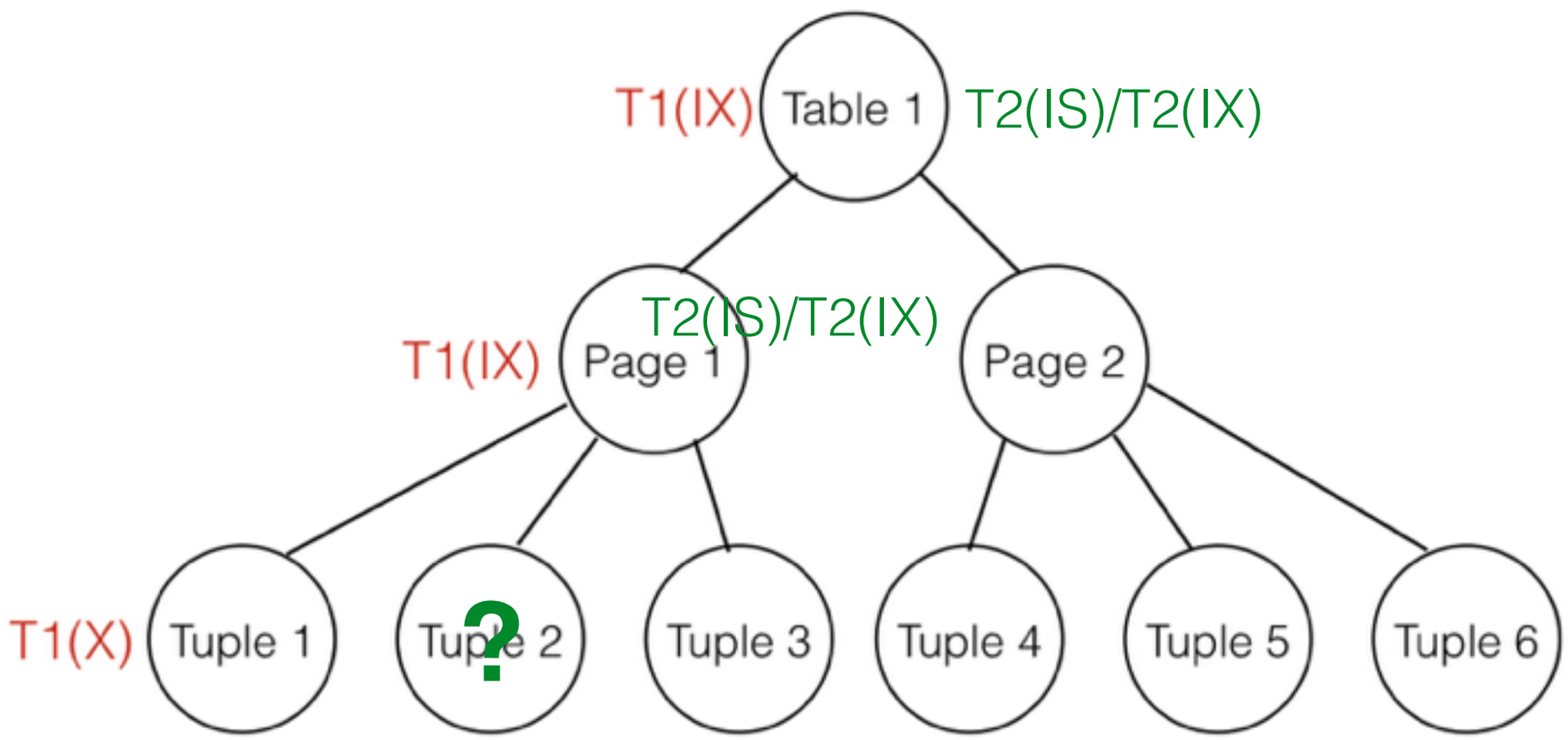
Worksheet - #3


T1(IX)
Table 1
T1(IX)
Page 1
Page 2
T1(X)
Tuple 1
Tuple 2
?
Tuple 3
Tuple 4
Tuple 5
Tuple 6

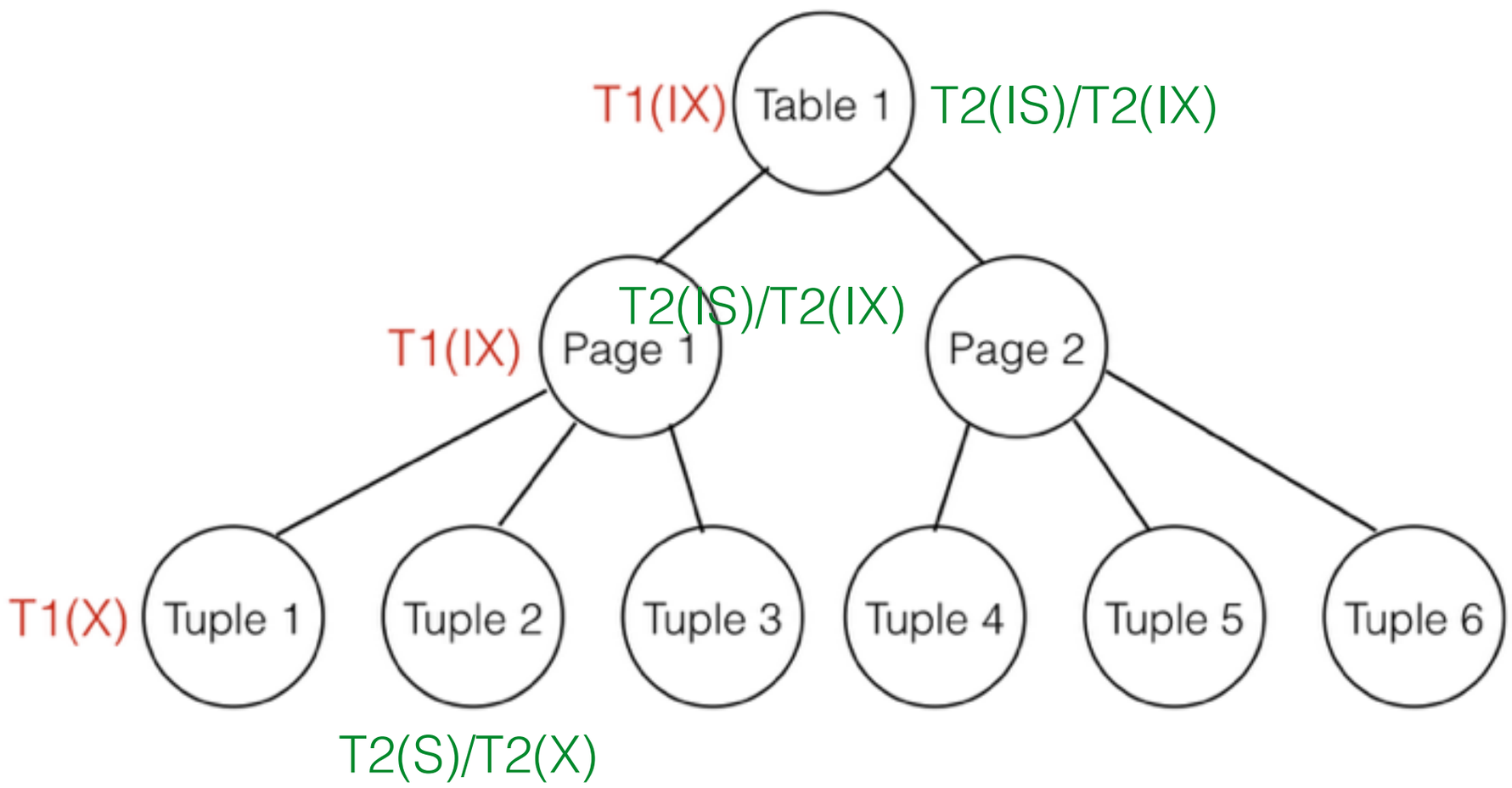
Worksheet - #3


T1(IX)
Table 1
T2(IS)/T2(IX)
T1(IX)
Page 1
Page 2
T1(X)
Tuple 1
Tuple 2
?
Tuple 3
Tuple 4
Tuple 5
Tuple 6

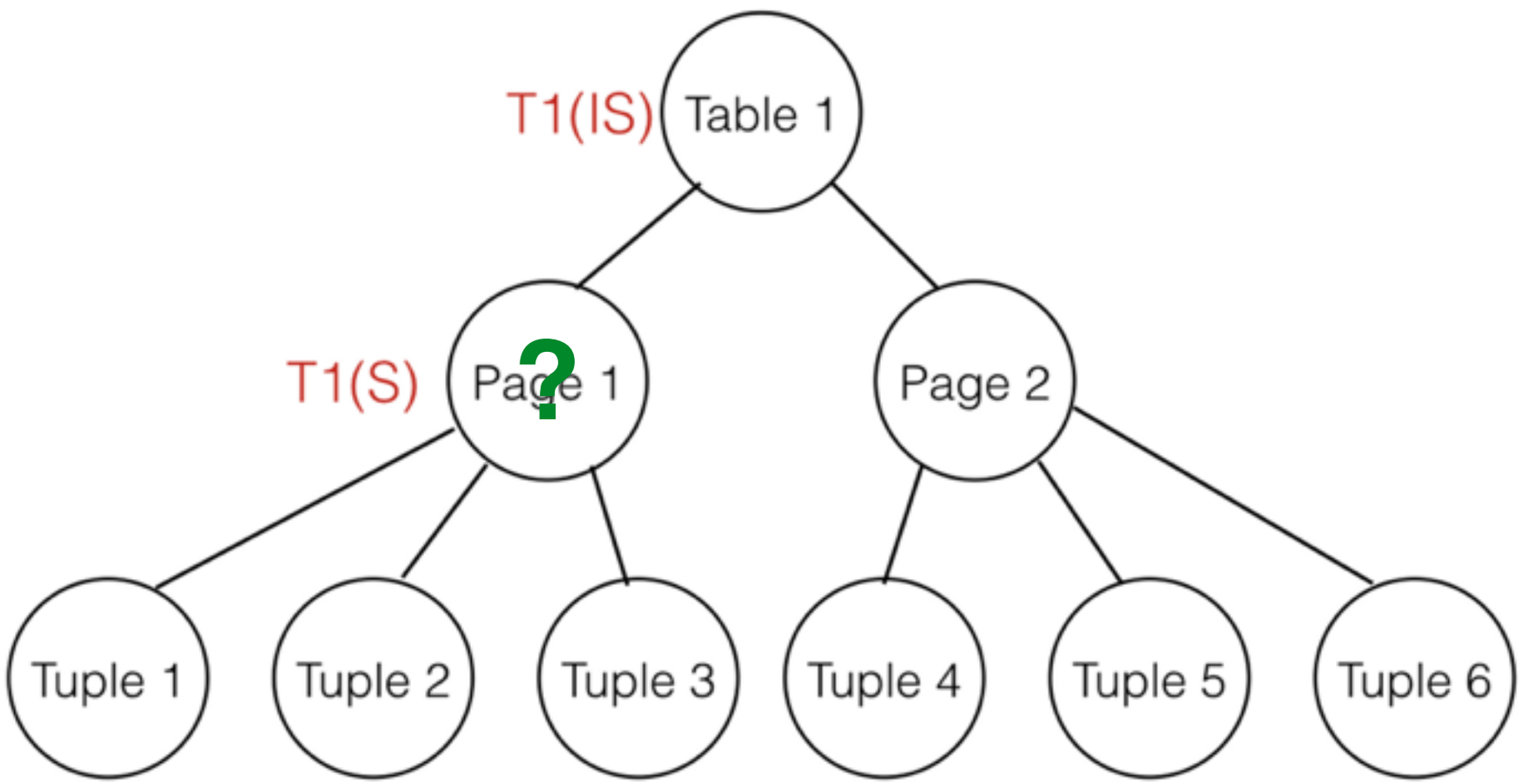
Worksheet - #3


T1(IX)
Table 1
T2(IS)/T2(IX)
T1(IX)
Page 1
T2(IS)/T2(IX)
Page 2
T1(X)
Tuple 1
Tuple 2
?
Tuple 3
Tuple 4
Tuple 5
Tuple 6

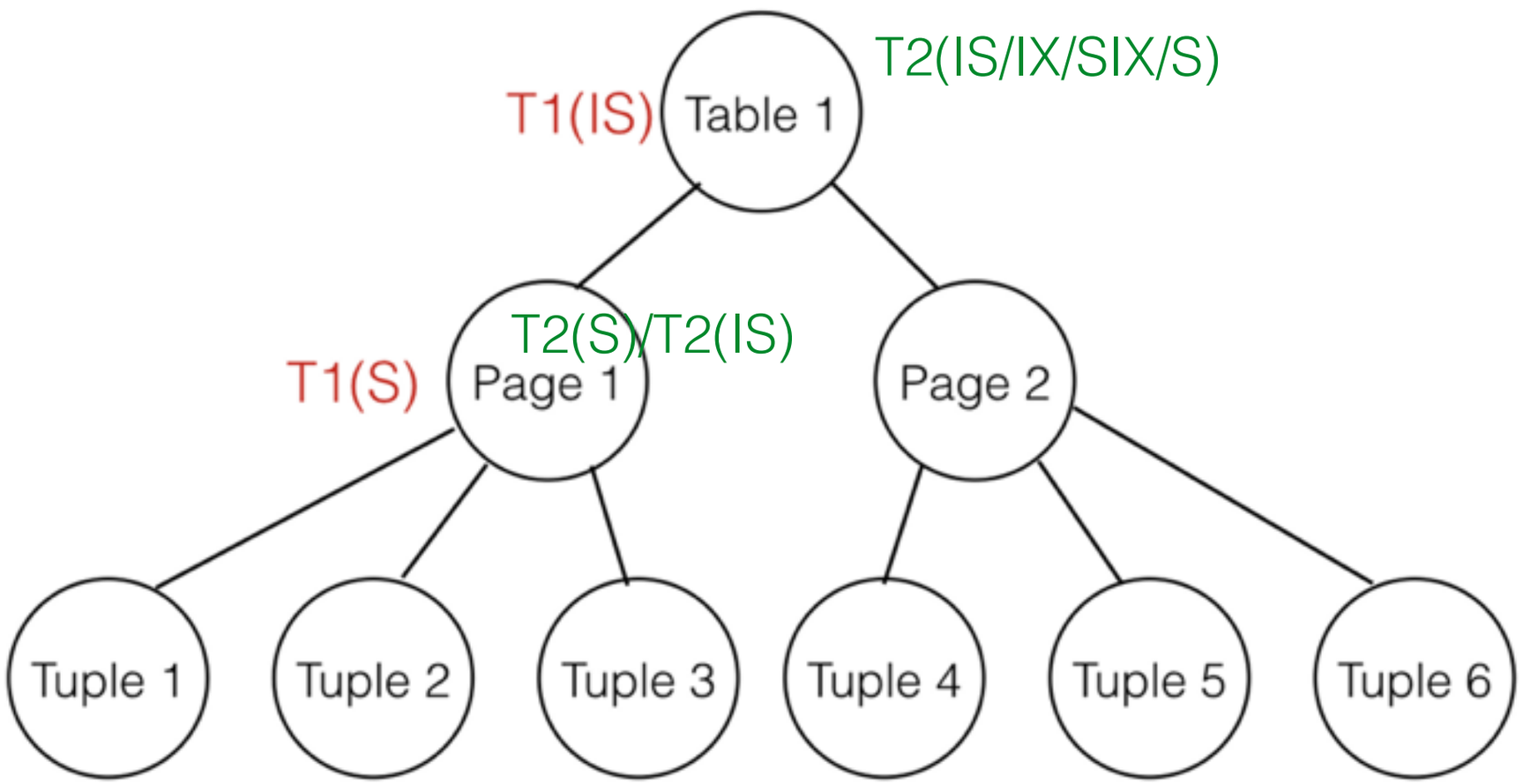
Worksheet - #3


T1(IX)
Table 1
T2(IS)/T2(IX)
T2(IS)/T2(IX)
T1(IX)
Page 1
Page 2
T1(X)
Tuple 1
Tuple 2
Tuple 3
Tuple 4
Tuple 5
Tuple 6
T2(S)/T2(X)

Worksheet - #3


T1(IS)
Table 1
T1(S)
Page 1
?
Page 2
Tuple 1
Tuple 2
Tuple 3
Tuple 4
Tuple 5
Tuple 6

Worksheet - #3


T2(IS/IX/SIX/S)
T1(IS)
Table 1
T2(S)/T2(IS)
T1(S)
Page 1
Page 2
Tuple 1
Tuple 2
Tuple 3
Tuple 4
Tuple 5
Tuple 6