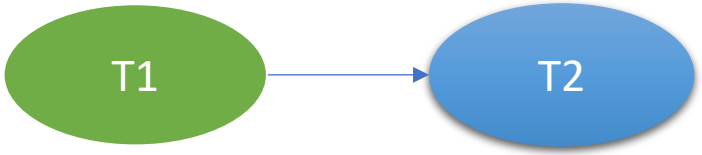
Q1:
A:
READ1(X,Y) and WRITE2(Y),
READ1(X,Y) and WRITE2(X)

B:
No

C:
WRITE1(X) and WRITE2(X)

D
Yes, serializable



Q2-A:

T1	T2	T3
	XLOCK2(X)	
	READ2(X)	
	SLOCK2(Y)	
	READ2(Y)	
		SLOCK3(Y)
		READ3(Y)
		XLOCK3(Z)
		READ3(Z)
{T1 requests XLOCK on Y but is blocked by T2 or T3}		
	WRITE2(X)	
	COMMIT2+REL2(X,Y)	
{T1 requests XLOCK on Y but is blocked by T2 or T3}		
		WRITE3(Z)
		COMMIT3+REL3(Y,Z)
XLOCK1(Y)		
READ1(Y)		
WRITE1(Y)		
COMMIT1+REL1(Y)		

Equivalent serial schedule:

T2 -> T3 -> T1, or
T3 -> T2 -> T1

No deadlock

Q2-B:

T1	T2	T3
	XLOCK2(X)	
	READ2(X)	
	SLOCK2(Y)	
	READ2(Y)	
		SLOCK3(Y)
		READ3(Y)
		XLOCK3(Z)
		READ3(Z)
{T1 requests XLOCK on Y but is blocked by T2 or T3}		
	WRITE2(X)	
	COMMIT2+REL2(X,Y)	
{T1 requests XLOCK on Y but is blocked by T2 or T3}		
		WRITE3(Z)
		COMMIT3+REL3(Y,Z)
XLOCK1(Y)		
READ1(Y)		
WRITE1(Y)		
COMMIT1+REL1(Y)		

Equivalent serial schedule:

T2 -> T3 -> T1, or
T3 -> T2 -> T1

Q2-C:

T1	T2	T3
	XLOCK2(X)	
	READ2(X)	
	SLOCK2(Y)	
	READ2(Y)	
		SLOCK3(Y)
		READ3(Y)
		XLOCK3(Z)
		READ3(Z)
{T1 requests XLOCK on Y but is blocked by T2 or T3}		
	ABORT2+REL2(X,Y)	
		ABORT3+REL3(Y,Z)
XLOCK1(Y)		
READ1(Y)		
WRITE1(Y)		
COMMIT1+REL1(Y)		

Equivalent serial schedule:

T1

Q2-D:

T1	T2	T3
BEGIN		
XLOCK1(Y)		
READ1(Y)		
WRITE1(Y)		
COMMIT1+REL(Y)		
	BEGIN	
	XLOCK2(X)	
	SLOCK2(Y)	
		BEGIN
		SLOCK3(Y)
		XLOCK3(Z)
	READ2(X)	
	READ2(Y)	
		READ3(Y)
		READ3(Z)
	WRITE2(X)	
	COMMIT2+REL2(X,Y)	
		WRITE3(Z)
		COMMIT3+REL3(Y,Z)

Equivalent serial schedule:

T1 -> T2 -> T3

Q3:
A: read uncommitted: because it only has read action

B: read committed: because it has a write action

C: not possible, because T2 will first get the XLOCK for A and A cannot get the SLOCK immediately after READ2(A)

D: possible, T1 will first get the SLOCK1(A) and READ1(A), but will release the SLOCK immediately after READ1(A), so that T2 can be granted with XLOCK.

E: possible, T1 and T3 only have leveraged A in common, both shared lock, so there won't conflict.

Q4:

A:
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
UPDATE box SET content=”Tangerine” WHERE id=1;
SELECT content FROM box WHERE id=1;
COMMIT;

B:
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;
SELECT content FROM box WHERE id=1;
SELECT content FROM box WHERE id=1;
COMMIT;

Q4:

C:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
SELECT content FROM box WHERE id=1;
SELECT content FROM box WHERE id=1;
SELECT content FROM box WHERE id=1;
COMMIT;
```

```
SET TRANSACTION ISOLATION LEVEL REPETABLE READ;
BEGIN TRANSACTION;
SELECT content FROM box WHERE id=1;
SELECT content FROM box WHERE id=1;
SELECT content FROM box WHERE id=1;
COMMIT;
```

Q4:

D:
First: Orange
Second: Tangerine

E:
First: Orange
Second: Tangerine
Third: Tangerine

F:
First: Orange
Second: Orange
Third: Orange