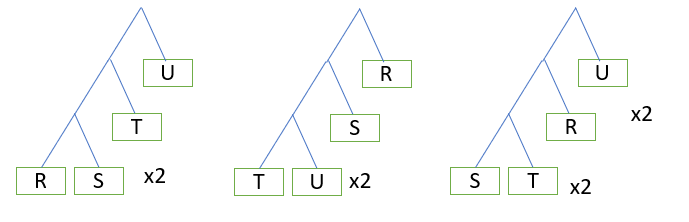
2.// Each page can hold 30 attr. RSTU has 6 attr. So 5 tuples per page

# tuples = 100\*200\*300\*400 / (50\*60\*100) = 8000

# pages = 8000 / 5 = 1600

3.// 8 trees



4. 1,2,3,5

5.// a,c,d

6. b,c,d,e (missing a)

(Argument against a: If you do not know your plan space, you don’t even know if the generated plan is inside the plan space, as such you cannot verify if your plan is legitimate) but I guess a non-legitimate query plan is still a plan//

7. Since V(R,A) < V(R,B), every A value will map to some B by the inclusion assumption (but not every B will map to some A). # of tuples = ||R|| / V(R,A) = 1000 / 10 = **100**

Might be 200.

Imagine dividing everything by 10. So ||R|| = 100, V(R,A)=1, V(R,B) = 5. All 100 tuples have the same value for A. There will be 20 tuples for each value of B. At most 20 will match. So it is just 100/5=20 tuples

Supposed for example V(R,A)=2 instead of 1. Then there will be 50 tuples for each value of A. Then there will be 20+20=40 matches

Now, going back to ||R||=1000, V(R,A)=10, V(R,B)=50. There will be 100 tuples for each of the 10 buckets of A. There will be 20 tuples for each of the 50 buckets of B. There can be at most 10 values that match, since there are only 10 distinct values of A. And the maximum number of matches for each value is at most 20, because there are only 20 tuples for each distinct value of B. So 10\*20 gives 200.

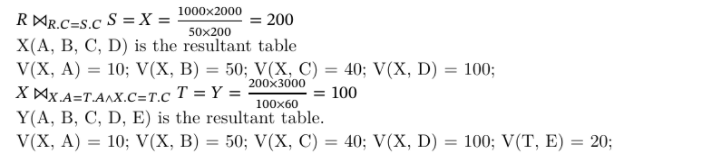
8. R join S join T

# tuples = 1000\*2000\*3000 / max(A) \* max(B) \* max(C)

= 6,000,000,000 / (100 \* 50 \* 200)

= **6000**

100 //



9.// b,c,d

10.// a,c

11. a,b (I think a,b works)

a,b,c only

12.// a,d

(e is probably wrong since serializability only deals with isolation of AC**I**D)

13. c

b,c only//

14.// c

15.//

1. FALSE. T3 val success
2. FALSE. T1 val success
3. TRUE.
4. FALSE.
5. TRUE. T2 val success
6. TRUE. T5 val success
7. FALSE.
8. FALSE.

16.// XXX={ABF} (max=3) YYY={ABDF} (max=4)

17.// c

(1->3, 3->1)

18. b

S1: **W1(X); R2(X);** W3(Y); W1(Y); **W3(Y);** W2(X); R3(Y); R2(Y);

S2: **W1(X);** W1(Y); W3(Y); **W3(Y);** R3(Y); **R2(X);** W2(X); **R2(Y);**

1->3->2

a only

19.// a,c,d

20. d

c only//

21. a,b,c

a,b only//

22.// a

23.// b,d

T1 locks X

T2 locks Y

T3 locks Z

T1 waits for T2

T2 waits for T3

T3 aborts

T2 locks Z

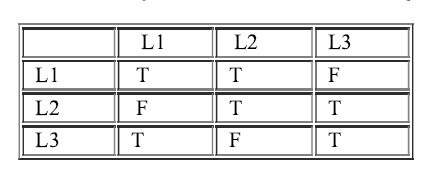
T2 completes and releases Y and Z

T1 locks Y

T1 completes

2->1->3

24. b (hold L3 first, then L1), c//



25. //b

(undo logging not allowed here as it CHKPT end can only occur after T2 commits)

26.\*\* b,d

b,c only (faulty)

27.//

X=30, 40 (T committed, so <T,X,20,30> which is before Start ckpt must be reflected in disk)

Y=10,20,30,40

28.// a

29. a,c

(c might be true since u need to wait for all active Xacts to commit/abort)

a only//

30.\*\* b,f (not sure)

a,b,d,f only//

31. b,d (I think d is really true, cos it refers to FORCE)

b only

32. ??

a,b only

33.

1. FALSE LSN 2
2. FALSE. LSN 5
3. TRUE. LSN 12
4. TRUE. LSN 10
5. FALSE. LSN 16 (not committed yet)
6. FALSE. Redo LSN 7

ii,iii,iv,vi only//

For redo logging, u go all the way back

34. d (by elimination)

(Value of A can be 2 (need not be), and must not be 1 and 3)

None are correct. Faulty question //