1617SEM2-CS3223

1.

||R(a,b,c)|| = 500, ||S(c,b,d)|| = 500, ||T(d,b,e)|| = 1000

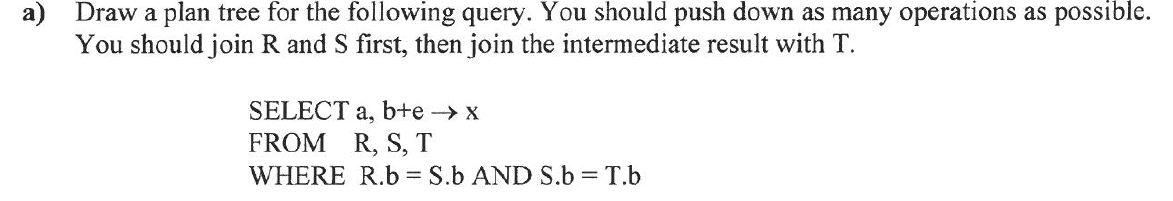
|R| = 50, |S| = 50, |T| = 100

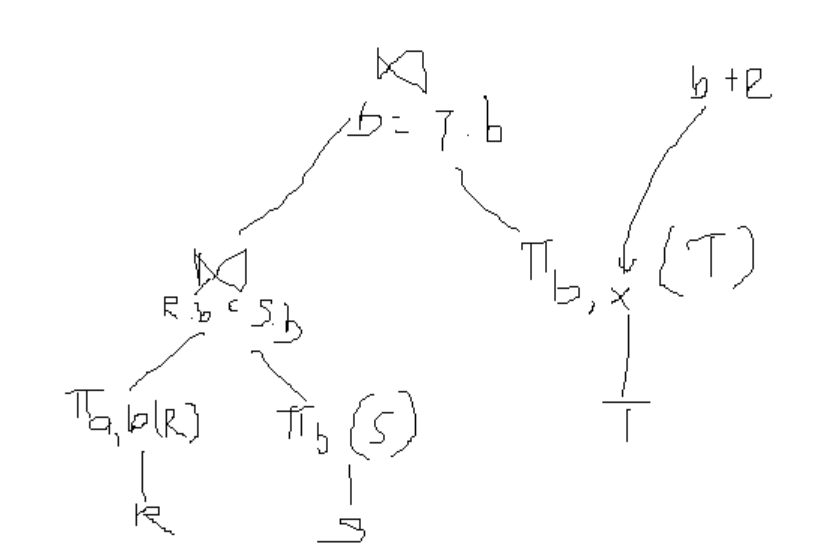
Dom(b) = [0, 99]

S(attr) same

1 page can hold 10 tuples of R, i.e. 1 page can hold 30 attr

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | Others |
| R.b | **72** | **30** | **10** | 4 | 4 | 388 |
| S.b | 2 | **40** | **16** | **250** | 2 | 194 |
| T.b | **100** | 9 | **10** | 9 | **17** | 873 |





(b) Estimate size of query in terms of pages and tuples

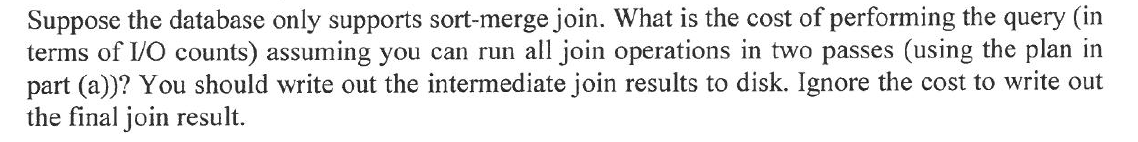
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | [95 col] | Others |
| R.b | **72** | **30** | **10** | 4 | 4 | 4 | 388 |
| S.b | 2 | **40** | **16** | **250** | 2 | 2 | 194 |
| T.b | **100** | 9 | **10** | 9 | **17** | 9 | 873 |
| R Jn S Jn T | 14400 | 10800 | 1600 | 9000 | 136 | 72 x 95 |  |

# tuples = 42776

Size per tuple = 2 attr

# tuples per page = 15

# pages = ceil(42776 / 15) = 2852

(c) 

Project a,b of R = 50 + 500/15 = 84 I/O [34 pages]

Project b of S = 50 + 500/30 = 67 I/O [17 pages]

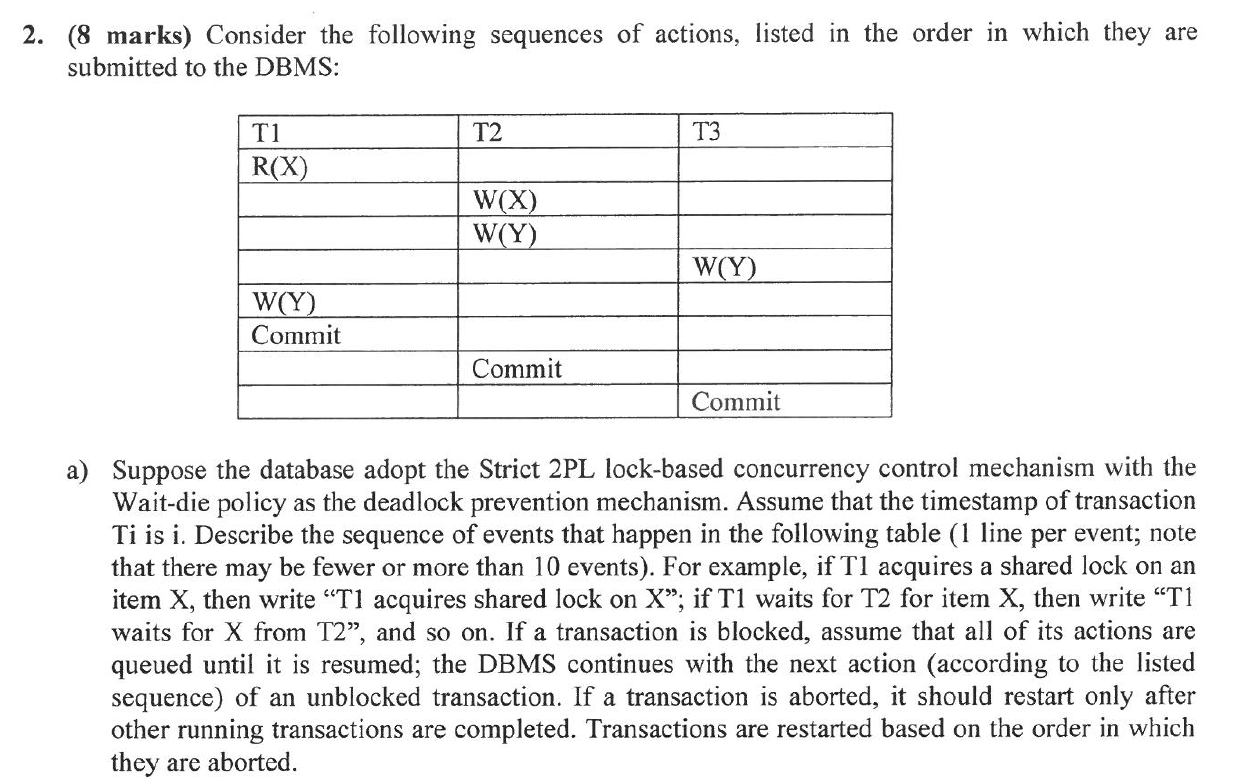
SMJ of R and S = 2\*|R|\*2 passes + 2\*|S|\*2 passes + |R|+|S| = 255 I/Os [3272 tup, 219 pages]

Write out Temp = 219 I/O

Project b,x of T = 100 + 1000/15 = 167 I/O [67 pages]

SMJ of Temp and T = 2\*|Temp|\*2passes + 2\*|T|\*2passes + |Temp| + |T| = 2860 I/O

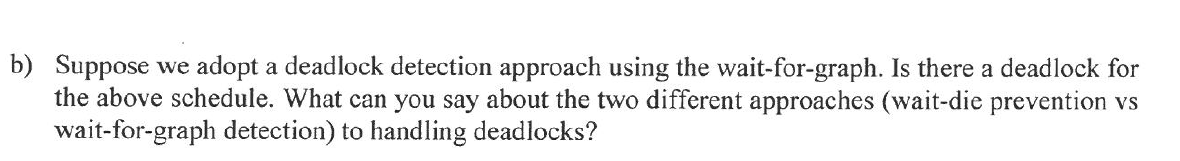
Total = 3902 I/O



Strict 2PL – Xacts hold on to locks until it commits/aborts

Wait-Die – High priority waits for low priority, Low priority does not wait and abort

|  |  |
| --- | --- |
| Event No | Event |
| 1 | T1 acquires shared lock on X |
| 2 | R1(X) |
| 3 | T2 waits for X from T1 (2🡪1); T2 aborts |
| 4 | T3 acquires exclusive lock on Y |
| 5 | W3(T) |
| 6 | T1 waits for Y from T3 |
| 7 | T3 commits  T3 releases exclusive lock on Y |
| … | T1 acquires exclusive lock on Y |
|  | W1(Y) |
|  | T1 commits  T1 releases shared lock on X  T1 releases exclusive lock on Y |
|  | T2 acquires exclusive lock on X |
|  | W2(X) |
|  | T2 acquires exclusive lock on Y |
|  | W2(Y) |
|  | T2 commits  T2 releases exclusive lock on X  T2 releases exclusive lock on Y |



(confused)

T1 acquires X first

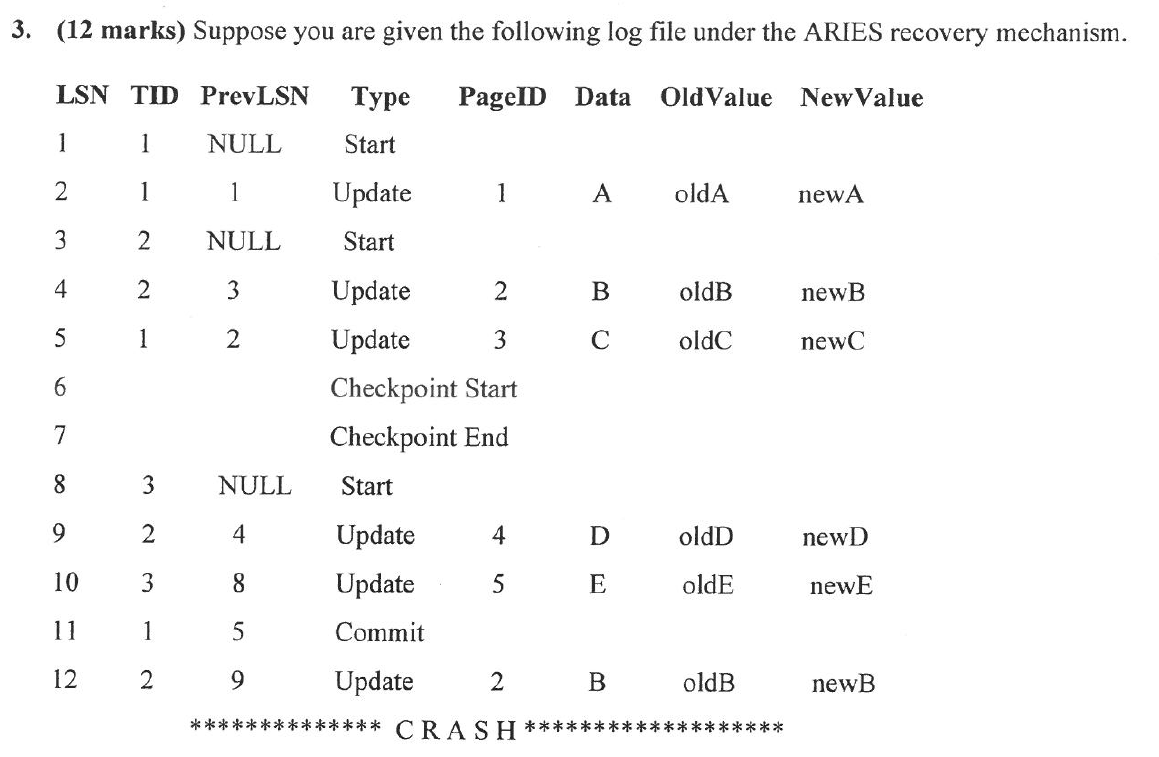
2🡪1

T2 acquires Y first (do we continue with the schedule (let T2 acquire lock for Y?)

3🡪2

1🡪2

Wait-for-graph leads to more wasted steps as deadlock is only detected at a later stage and abortion would mean repeating of more steps. Wait-die on the other hand terminates a waiting low-priority transaction early so that it doesn’t carry out actions that will eventually be rolled-back.

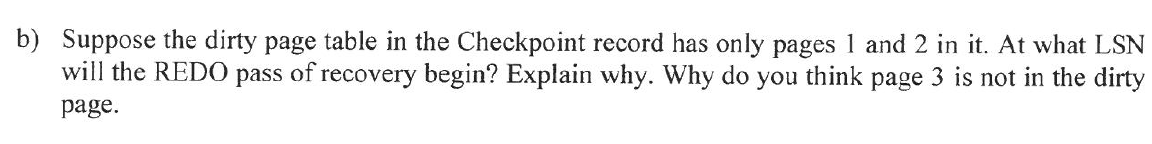


ARIES recovery mechanism - undo/redo logging

(a) After the recovery, which transactions will be committed and which will be aborted?

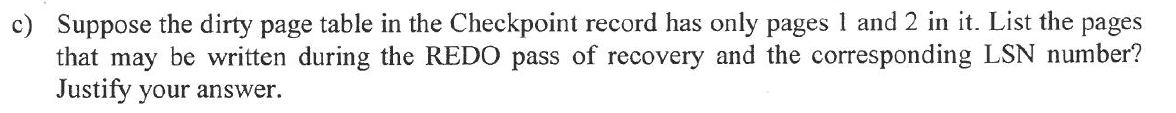
Committed:1

Aborted: 2,3



Start from LSN=6. Everything before CHKPT START is guaranteed to be flushed to disk. So only need to redo from CHKPT START onwards.

Why is 3 not in dirty page? Delayed write to minimize disk I/O



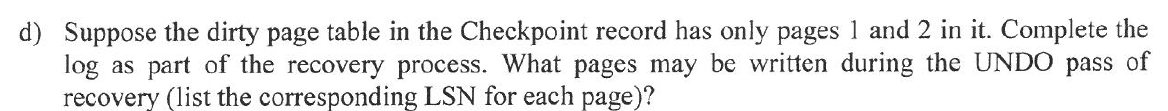
[4] T2 updates B

[9] T2 updates D

[12] T2 updates D

(T1 will be redone – but no T1 actions after checkpoint start)

(T2 and T3 will be undone)



[4] T2 updates B

[9] T2 updates D

[12] T2 updates D

[10] T3 updates E