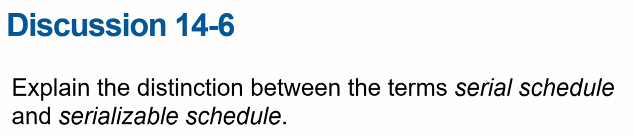
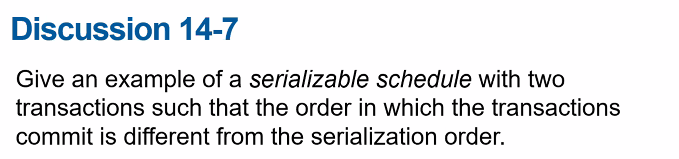
**Discussion 06/08**

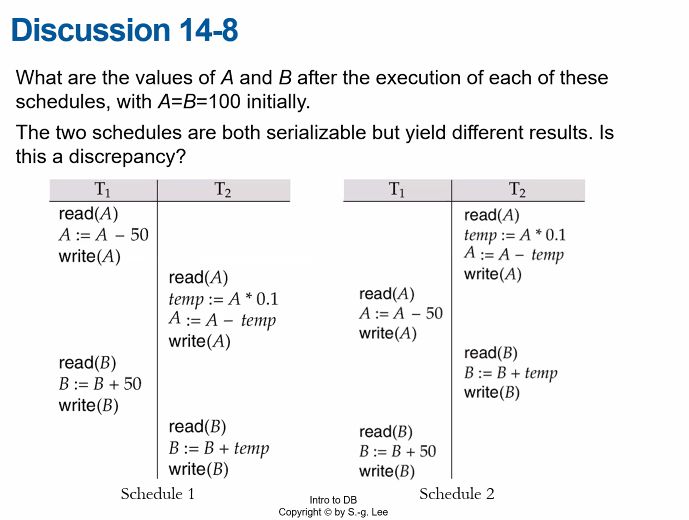


Serial 스케쥴은 트랜잭션이 순서대로 실행되는 것. Serializable은 병렬으로 실행되는데 그 실행 결과가 serial 스케쥴과 같으면 serializable 이라고 한다.



T2 – read A, write A => T1 – read A, commit => T2 - commit

이러면 T1에서 먼저 커밋이 되었지만 serial하게 바꾸면 T2 커밋이 먼저 되어야 한다.



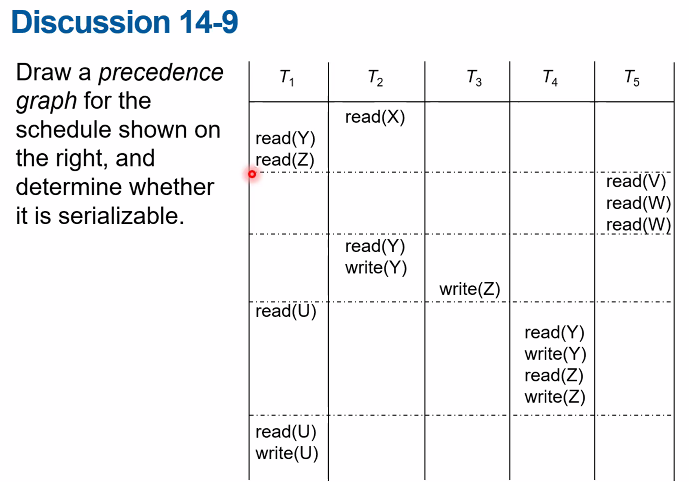
S1은 T1 실행 후 A 50 => T2 실행 후 A 45, temp는 5 => T1 실행 후 B 150 => T2 실행 후 B 155

최종적으로 A 45, B 155.

S2는 T2 실행 후 A 90, temp는 10 => T1 실행 후 A 40 => T2 실행 후 110 => T1 실행 후 160

최종적으로 A 40, B 160.

=> 교수님 설명: T1과 T2는 독립적인 트랜잭션. 각각은 valid. 그래서 어떤 걸 먼저 수행하는지는 DBMS의 컨트롤 영역 밖. 따라서 둘 다 valid.

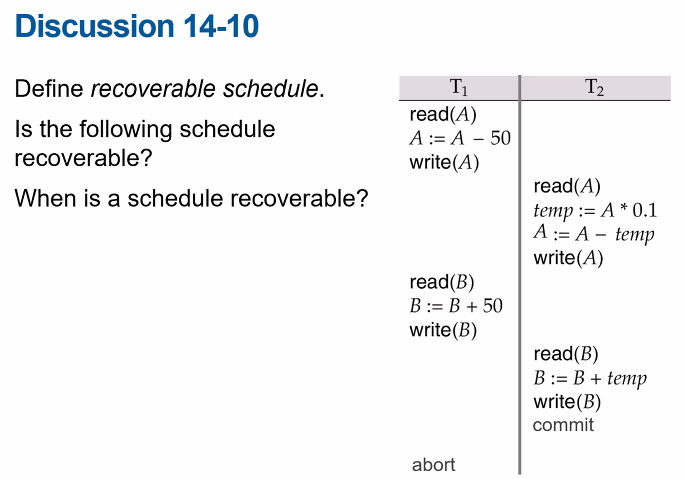


T1 => T2, T3, T4

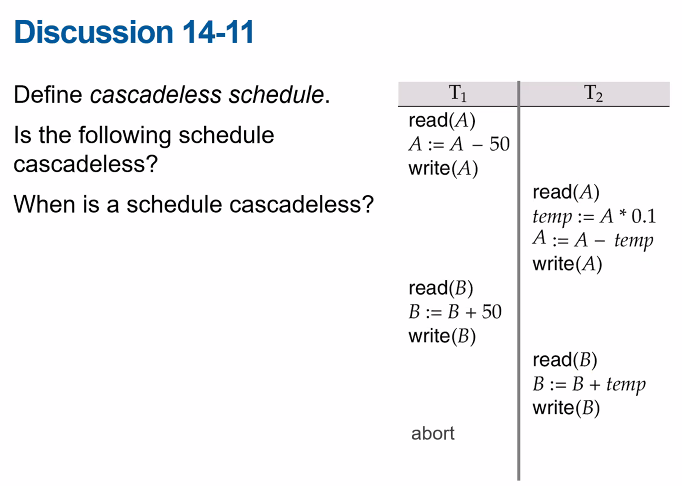
T2 => T4

T3 => T4

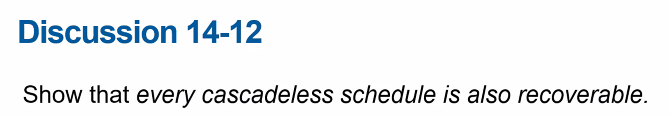
사이클이 없기 때문에 serializable함.



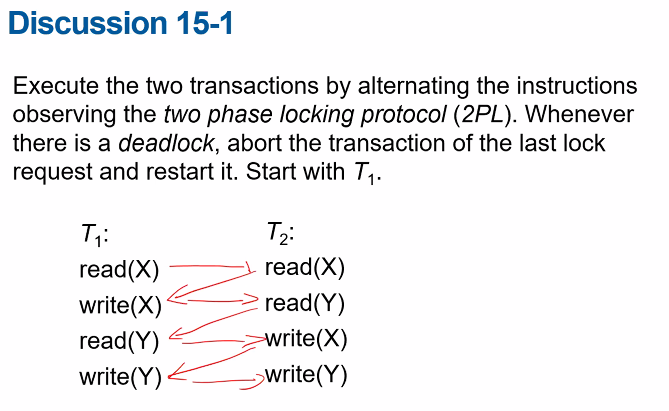
해보기!



이것도 해보기!



Cascadeless는 한 트랜잭션에서 write를 하고 commit한 뒤에 다른 트랜잭션에서 read를 한다. Recoverable의 정의가 write, read가 있을 때 write하는 트랜잭션의 커밋이 read하는 트랜잭션의 커밋보다 빨라야 하는 것이므로, cascadeless에서는 write한 뒤 커밋을 하지 않으면 다른 트랜잭션에서 읽을 수가 없으므로 항상 성립한다.

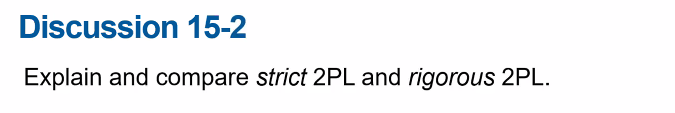


~~T1 read(X) => T2 read (X) => T1 write(X). 여기서 T1에서 X에 대해 lock-X 획득 => T2 read(Y) => T1 read(Y) => T2에서 write(X)를 할 때 deadlock 발생.~~

이거 틀림!

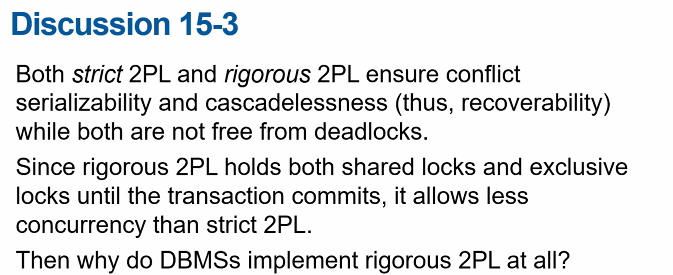
=> T1에서 write(X)를 할 때 T2도 X에 대해 lock-S를 들고 있으므로 기다려야 함. 그래서 T2에서 read(Y)를 하고 write(X)를 할 때는 T1에서 lock-S 있으므로 데드락 발생.

그래서 T2를 롤백함. => T1이 write(X) 수행. 그 뒤 T2에서 T1이 X에 대해 lock-X를 들고 있으므로 read(X)를 할 때 기다림. => T1이 read(Y) 수행. T2는 여전히 기다림. => T1이 write(Y)까지 수행. => T1이 모든 lock 해제. => T2에서 read(X)부터 write(Y)까지 모두 수행.



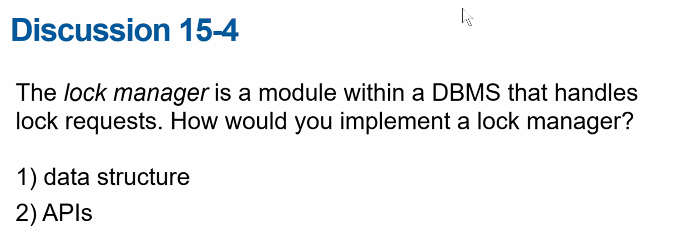
Strict 2PL은 exclusive lock을 commit이나 abort할 때까지 release하면 안 되는 규칙.

Rigorous 2PL은 shared lock까지 포함.



Rigorous 2PL은 모든 락을 획득하는 순간 release할 수 없으므로 트랜잭션을 serialize하기 쉬움. 트랜잭션들이 커밋된 순서대로 serialize 하면 됨.

=> 트랜잭션 매니저가 growing phase인지 shrinking phase인지 판단하기가 거의 불가능에 가까움. 그래서 구현을 편하게 하기 위해 rigorous 2PL을 많이 씀.



해보기!