Exercises on Concurrency Control (part 1)

Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza"

Anno Accademico 2017/2018

http://www.dis.uniroma1.it/~lenzerin/index.html/?q=node/53

Exercise 1

Consider the following sequence of actions S, and answer the following questions:

1. Is the schedule S view-serializable? If so, provide a view-equivalent serial schedule
2. Is the schedule S conflict-serializable? If so, describe all the conflict-equivalent serial schedules
3. Is the schedule S a 2PL schedule (with exclusive locks)?
4. Is the schedule S a 2PL schedule (with shared and exclusive locks)?

S: *r1(A), r2(A), r3(B), w1(A), r2(C), r2(B), w2(B), w1(C)*

Solution to Exercise 1

Since we know that:

- every 2PL schedule is conflict-serializable,
- every conflict-serializable schedule is view-serializable,

let us first check whether S is in 2PL with exclusive locks.

Solution to Exercise 1.3

3. Is the schedule 2PL (with exclusive locks)?

- In order for T2 to read A, it is necessary that T1 releases the lock on A (which was obtained by T1 for reading A)
- It follows that T1 enters the shrinking phase when T2 reads A
- In order for T1 to write A, it is necessary that T1 gets again the lock on A
- T1 cannot get the lock on A before T2 reads A
- It follows that T1 should request a lock during the shrinking phase

IMPOSSIBLE: S is not a 2PL schedule with exclusive locks

Solution to Exercise 1.4

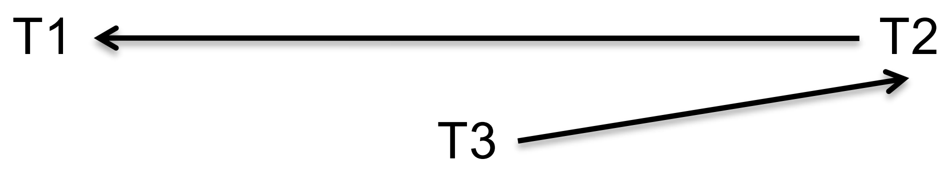
4. Is S a 2PL schedule with shared and exclusive locks?

Yes, by anticipating the exclusive lock on B by transaction T2 and the shared lock on C by the same transaction!

sl1(A) r1(A) sl2(A) r2(A) sl3(B) r3(B) u3(B) xl2(B) sl2(C) u2(A) xl1(A) w1(A) r2(C) r2(B) w2(B) u2(C) u2(B) xl1(C) w1(C) u1(A) u1(C)

Solution to Exercise 1.2 and 1.1

2. The precedence graph of S is the following:



- Given that P(S) is acyclic, the schedule is conflict-serializable (as we already knew)
- The conflict-equivalent schedules are those corresponding to the only topological order of P(S), i.e.

  – T3 T2 T1: *r3(B), r2(A), r2(C), r2(B), w2(B), r1(A), w1(A), w1(C)*

1. From the theory of serializability we know that S is also view-serializable, and the serial schedule above is view-equivalent to S

Exercise 2

Consider the following sequence S of actions, and answer these questions:

1. Is the schedule S view-serializable? If so, provide a view-equivalent serial schedule
2. What is the precedence graph associated to S? Is the schedule S conflict-serializable? If so, describe all the conflict-equivalent serial schedules
3. Is the schedule S a 2PL schedule (with exclusive locks)?

S: *r2(A), r3(B), w1(A), r2(C), r2(D), w1(D)*

Solution to Exercise 2.3

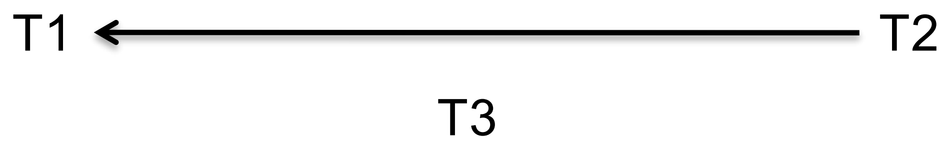
As in Exercise 1, let us first check whether S is 2PL

3. S can give rise to the following 2PL schedule with the commands for exclusive locks:

l2(A), r2(A), l3(B), r3(B), u3(B), l2(C), l2(D), u2(A), l1(A), w1(A), r2(C), u2(C), r2(D), u2(D), l1(D), w1(D), u1(A), u1(D)

Solution to Exercise 2.2 and 2.1

2. Given that S is 2PL, S is also conflict-serializable. The precedence graph of S is the following:



All conflict-equivalent schedules are those corresponding to possible topological order of P(S), i.e.

– T2 T1 T3: *r2(A), r2(C), r2(D), w1(A), w1(D), r3(B)*
– T3 T2 T1: *r3(B), r2(A), r2(C), r2(D), w1(A), w1(D)*
– T2 T3 T1: *r2(A), r2(C), r2(D), r3(B), w1(A), w1(D)*

1. Given that S is conflict-serializable, it is also view-serializable and all schedules above are also view-equivalent to S

Exercise 3

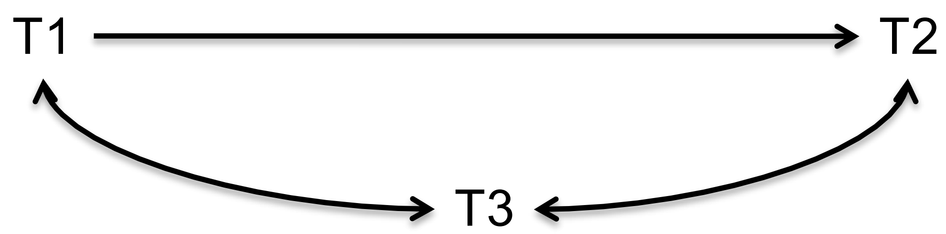
Consider the following sequence S of actions, and answer these questions:

1. Is the schedule S view-serializable? If so, provide a view-equivalent serial schedule
2. What is the precedence graph associated to S? Is the schedule S conflict-serializable?

S: *r3(B), w1(A), w3(B), r1(B), r2(A), w3(A), w2(A)*

Solution to Exercise 3

2. Let us first check whether S is conflict-serializable. The precedence graph of S is the following:



Since P(S) is cyclic, S is not conflict-serializable

1. Let us compute the READS-FROM and FINAL-WRITE sets:
   – READS-FROM={(r1(B),w3(B)), (r2(A),w1(A))}, FINAL-WRITE={w2(A), w3(B)}

A serial schedule that has the same READS-FROM and FINAL-WRITE sets is the following:

r3(B), w3(B), w3(A), w1(A), r1(B), r2(A), w2(A)

Hence, S is view-serializable!

Exercise 4

Consider the following sequence of actions, and tell whether

1. it is a view-serializable schedule or not,
2. it is a conflict-serializable schedule or not
3. it is a 2PL schedule (with shared and exclusive locks)

S: *r1(x), w2(x), r3(x), r1(y), r4(z), w2(y), r1(v), w3(v), r4(v), w4(y), w5(y), w5(z)*

Solution to Exercise 4.3

Let us first check whether S is 2PL

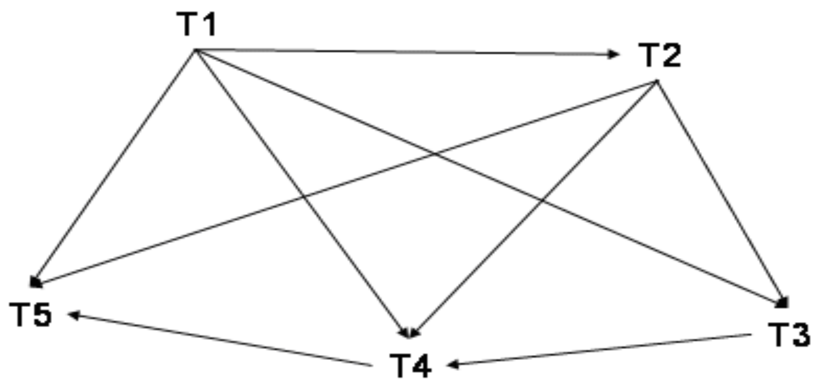
- In order for T3 to read x, it is necessary that T2 releases the exclusive lock on x (which was obtained by T2 for writing x)
- It follows that T2 has entered the shrinking phase when T3 reads x
- In order for T1 to read y, it is necessary that T1 has a shared lock on y
- It follows that, when T1 reads y, T2 cannot have the exclusive lock on y, and therefore T2 must request the exclusive lock for writing y after the reading of y by T1
- Therefore: T2 should request a lock during its shrinking phase

IMPOSSIBLE: S is not a 2PL schedule (with shared and exclusive locks)

Solution to Exercise 4.2 and 4.1

2. Let us now check whether S is conflict-serializable.

The precedence graph P(S):



The graph is acyclic, and therefore S is conflict-serializable. This is a serial schedule that is conflict-equivalent to S:

r1(x), r1(y), r1(v), w2(x), w2(y), r3(x), w3(v), r4(z), r4(v), w4(y), w5(y), w5(z)

1. Obviously, S is also view-serializable!

Exercise 5

Consider the following schedule, and tell whether it is a 2PL (with shared and exclusive locks) schedule or not

$$r_1(X)\ r_2(Z)\ r_1(Z)\ r_3(X)\ r_3(Y)\ w_1(X)\ w_3(Y)\ r_2(Y)\ w_4(Z)\ w_2(Y)$$

Solution to Exercise 5

it is a 2PL (with shared and exclusive locks) schedule, as shown as follows:

sl1(X) r_1*(X) sl2(Z)* r_2*(Z) sl1(Z)* r_1*(Z) sl3(X)* r_3*(X) sl3(Y)* r_3*(Y)*
xl3(Y) u3(X) xl1(X) w_1*(X)* w_3*(Y) u3(Y) sl2(Y)* r_2*(Y) xl2(Y)*
u2(Z) u1(Z) xl4(Z) w_4*(Z)* w_2*(Y) u1(X) u4(Z) u3(X) u2(Y)*

Exercise 6

Given a schedule S on transactions {T1, . . . , Tn}, the "strong graph" associated to S is a graph that has one node for each transaction Th $\in$ {T1,...,Tn}, and one edge from Ti to Tj, with i ≠ j, for each pair of actions ⟨ai(X),aj(X)⟩ on the same element X such that ai(X) belongs to Ti, aj(X) belongs to Tj, and ai appears before aj in S. Prove or disprove the following claims:

- If the strong graph associated to S is acyclic, then S is conflict serializable
- If S is conflict serializable, then the strong graph associated to S is acyclic

Solution to Exercise 6

It is immediate to see that the strong graph associated to a schedule S is a superset of the precedence graph of S.

It follows that:

1. If the strong graph associated to S is acyclic, then the precedence graph associated to S is also acyclic, and therefore S is conflict serializable
2. However, S can be conflict serializable even when the strong graph associated to S is acyclic, as shown by the following example:

r1(X) r2(X) w2(Y) r1(Y)

Exercise 7

Prove or disprove the following statement: if the schedule S creates a deadlock situation when processed by a 2PL scheduler, then S is not conflict serializable.

Solution to Exercise 7

If the schedule S creates a deadlock situation when processed by a 2PL scheduler, then there is a cycle in the wait-for graph. We will prove that this implies that there is a cycle in the precedence graph P(S) associated to S.

An edge from T1 to T2 in the wait-for graph of S means that

1. there is an action a1(X) of T1 in S requiring a lock on X,
2. T2 has the lock on X, and
3. at least one of the two locks is exclusive.

Case 1: T2 has an exclusive lock on x because it has a write on X → there is an edge from T2 to T1 in P(S)

Case 2: T2 has a shared lock on x because it has a read action on X → a1(X) is a write action, and there is an edge from T2 to T1 in P(S).

Solution to Exercise 7

The above considerations show that, for each edge in the wait-for graph associated to S there is a “reverse” edge in P(S).

Therefore, the presence of a cycle in the wait-for graph associated to S implies the presence of a cycle in P(S), which in turn implies that S is not conflict-serialilzable.

In other words, we have proved that if the schedule S creates a deadlock situation when processed by a 2PL scheduler, then S is not conflict serializable.

Esercise 8

A schedule S on transactions T1, . . . , Tn is called soft if (i) the commit command ci of every transaction in {T1, . . . , Tn} appears in S, (ii) each read action in S reads only from transactions that have already committed, and (iii) no write action in S writes on another transaction in S (i.e., comes after another write action of a different transaction on the same element). Prove or disprove that every soft schedule is a 2PL schedule with both shared and exclusive locks.

Solution to Exercise 8

- We remind the reader that a 2PL schedule (with shared and exclusive locks) is a legal schedule with shared and exclusive locks constituted by well-formed transactions following the 2PL protocol.
- The question reduces to check whether there is a soft schedule that is not in 2PL, i.e., whether there is a schedule with a typical no-2PL pattern that is soft. Now the typical no-2PL pattern is when a transaction that must release a lock cannot anticipate another lock without blocking a different transaction. Is it possible to create such a situation and still be coherent with the notion of soft schedule?
- The answer is yes. Indeed, consider the following schedule:

  S = r1(x) w2(x) w3(y) c3 c2 r1(y) c1

  It is immediate to verify that S is a soft schedule. However, S is not in 2PL, because transaction T1 must release the lock on x and cannot acquire the lock on y before entering the shrinking phase without blocking T3.
- So, we have disproved the claim by showing a soft schedule that is not in 2PL (with shared and exclusive locks).