MODIS Hand-In 4 Mandatory Exercises

Dennis Thinh Tan Nguyen, Jakob Holm, Jacob Mullit Mniche,
Pernille Lous, Thor Valentin Aakjr Olesen, William Diedrichsen Marstrand

25. november 2015

Indhold

1 Assignment 15.1

2 Assignment 15.3

3 Assignment 15.4

4 Assignment 15.6

5 Assignment 15.22

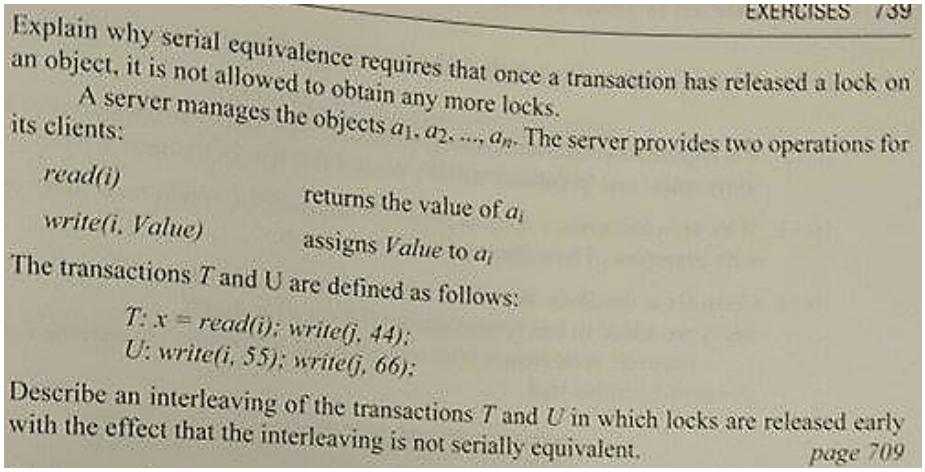
6 Assignment 15.23

7 Assignment 16.2

8 Assignment 16.3

9 Assignment 16.8

Assignment Description



EXERCISES 739

Explain why serial equivalence requires that once a transaction has released a lock on an object, it is not allowed to obtain any more locks.

A server manages the objects $a_1, a_2, \ldots, a_n$. The server provides two operations for its clients:

read(i) returns the value of a_i

write(i, Value) assigns *Value* to a_i

The transactions T and U are defined as follows:

T: x = read(i); write(j, 44);
U: write(i, 55); write(j, 66);

Describe an interleaving of the transactions T and U in which locks are released early with the effect that the interleaving is not serially equivalent.

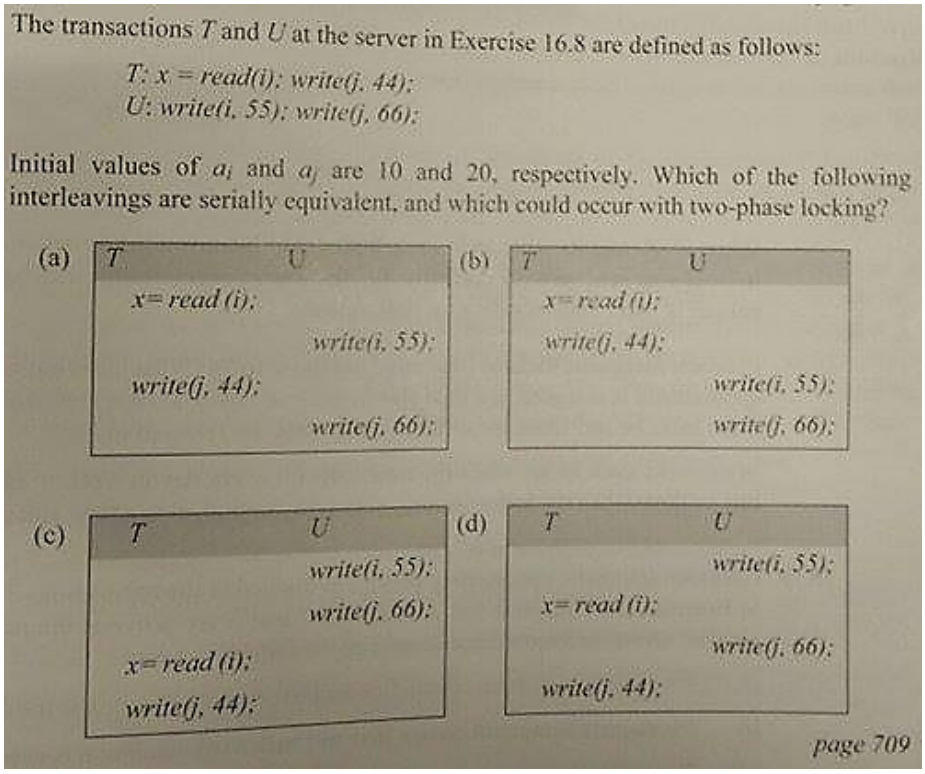
page 709

The reason why serial equivalence requires that once a transaction has released a lock on an object, it is not allowed to obtain any more locks is the following:
If a transaction locks an object after already having released it once, other transactions could potentially try to access and manipulate the object. This could result in the transaction ending up with a wrong result e.g. if a bank transaction is not serial equivalent, there could be to much money or to little in an account after the transaction has ended.

A non serial equivalent interleaving of the transactions T and U could be:
U: write(i,55) T: x = read(i) T: write(j,44) U: write(j,66)

10 Assignment 16.9

Assignment Description



The transactions T and U at the server in Exercise 16.8 are defined as follows:

T: x = read(i); write(j, 44);
U: write(i, 55); write(j, 66);

Initial values of a_i and a_j are 10 and 20, respectively. Which of the following interleavings are serially equivalent, and which could occur with two-phase locking?

(a)

T	U
x= read (i);	
	write(i, 55);
write(j, 44);	
	write(j, 66);

(b)

T	U
x= read (i);	
write(j, 44);	
	write(i, 55);
	write(j, 66);

(c)

T	U
	write(i, 55);
	write(j, 66);
x= read (i);	
write(j, 44);	

(d)

T	U
	write(i, 55);
x= read (i);	
	write(j, 66);
write(j, 44);	

page 709

a) and b) are serially equivalent, because the write operations on i and on j are equivalent to writing j and i since the operations happen on different objects and will not interfere. c) and d) are serially equivalent, because reading i and writing j in d) is equivalent to writing to j and reading i as in c), since the action happens on different objects, which do not interfere with each other.

The 2-phase-locking protocol states that a transaction must handle its locks in two distinct, consecutive phases during the transaction's execution:

1. **Expanding phase**' (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).

2. **Shrinking phase**: locks are released and no locks are acquired.

therefore only b) and c) could occur with two-phase locking. As an example of why a) or b) could not occur one could look at a, where T needs to acquire a lock to read i, then it needs to release the lock for U to use it, but then another

lock is acquired for T to make a write to j. This is not allowed according to the 2pl protocol.

11 Assignment 16.16

12 Assignment 16.18