CHAPTER 17: INTRODUCTION TO TRANSACTION PROCESSING CONCEPTS AND THEORY

17.14 Change transaction T 2 in Figure 17.2b to read:
read_item(X);
X:= X+M;
if X > 90 then exit
else write_item(X);
Discuss the final result of the different schedules in Figure 17.3 where M = 2 and N = 2, with respect to the following questions.

- Does adding the above condition change the final outcome?
- Does the outcome obey the implied consistency rule (that the capacity of X is 90)?

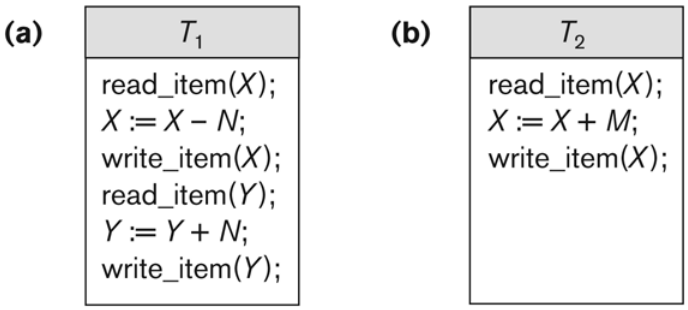


Figure 17.2
Two sample transactions.
(a) Transaction T_1.
(b) Transaction T_2.

Answer:

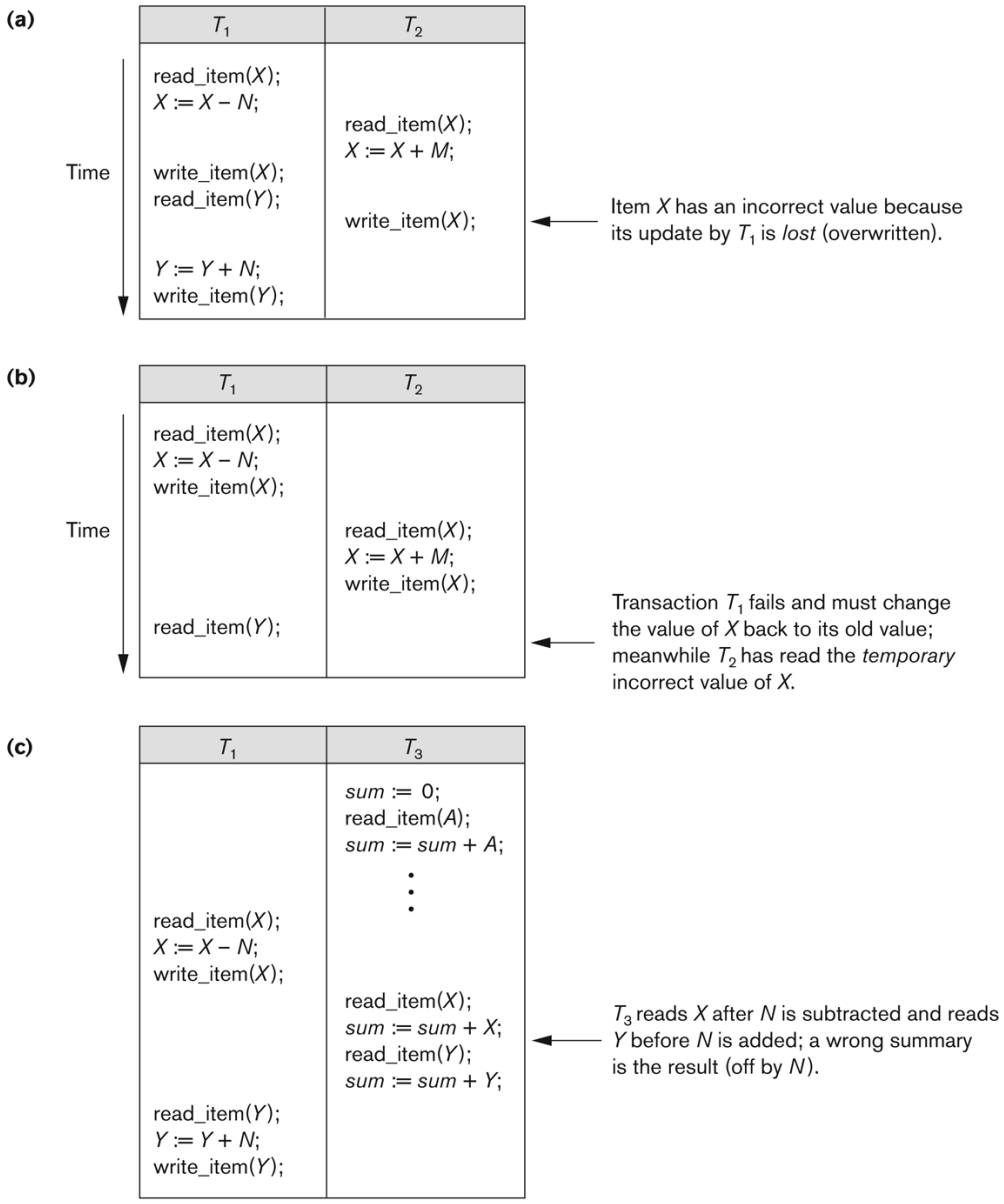
The above condition does not change the final outcome unless the initial value of X > 88. The outcome, however, does obey the implied consistency rule that X < 90, since the value of X is not updated if it becomes greater than 90.

17.15 Repeat Exercise 17.14 adding a check in T 1 so that Y does not exceed 90.

Answer:

T1 T2
read_item(X);
X := X-N;
read_item(X);
X := X+M;
write_item(X);
read_item(Y);
if X > 90 then exit
else write_item(X);
Y := Y+N;
if Y> 90 then
exit
else write_item(Y);
The above condition does not change the final outcome unless the initial value of X > 88 or Y > 88. The outcome obeys the implied consistency rule that X < 90 and Y < 90.

Figure 17.3
Some problems that occur when concurrent execution is uncontrolled. (a) The lost update problem. (b) The temporary update problem. (c) The incorrect summary problem.



17.16 Add the operation commit at the end of each of the transactions T 1 and T 2 from Figure 17.2; then list all possible schedules for the modified transactions.

17.17 List all possible schedules for transactions T 1 and T 2 from figure 17.2, and determine which are conflict serializable (correct) and which are not.

17.18 How many serial schedules exist for the three transactions in Figure 17.8 (a)? What are they? What is the total number of possible schedules?

Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(a)

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

17.22 Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

(a) r1 (X); r3 (X); w1(X); r2(X); w3(X)

(b) r1 (X); r3 (X); w3(X); w1(X); r2(X)

(c) r3 (X); r2 (X); w3(X); r1(X); w1(X)

(d) r3 (X); r2 (X); r1(X); w3(X); w1(X)

.

17.23 Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below. Draw the serializibility (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).
T1: r1(x); r1(z); w1(x)
T2: r2(z); r2(y); w2(z); w2(y)
T3: r3(x); r3(y); w3(y)
S1: r1(x); r2(z); r1(x); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y)
S2: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y)

18.24 Prove that cautious waiting avoids deadlock.
18.25 Apply the timestamp ordering algorithm to the schedules of Figure 17.8(b) and (c), and determine whether the algorithm will allow the execution of the schedules.

18.26 Repeat Exercise 18.25, but use the multiversion timestamp ordering method

Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(a)

Transaction T_1	Transaction T_2	Transaction T_3
read_item(X);	read_item(Z);	read_item(Y);
write_item(X);	read_item(Y);	read_item(Z);
read_item(Y);	write_item(Y);	write_item(Y);
write_item(Y);	read_item(X);	write_item(Z);
	write_item(X);	

(b)

Time

Transaction T_1	Transaction T_2	Transaction T_3
	read_item(Z);	
	read_item(Y);	
	write_item(Y);	
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(X);	
read_item(Y);		
write_item(Y);		
	write_item(X);	

Schedule E

(c)

Time

Transaction T_1	Transaction T_2	Transaction T_3
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(Z);	
read_item(Y);		
write_item(Y);		
	read_item(Y);	
	write_item(Y);	
	read_item(X);	
	write_item(X);	

Schedule F