answer— 23·1 1605023

There is an account A under organization 1 and another account B under organization 2.

a. Show all the steps to transfer Tk. 5000 from account A to account B using persistent messaging protocol.

b. Discuss all types of failures and atomicity issues.

Solution

a Considering no failure during the entire process, the steps to complete transaction using persistent messaging protocol are as follows :

i. Transaction is initiated by account A and account A performs database update by withdrawing/subtracting Tk. 5000 from the account.

ii. A transaction message is written and inserted in messages_to_send relation with a unique id.

iii. Message Delivery processor at site A generates the inserted message and sends it to site B.

iv. At site B, upon receiving message, message receiving processor executes transaction to add corresponding message to received_messages relation with a unique id. Assigning each message with unique id is a must to ensure that message for a specific transaction is written only once.

v. Account B processes the received message, updates database by depositing/adding Tk. 5000 to its account, and marks the message as processed.

vi. Transaction is committed at site B.

vii. Acknowledgement message is sent to site A from site B.

viii. At site A, upon receiving the acknowledgement, the corresponding transaction message in messages_to_send relation is deleted.

ix. Site A commits the transaction.

Thus, atomicity is maintained throughout the whole process.

b Failures and Atomicity Issues

i. If destination account B does not exist or is unreachable, failure message must be sent back to source account A.

ii. If sending transaction aborts, message must not be sent to site B.

iii. When transaction is aborted at site A or site A receives failure message, tk. 5000 must be deposited back in source account A. This process may require hooman interrention if account A is found to be closed.

Ans.

answer-24.1

1605023

Put appropriate lock and show lock status.

Solution

T1	T2	T3	T4	lock Status
LOCK-S(P)				GRANT(P,T1)
READ(P)				
	LOCK-X(Q)			GRANT(Q,T2)
	WRITE(Q)			
		LOCK-S(P)		GRANT(P,T3)
		READ(P)		
			LOCK-X(P)	WAIT(P,T4)
			WRITE(P)	

Ans.

answer – 24.2 1605023

a. given A = 100 and B = 100. Prove that the above concurrent schedule preserves database consistency.

b. Explain conflict serializability of given schedule 2.

Solution

a If the value of A+B is preserved after the execution of transactions T_1 and T_2, then we can say that given concurrent schedule preserves DB consistency.

Here, at the beginning: A = 100 and B = 100.

i) After first block of operation on A by T_1: $A = A-50 = 50$, $B = 100$.

ii) After first block of operation on A by T_2: $A = A - 0.1 \cdot A = 45$, $B = 100$.

Also, we have $temp = A \cdot 0.1 = 5$ in T_2 now.

iii) After second block of opt. on B by T_1: $A = 45$, $B = B+50 = 150$.

T_1 commits at the end of this block of opt.

iv) After second block of opt. on B by T_2: $A = 45$, $B = B + temp = 155$.

T_2 commits at the end of this block of opt.

So, $A_{before} + B_{before} = 100 + 100 = 200$.

$A_{after} + B_{after} = 45 + 155 = 200$.

Thus, DB consistency is preserved.

b A conflict serializable schedule is a non-serial schedule which can be made a serial one by swapping non-conflicting instructions among participating transactions.

Here, schedule 1 is a serial schedule because T_2 starts after T_1 transaction's execution and commitment.

On the other hand, schedule 2 is not a serial schedule for its concurrent nature. But, schedule 2 is equivalent to serial schedule 1 as both of them yield same output after execution and preserve DB consistency.

Now, we can make the following changes in schedule 2 without introducing any conflict:

T1	T2

read(A) B1
A:=A−50
write(A)	

swap1	B2
	read(A)
	temp:=A×0·1
	A:=A−temp
	write(A)

swap2	

B3
read(B)
B:=B+50
write(B)
commit	

	B4
	read(B)
	B:=B+temp
	write(B)
	commit

⇒

T1	T2

read(A) B1
A:=A−50
write(A)	

B3
read(B)
B:=B+50
write(B)
commit	

	B2
	read(A)
	temp:=A×0·1
	A:=A−temp
	write(A)

	B4
	read(B)
	B:=B+temp
	write(B)
	commit

Here, swap1 introduces conflict as read and write on same resource A.
swap2 does not introduce conflict as read and write on two different resources A and B.

Thus, we can convert schedule 2 to schedule 1 by swapping block2 and block3 in execution timeline.

Hence, schedule 2 is conflict serializable as schedule 1 is a serial one.

Ans.