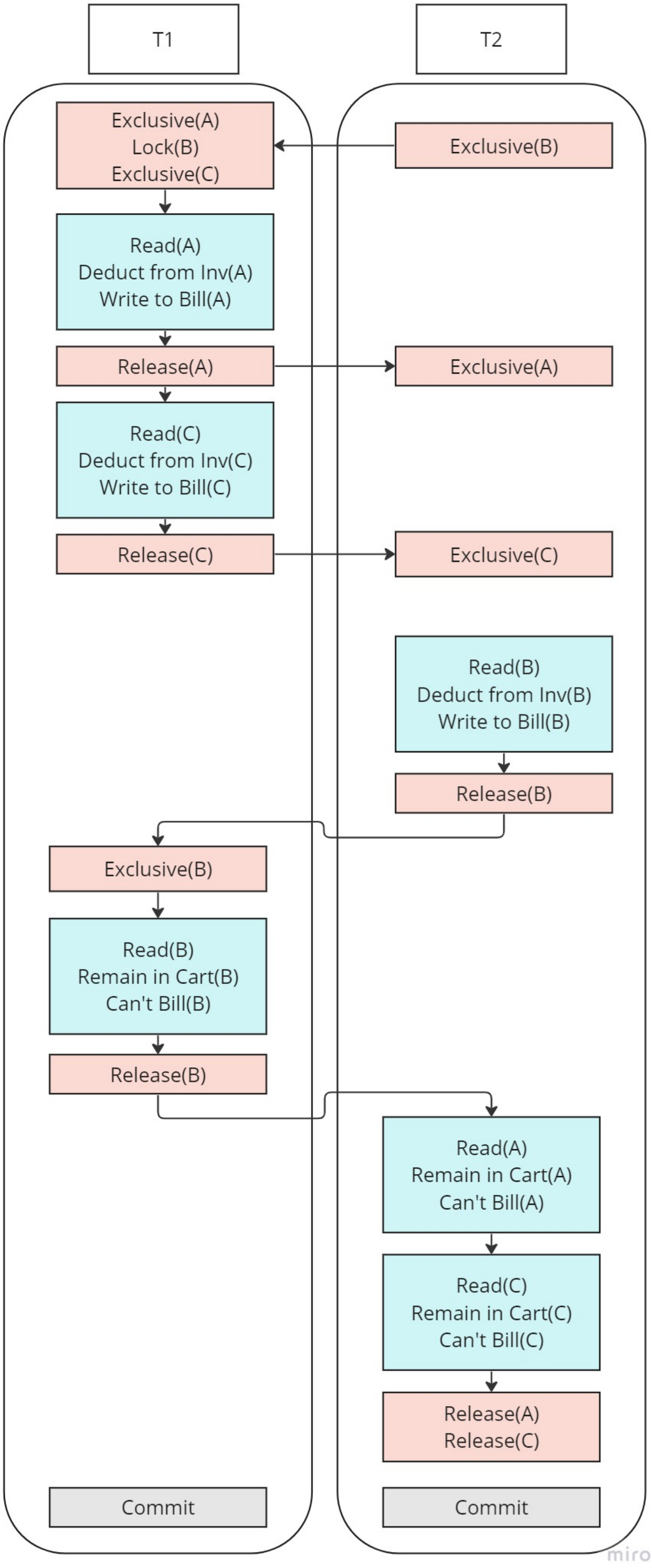
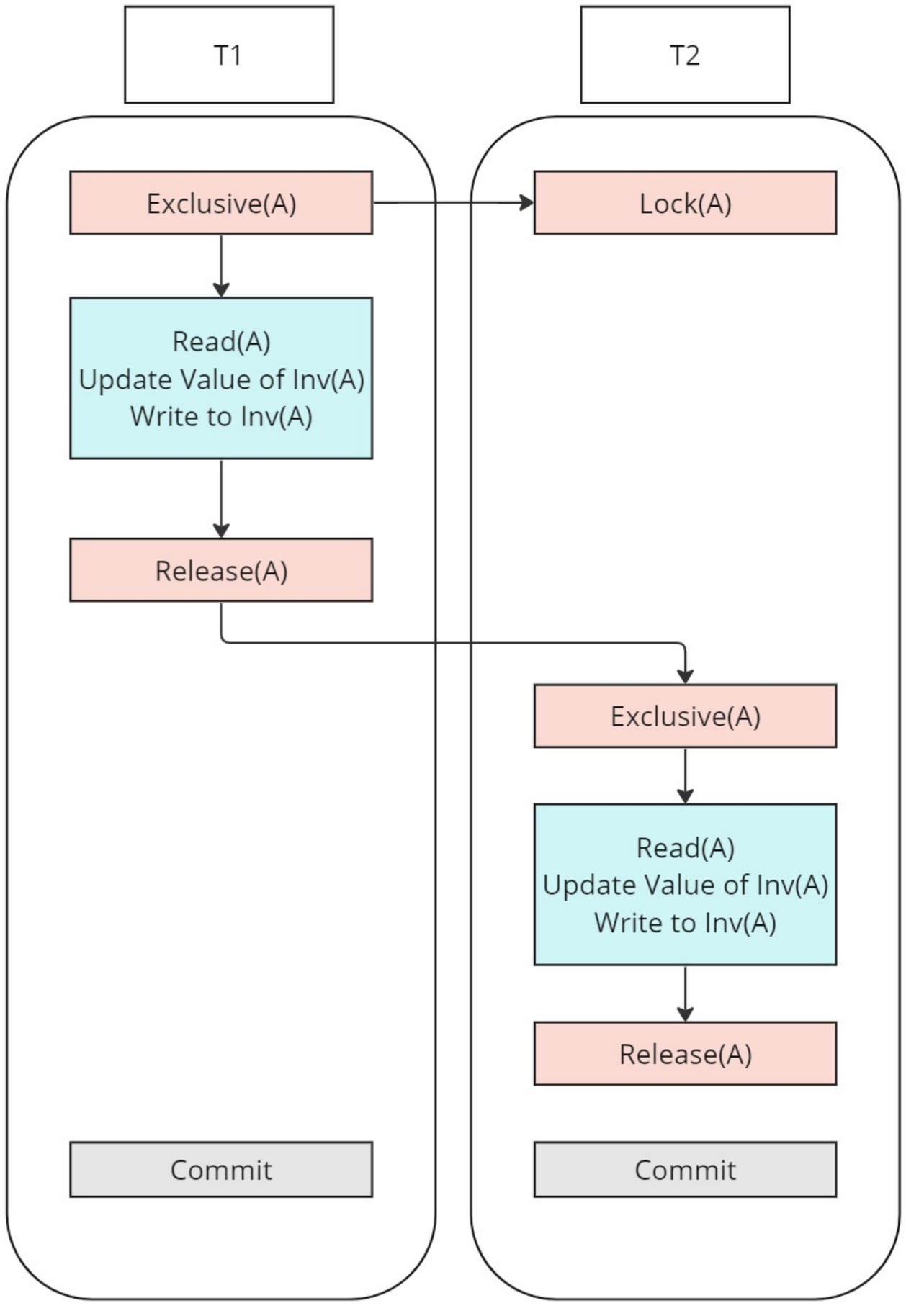
Problem Description CT1

2 Customers place an order at once but 3 items in their carts exist that the quantity in the inventory is just enough for one of them to be able to place each item. It is predetermined that Customer 1 gets A and B, Customer 2 gets C

Allowed relaxation in protocol so that locks can be released before new lock requests





Problem Description CT2

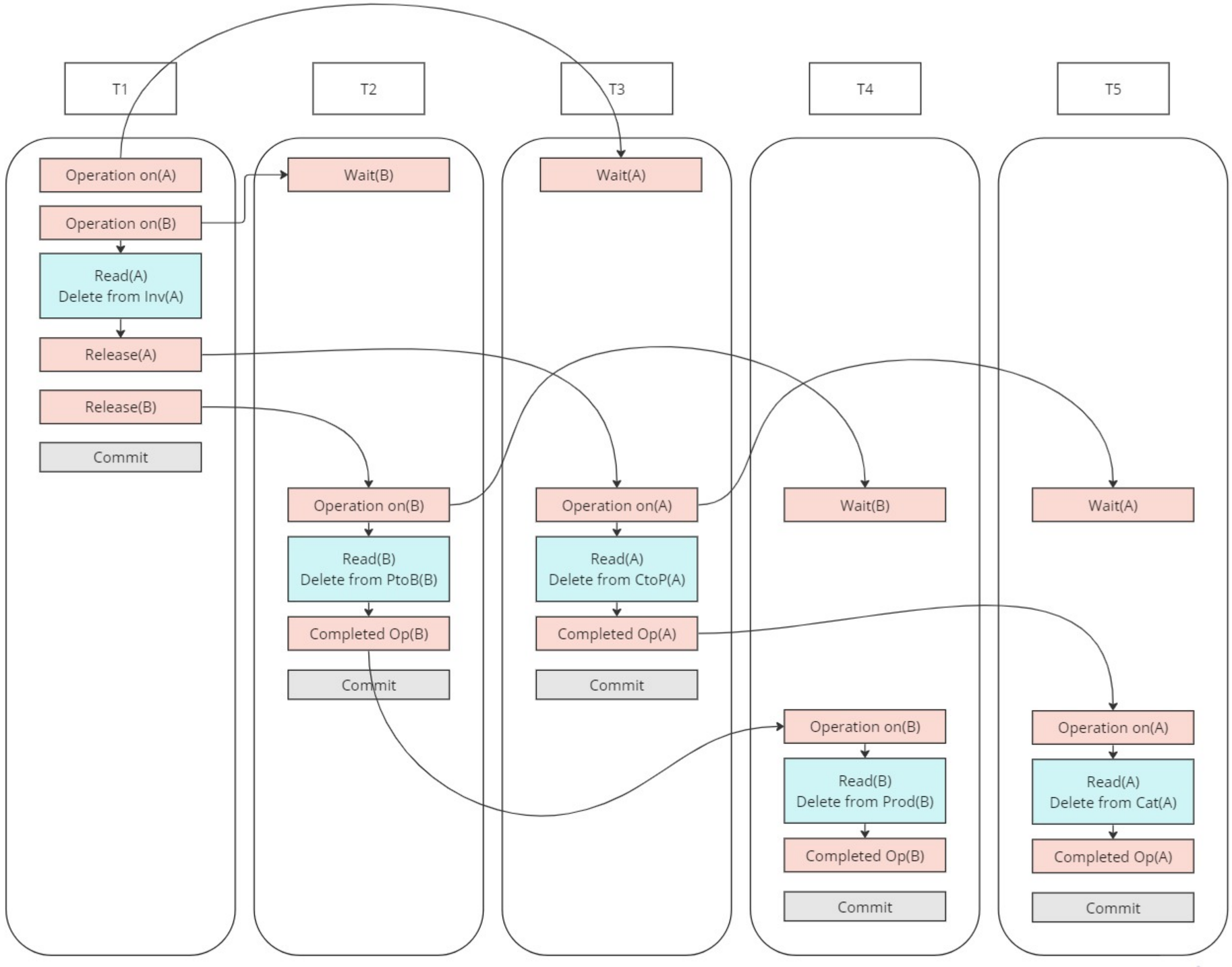
A customer tries to place an order for an item A in his cart but simultaneously admin attempts to change price of item A

Follows 2 Phase Locking Method. To apply strictly simply move commit of T1 above before release

Problem Description NCT2

Transaction involves removing a category A from the store, thus entries that feature this category will have to be removed. Also products(Say 1 called B) of this category A will also have to be removed from their respective tables

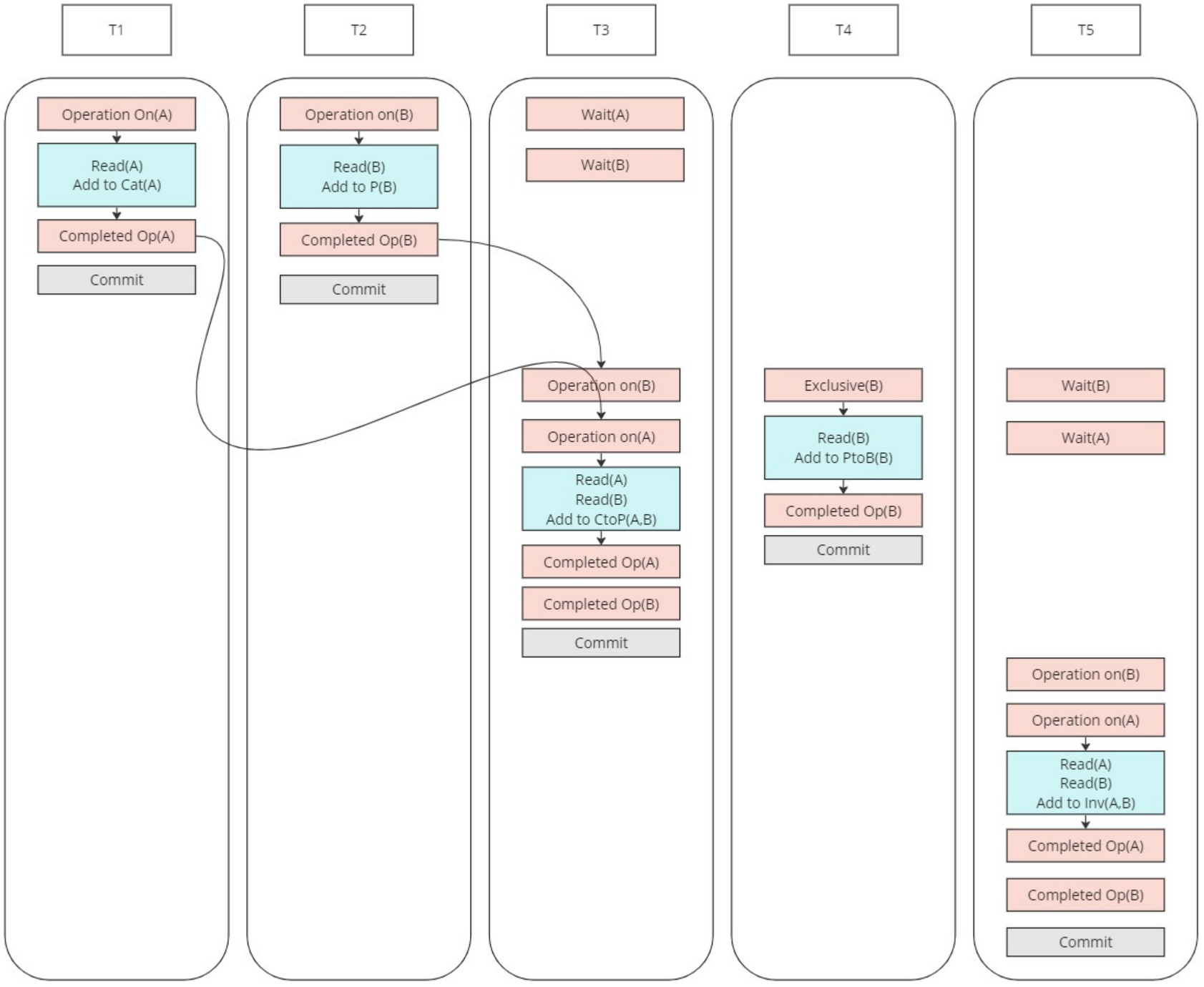
For Non Concurrent operations reason is that we need information of the previous table



Problem Description NCT1

Transaction involves adding a Product B from the store, whose brand exists but its category A is a new one.
Thus first we need to take of adding category first.

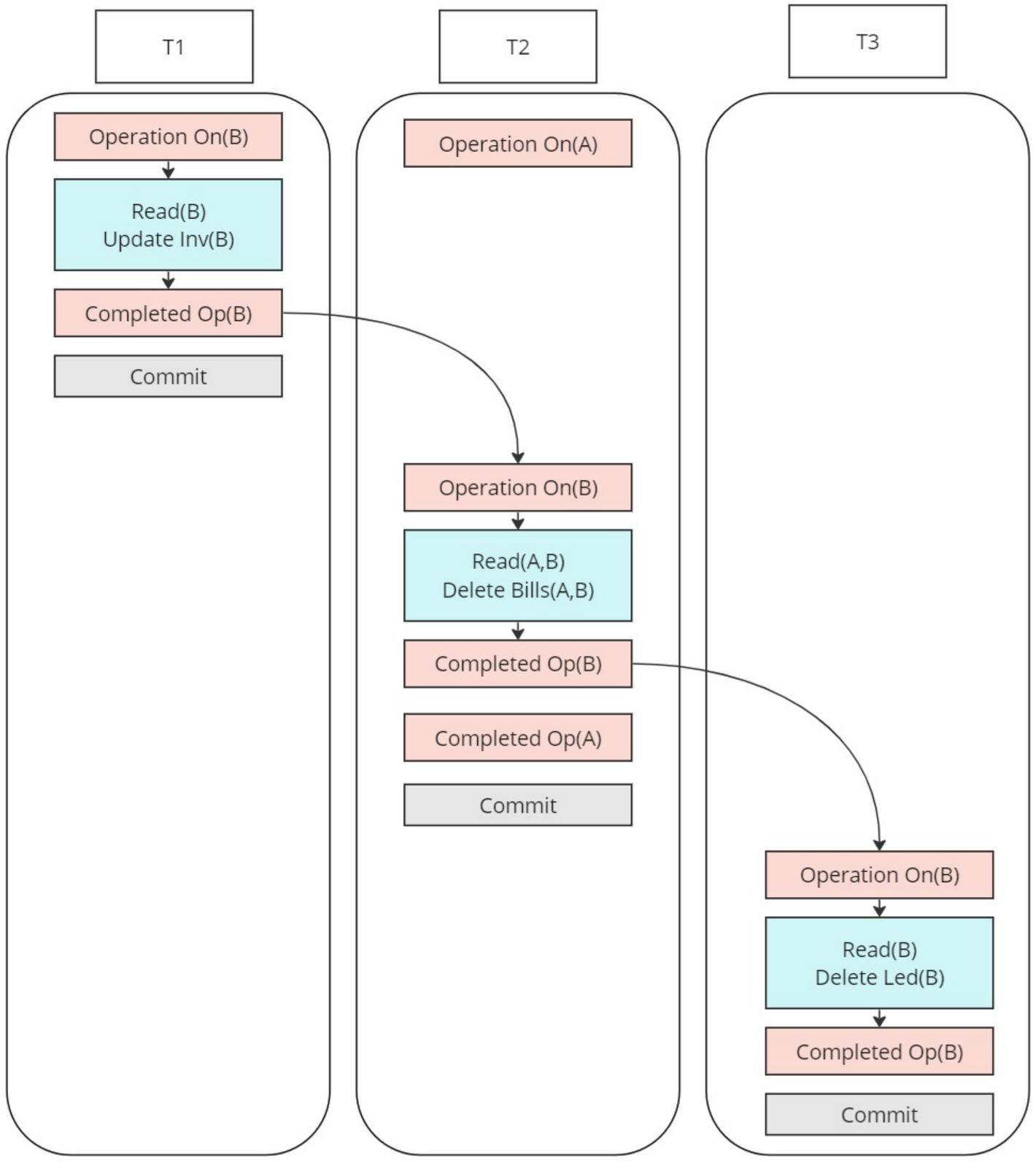
For Non Concurrent operations reason is that we need information of the previous table



Problem Description NCT3

Cancelling an order completely. It not only involves deleting data(A) item from ledger but also from the bills table. In addition we also have to update the quantity of every product(B) of bill(A) in the inventory

For Non Concurrent operations reason is that we need information of the previous table



Problem Description NCT4

Implementing a minimum order amount requirement.
As soon as bill(A) is generated then it's total is calculated adding each item(B) and if it exceeds the limit then it is committed else it is rolled-back. Here the rollback situation is described

For Non Concurrent operations reason is that we need information of the previous table