Transaction Pair I

Let's suppose there are two customers buying the same product which is a laptop at the same time.

Variable Notations used:-
Q represents the quantity of Laptop.
q1 represents the quantity of laptops which customer A purchases.
q2 represents the quantity of laptop which customer B purchases.
B1 represents the balance of the customer A.
B2 represents the balance of the customer B.
B1' represents the balance of customer A after purchasing the product.
B2' represents the balance of customer B after purchasing the product.
O1 represents the Order of the customer A.
O2 represents the Order of the customer B.

Transaction-1 (T-1)	Transaction-2 (T-2)
Read(Q) Q = Q - q1 Write(Q) Read(B1) Write(B1') Commit	Read(Q) Q = Q − q2 Write(Q) Read(B2) Write(B2') Commit

A Serial Schedule of the two transactions

Transaction-1 (T-1)	Transaction-2 (T-2)
Read(Q) Q = Q - q1 Write(Q) Read(B1) Write(B1') Commit	Read(Q) Q = Q − q2 Write(Q) Read(B2) Write(B2') Commit

Conflict-Serializable Schedule(isme conflict nhi h)

Transaction-1 (T-1)	Transaction-2 (T-2)
	Read(Q) Q = Q − q2 Write(Q) Read(B2) Write(B2') Commit
Read(Q) Q = Q - q1 Write(Q) Read(B1) Write(B1') Write(Order O1) Commit	 Write(Order O2)

This schedule is serializable since we can swap the first three statement of t1 i.e: Read(Q), Q = Q - q1, Write(Q) with statement write (Order O2) of t2 which makes it a serial schedule with order as t2 executing be

Non-Conflict Serializable Schedule

Transaction-1 (T-1)	Transaction-2 (T-2)
Read(Q) Q = Q - q1 Write(Q) Read(B1) Write(B1') Commit	 Read(Q) Q = Q − q2 Write(Q) Read(B2) Write(B2') Commit

This schedule is non-conflict serializable as there exists a loop in the precedence graph between T-2 and T-1 on data item Q.
This is because T1 writes to Q before and after a read and write, respectively, which are executed by T2.

The below is the SQL format of the above transactions.

```
-- Transaction T1: Customer A purchasing a laptop and headphones
START TRANSACTION;

-- Step 1: Inserting a new shopping order for Customer A
// write
INSERT INTO ShoppingOrder (orderId, order_date, delivery_address, customer_id)
VALUES (12, NOW(), '123 Main St', 1); -- Assuming customer_id 1 is Customer A

-- Step 2: Updating inventory for the laptop and headphones
UPDATE Inventory
// doing read operation on product quantity Q
SET product_quantity = product_quantity - 1
// write operation on product quantity Q
WHERE product_id IN (101, 102); -- Assuming product_id 101 is the laptop and 102 is
the headphones

-- Step 3: Deleting items from Customer A's shopping cart
// reading customer balance B1
DELETE FROM ShoppingCart
WHERE customer_id = 1; -- Assuming customer_id 1 is Customer A
// write B1
COMMIT;
```

```
-- Transaction T2: Customer B purchasing the same laptop and a TV
START TRANSACTION;

-- Step 1: Inserting a new shopping order for Customer B
// write
INSERT INTO ShoppingOrder (orderId, order_date, delivery_address, customer_id)
VALUES (13, NOW(), '456 Elm St', 2); -- Assuming customer_id 2 is Customer B

-- Step 2: Updating inventory for the laptop and TV
UPDATE Inventory
// doing read operation on product quantity Q
SET product_quantity = product_quantity - 1
// write operation on product quantity Q
WHERE product_id IN (101, 103); -- Assuming product_id 101 is the laptop and 103 is
the TV
-- Step 3: Deleting items from Customer B's shopping cart
// reading customer balance B2
DELETE FROM ShoppingCart
// reading customer balance B2
WHERE customer_id = 2; -- Assuming customer_id 2 is Customer B
//write B2

COMMIT;
```

Non-Conflict Serializable Schedule With Locks

Transaction-1 (T-1)	Transaction-2 (T-2)
Lock-S(Q) Read(Q) UnLock(S) Lock-X(Q) Q = Q - q1 Write(Q) Unlock(Q) Read(B1) Write(B1') Commit	 Lock-S(Q) Read(Q) UnLock(Q) Lock-X(Q) Q = Q − q2 Write(Q) Unlock(Q) Lock-X(B2) Read(B2) Write(B2') Unlock(B2) Commit

In this schedule, we add shared locks to each of the data items to avoid dirty read. And exclusive locks around every pair of write&read. Adding locks makes sure that no two transactions simultaneously manipulate the same data item.