## Conflicting transactions

Suppose we have decided that after the sale of 10 milk packets, we will start giving 10% discount on milk.

Transaction T1: A customer orders 10 milk packets from the app
Transaction T2: Another customer orders 5 milk packets from the app at the same time.

Let A be the no. of packets of milk sold and B be the price of each packet. Assume A is initially equal to zero. Ideally, if T2 occurs after T1 , no conflict shall occur but as they both are occuring potentially at the same time, it is leading to conflict whether in T2, person will get discount or not .

We have to make 2 schedules of this conflicting transaction- one conflicting serializable and other non- conflicting serializable.

Let read be represented by R and write be represented by W.

### Conflicting serializable schedule

| T1 | T2 |
|---|---|
| r(A) | |
| r(B) | |
| w(A) | |
| | r(A) |
| w(B) | |
| | r(B) |
| | w(B) |
| commit | w(A)<br>commit |

We can make this schedule serializable by replacing w(B) in T1 by r(A) in T2.
Making this schedule serializable, we get :

| T1 | T2 |
|---|---|
| r(A) | |
| r(B) | |
| w(A) | |
| w(B) | |
| | r(A) |
| | r(B) |
| | w(B) |
| commit | w(A)<br>commit |

<u>Non-conflicting serializable schedule for these transactions</u>

| T1 | T2 |
|---|---|
| r(A) | |
| r(B) | |
| w(A) | |
| | r(A) |
| | r(B) |
| w(B) | |
| | w(B) |
| commit | w(A)<br>commit |

In this, we cannot make a serializable schedule for these transactions, even after swapping.

So, to resolve conflict in such a type of transaction, we have to use lock mechanism in this.

| T1<br>lock(B)<br>lock(A) | T2<br>lock(B)<br>lock(A) |
|---|---|
| r(A) | |
| r(B) | |
| w(A) | |
| | r(A) |
| | r(B) |
| w(B)<br>commit<br>unlock(B)<br>unlock(A) | |
| | w(B) |
| | w(A)<br>unlock(A)<br>unlock(B)<br>commit |