# DBMS Deadline 6

1) **Transactions that we are using:**

T1 : While checking-out a cart, for each cart-item A, we are reducing the availability of the product( i.e stock) by the number of items that has been bought by that customer while checking out.

T1: **START TRANSACTION;**
**SELECT ci.product_id INTO productID, ci.quantity INTO quantity**
**FROM cart_item ci**
**WHERE ci.cart_id=c.cart_id;** -- 'c' is for cart
/*This statement finds the quantity and product_id of an cart_item present in that customer's cart and then stores them into two variables 'productID' which stores the product_id of that product and 'quantity' which stores the quantity of that product*/

**UPDATE Product P**
**SET P.stock=P.stock-quantity**
**WHERE P.product_id=productID;**
/*This statement finds the product with that product_id and deducts the quantity(the total number of items brought by the customer) from the total available stock of that product*/

**COMMIT;**

T2: This transaction is for someone who is viewing that product and after finding it available, he adds that product into his cart

Here for simplicity, we have assumed three variables 'desired_product_id' which stores the desired product_id of the product desired by the customer, 'desired_quantity' which stores the desired quantity of the product and 'current_customer_id' which stores the current customer's id

T2: **START TRANSACTION;**
**SELECT C.cart_id INTO current_cart_id**
**FROM customer C**
**WHERE C.customer_id=current_customer_id;**
/*This statement find the current customer's cart and stores them into variable 'current_cart_id'*/

**UPDATE cart_item**
**INSERT INTO cart_item (cart_id, product_id, quantity)**
**VALUES (current_cart_id, desired_product_id, desired_quantity)**
**WHERE product_id=desired_product_id AND ;**
/*This statement creates an entry for cart_item table using the values provided*/

**COMMIT;**

T3: This transaction is for checking coupon discount from the entered coupon code and then applying this to the cart total amount

T3:
```
START TRANSACTION;
SELECT C.percentage_discount
FROM coupon C
WHERE C.coupon_code = entered_coupon_code;
//save this in a variable discount

UPDATE cart C
SET C.cart_total_amount = C.cart_total_amount*(100-discount)/100
WHERE C.cart_id = customer.cart_id;

COMMIT;
```

**T1: { READ(quantity) from the cart_item, WRITE(quantity) to product, COMMIT}**
**T2: { READ(quantity) from product, WRITE(quantity) to cart_item, COMMIT }**
**T3: { READ(coupon_discount) from coupon, WRITE(cart_total_amt) to cart, COMMIT}**

**SERIAL SCHEDULE:**

| T1 | T2 | T3 |
|---|---|---|
| READ(quantity) from cart_item | | |
| WRITE(quantity) to product | | |
| COMMIT | | |
| | READ(quantity) from product | |
| | WRITE(quantity) to cart_item | |
| | COMMIT | |
| | | READ(coupon_discount) |
| | | WRITE(cart_total_amt) |
| | | COMMIT |

**CONFLICT SERIALIZABLE SCHEDULE:**

| T1 | T2 | T3 |
|---|---|---|
| READ(quantity) from cart_item | | |
| | | READ(coupon_discount) |
| WRITE(quantity) to product | | |
| COMMIT | | |
| | READ(quantity) from product | |
| | | WRITE(cart_total_amt) |
| | | COMMIT |
| | WRITE(quantity) to cart_item | |
| | COMMIT | |

**No swap is required between conflicting instructions** to make it equivalent to the above shown (first table) serial schedule so **it is a conflict serializable schedule.**

Serializable: As it produces the same effect as the serial schedule

**NON-CONFLICT SERIALIZABLE SCHEDULE:**

| T1 | T2 | T3 |
|---|---|---|
| READ(quantity) from cart_item | | |
| | | READ(coupon_discount) |
| | READ(quantity) from product | |
| WRITE(quantity) to product | | |
| | WRITE(quantity) to cart_item | |
| | | WRITE(cart_total_amt) |
| | | COMMIT |
| | COMMIT | |
| COMMIT | | |

Such **swaps would be necessarily required that would be between conflicting instructions** (read and write on same data object 'quantity' in T1 and T2 shown above) to make it equivalent to the above shown (first table) serial schedule so it is **not a conflict serializable schedule.**

Conflict: read write of same data object quantity by T1 and T2

**RESOLUTION OF THE CONFLICT USING LOCKING SYSTEM:**

| T1 | T2 | T3 |
|---|---|---|
| READ(quantity) from cart_item | | |
| | lockS(cart_item) | |
| | | READ(coupon_discount) |
| | READ(quantity) from product | |
| WRITE(quantity) to product | | |
| | WRITE(quantity) to | |

| | | |
|---|---|---|
| | **cart_item** | |
| | **unlockS(cart_item)** | |
| | | **WRITE(cart_total_amt)** |
| | | **COMMIT** |
| | **COMMIT** | |
| **COMMIT** | | |