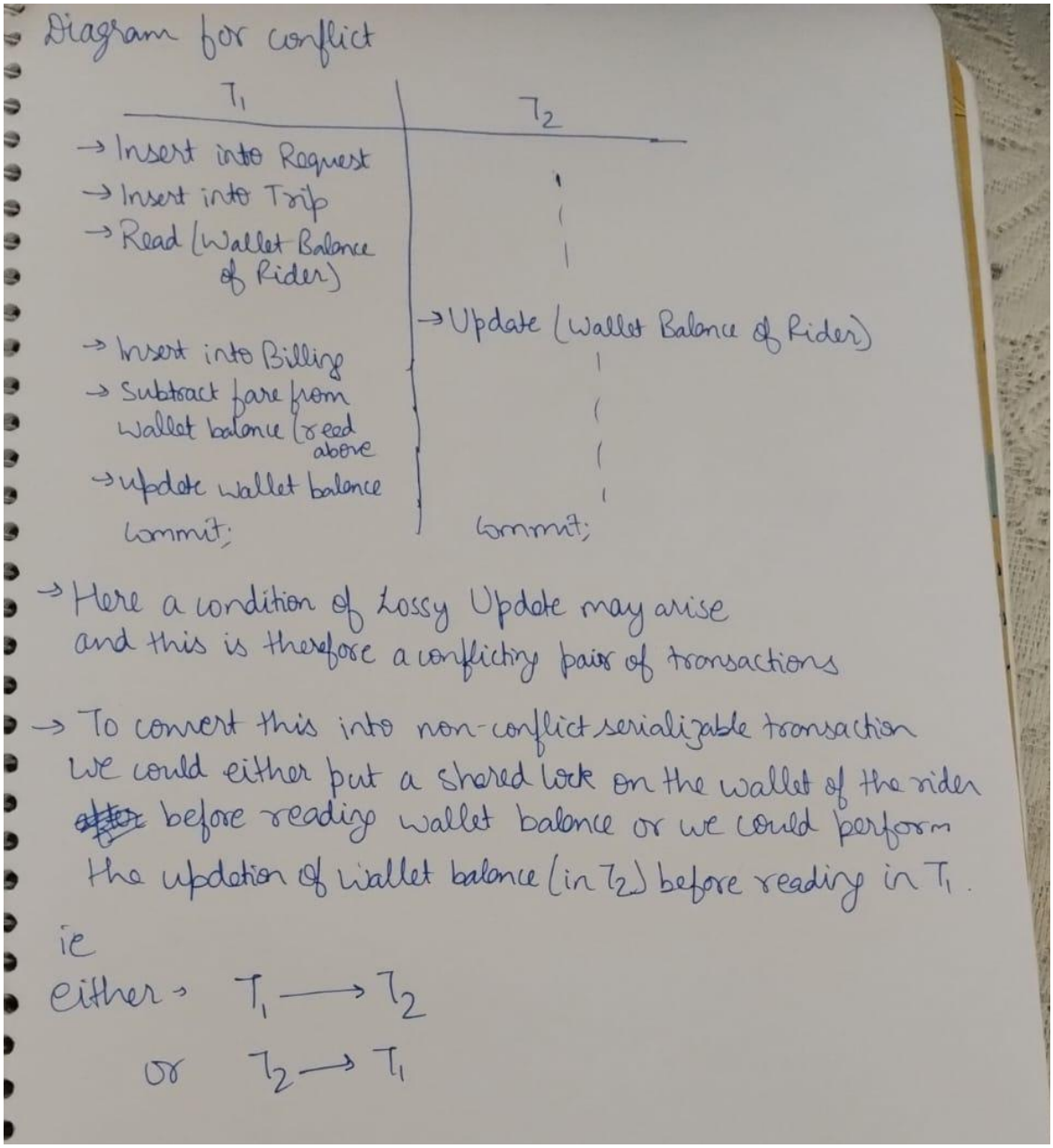
Following is the transaction 1 query

```
START TRANSACTION;

SET @wallet_id = NULL;
SET @balance = NULL;

SET @wallet_id = (SELECT Wallet_id FROM rider WHERE Rider_Id = '199');

SET @request_id = ((SELECT MAX(Request_ID) FROM request) + 1);

INSERT INTO request (Request_ID, Rider_Id, preference, Preffered_Vehicle,
Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y, Status) VALUES
(@request_id, '199', 1, 'sedan', 0, 0, 1, 1, 1);

SET @driver_id = (SELECT d.driver_id FROM driver d JOIN vehicle v ON d.Vehicle_id =
v.vehicle_id WHERE d.Status = 1 AND v.type = 'sedan' ORDER BY d.Current_Rating DESC
LIMIT 1);

SET @p_x = (SELECT Pickup_Location_x FROM request WHERE Request_ID =
@request_id);
SET @p_y = (SELECT Pickup_Location_y FROM request WHERE Request_ID =
@request_id);
SET @d_x = (SELECT Drop_Location_x FROM request WHERE Request_ID = @request_id);
SET @d_y = (SELECT Drop_Location_y FROM request WHERE Request_ID = @request_id);

SET @distance = SQRT(POW(@p_x - @d_x, 2) + POW(@p_y - @d_y, 2));
SET @rate = (SELECT rate FROM driver WHERE driver_id = @driver_id);
SET @fare = @distance * @rate;

INSERT INTO trip (Request_ID, Driver_id, Fare, Date_Time, Distance, Rating) VALUES
(@request_id, @driver_id, @fare, NOW(), @distance, 4);

SET @balance = (SELECT amount FROM wallet WHERE wallet_id = @wallet_id);

IF (@fare <= @balance) THEN
   INSERT INTO billing (Request_ID, Driver_ID, Rider_ID, Date_TIME, Distance, Fare,
Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y) VALUES
(@request_id, @driver_id, '199', NOW(), @distance, @fare, @p_x, @p_y, @d_x, @d_y);
   SET @balance = @balance - @fare;
   UPDATE wallet SET amount = @balance WHERE wallet_id = @wallet_id;
   COMMIT;
ELSE
   ROLLBACK;
END IF;
```

Following is the transaction 2 query:

```
START TRANSACTION;
 UPDATE wallet SET amount = amount + 1000   WHERE Wallet_id = @wallet_id;
COMMIT;
```



To make it serializable we can either put a share lock on the wallet in transaction 1 which is done in the following code

```
START TRANSACTION;
SET @wallet_id = NULL;
SET @balance = NULL;
SELECT Wallet_id FROM rider WHERE Rider_Id = '199' LOCK IN SHARE MODE;
SET @wallet_id = (SELECT Wallet_id FROM rider WHERE Rider_Id = '199');
SELECT MAX(Request_ID) FROM request;
SET @request_id = ((SELECT MAX(Request_ID) FROM request) + 1);
INSERT INTO request (Request_ID, Rider_Id, preference, Preffered_Vehicle,
Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y, Status) VALUES
(@request_id, '199', 1, 'sedan', 0, 0, 1, 1, 1);
SET @driver_id = (SELECT d.driver_id FROM driver d JOIN vehicle v ON d.Vehicle_id =
v.vehicle_id WHERE d.Status = 1 AND v.type = 'sedan' ORDER BY d.Current_Rating DESC
LIMIT 1);
SELECT Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y FROM
request WHERE Request_ID = @request_id LOCK IN SHARE MODE;
SET @p_x = (SELECT Pickup_Location_x FROM request WHERE Request_ID =
@request_id);
SET @p_y = (SELECT Pickup_Location_y FROM request WHERE Request_ID =
@request_id);
SET @d_x = (SELECT Drop_Location_x FROM request WHERE Request_ID = @request_id);
SET @d_y = (SELECT Drop_Location_y FROM request WHERE Request_ID = @request_id);
SET @distance = SQRT(POW(@p_x - @d_x, 2) + POW(@p_y - @d_y, 2));
SELECT rate FROM driver WHERE driver_id = @driver_id LOCK IN SHARE MODE;
SET @fare = @distance * @rate;
INSERT INTO trip (Request_ID, Driver_id, Fare, Date_Time, Distance, Rating) VALUES
(@request_id, @driver_id, @fare, NOW(), @distance, 4) LOCK IN EXCLUSIVE MODE;
SELECT amount FROM wallet WHERE wallet_id = @wallet_id FOR UPDATE;
SET @balance = (SELECT amount FROM wallet WHERE wallet_id = @wallet_id);
IF (@fare <= @balance) THEN
   INSERT INTO billing (Request_ID, Driver_ID, Rider_ID, Date_TIME, Distance, Fare,
Pickup_Location_x, Pickup_Location_y, Drop_Location_x, Drop_Location_y) VALUES
(@request_id, @driver_id, '199', NOW(), @distance, @fare, @p_x, @p_y, @d_x, @d_y)LOCK
IN EXCLUSIVE MODE;;
   SET @balance = @balance - @fare;
   UPDATE wallet SET amount = @balance WHERE wallet_id = @wallet_id;
   COMMIT;
ELSE
   ROLLBACK;
END IF;
```

Or we could run the transaction 2 first and then the transaction 1 in that way there would be no condition of lossy update.