Seminar : Concurrency Control

There are several problems that are found when concurrent transaction execute in an uncontrolled manner.

Let's take an example of a simplified airline reservations database in which a record is stored for each airline flight.

* Each record includes the No of Reserved seats on that flight assigned (uniquely identifiable) data item, among other information

T1
read_item(X)
X = X - N
write_item(X)
read_item(Y)
Y = Y + N
write_item(Y)

T1, transaction that transfers N reservations from one flight whose No of Reserved seats is stored in db item named X to another flight whose no of Reserved seats is stored in db item named Y.

T2.
read_item(X)
X = X + M
write_item(Y)

T2 transaction that just reserves M seats on the first flight (X)

So when a db access program is written it has a flight number, flight date and no of seats to be booked as parameters
hence the same program can be used to execute many different transactions, each with a different flight number, flight date & No of seats to be booked

For concurrency control purpose, a transaction is a particular execution of a program on a specific date, flight and number of seats.

T1 and T2 are specific execution of the program that refer to specific flights whose no of seats are stored in data item X and Y.

When these 2 simple transaction execute concurrently, we may encounter many problems

1. The Lost Update Problem :-

When 2 transaction that access the same db items have their operations interleaved such that makes value of some db items incorrect.

If X = 80 at start
N = 5
X = 80 - 5 = 75
M = 4
X = 75 + 4 = 79
should be the final result

However in the interleaving of operation in T1 and T2
X = 80
X = 80 + 4 = 84 coz the update in T1 that removed 5 seats was lost.

Suppose transactions T1, T2 submitted at approximately same time and suppose operations are interleaved, the final value of X is incorrect, as T2 reads the value of X before T1 changes in db, hence updated value resulting from T1 is lost.

| T1 | T2 | |
|---|---|---|
| read_item (x) | | |
| X := X - N; | | |
| | read_item (x); | |
| | X := X + M; | |
| write_item (x); | | |
| read_item (y) | | |
| | write_item (x); | ← Item X has an incorrect value because its update by T1 is lost (overwritten) |
| Y := Y + N; | | |
| write_item (y); | | |

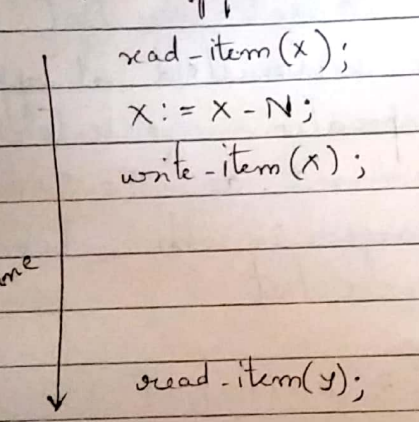
Time ↓

2. The Temporary Update (or Dirty Read) Problem.

This occurs when 1 transaction updates a db item and then transaction fails for some reason.
Meanwhile, the updated item is accessed (read) by another transaction before it is changed/rolled back to its original value

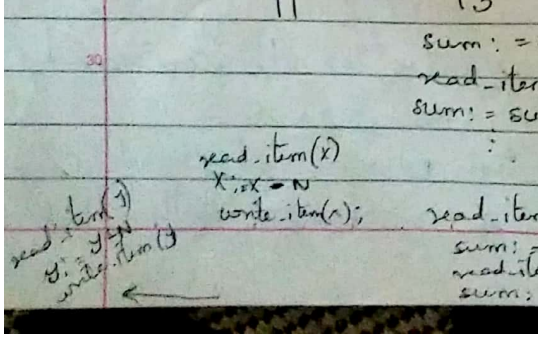
This value of db item is called as dirty data since it has been created by a transaction that has not completed & committed yet. hence the problem is Refered as dirty read problem

Eg: $T_1$ updates item X and fails before completion at such case, the system rolls back changes. But before it can do so, transaction $T_2$ reads temporary/updated value of X which will not be recorded permanently in db item as $T_1$ failed. The value of item X read by $T_2$ is called dirty data




3. The Incorrect Summary Problem: If 1 transaction is calculating an aggregate summary function on a number of database items while other transactions are updating some of these items, the aggregate function may calculate some values before they are updated and others after they are updated.

Eg: If $T_3$, transaction calculates total No of reservation on all the flight, meanwhile if $T_1$ is executing If interleaving of operation occurs as shown, the result




T3 reads X after N is subtracted and reads Y before N is added & a wrong summary is the result

The Unrepeatable Read Problem

If a Transaction T reads the same item twice & the item is changed by another transaction T' between the 2 reads. Hence T receives different values for its 2 reads at the same time.

Eg: If during an airline reservation transaction a customer inquires seat availability on different several flights.

When customer decides on a particular flight, the transaction then reads the No of seats on flight a second time before completing reservation and it may end up reading a different value for the item.