Concurrent Transactions

Concurrent Transactions

Motivation

- The main objective of developing database is to enable many users share data concurrently.

- Concurrent access is relatively easy if all users are only reading data; however when users are updating data, there may be interference (conflicts) that can result to inconsistencies

Concurrent Transactions Definitions:

Concurrency

- In computer science, concurrency is a property of systems in which several computations are executing simultaneously and potentially interacting with each other.

Concurrency control

- It's the process of managing simultaneous operations on the data base without having the interfere with each another.

Concurrent Transactions

Purpose of Concurrency Control

– To Enforce isolation among conflicting transactions

– To Preserve database consistency

– To resolve conflicts among conflicting transactions

Concurrent Transactions

Conflicting Operations

- Two operations on the same data item in different transactions conflict if one of them is a write.
- If we denote database read and write operations on a data item x as r(x) and w(x), then we say the operations conflict:
  1. If the operations belong to different transactions
     - One transaction can not conflict
  2. If the operations must access the same data item and
  3. At least one of the operation should be write

Concurrent Transactions

Conflicting Operations

- Two operations on the same data item conflict if at least one of the operations is a write:
  - r(x) and w(x) conflict
  - w(x) and r(x) conflict
  - w(x) and w(x) conflict
  - r(x) and r(x) do not
  - r/w(x) and r/w(y) do not
- Order of conflicting operations matters
  - If T1 .r(a) precedes T2 .w(a), then conceptually, T1 should precede T2

Concurrent Transactions

Conflicting Operations~example

Consider these two different sequences of execution :

- T1

  A=A+100

  B=B~100

  COMMIT

- T2

  A=A*1.05

  B=B*1.05

  COMMIT

~~▶ T1~~

~~A+100~~

~~▶ T2~~

~~A=A*1.05~~
~~B=B*1.05~~
~~COMMIT~~

~~▶ T1~~

~~B=B-100~~
~~COMMIT~~

INTERFERENCE

A = 210 DH	**A = 210 DH**
B = 0 DH	**B = 5 DH**

Concurrent Transactions

Examples of Interference Between Concurrent Transactions

- There are many ways in which concurrently executing transactions can interfere with one another and so compromise the integrity and consistency of the DB.
- Three examples of such interference are:
  - Lost Update Problem
  - Uncommitted Dependency Problem
  - Inconsistency Analysis Problem

Concurrent Transactions

The Lost Update Problem

- An apparently successfully completed update operation by one user can be overridden by another user. This is known as the Lost Update Problem.

Concurrent Transactions

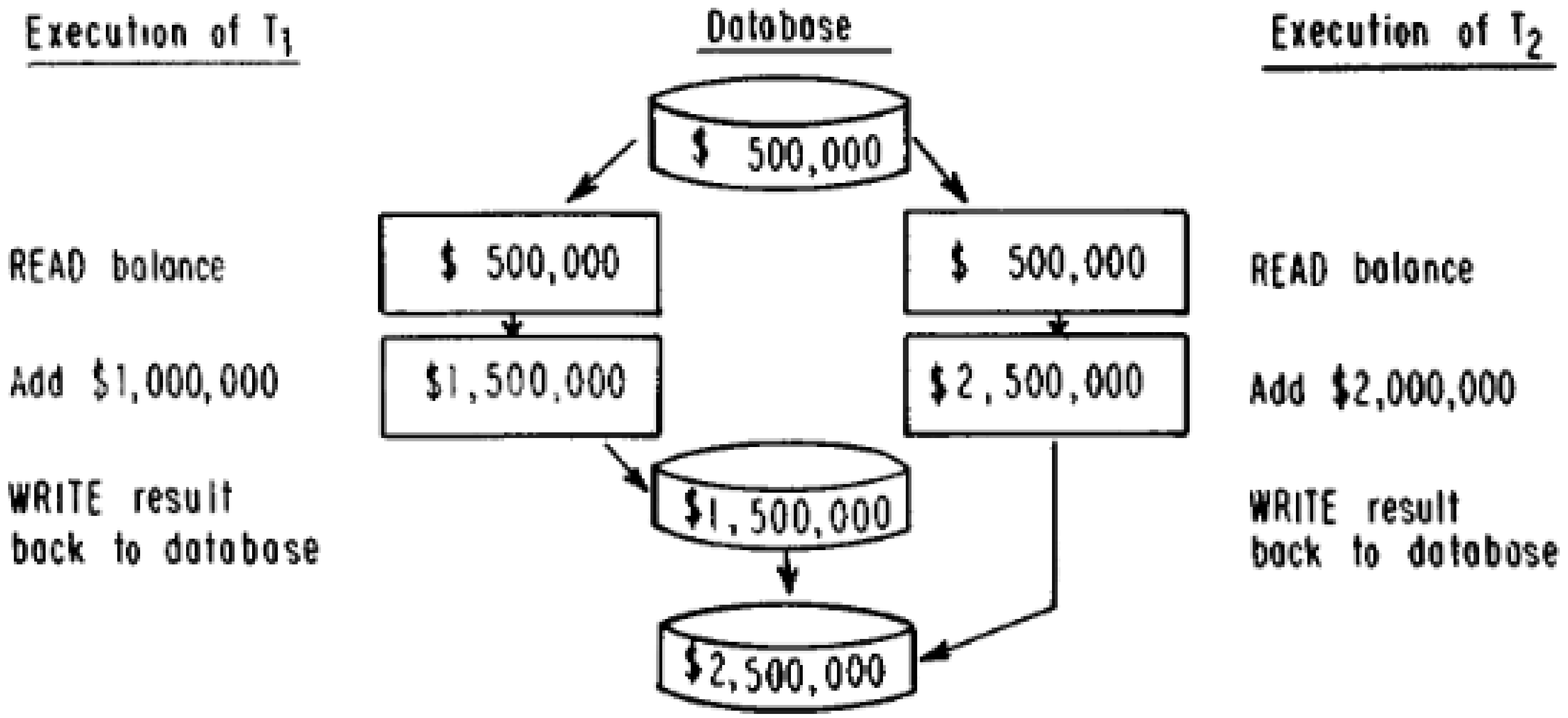
The Lost Update Problem

Example I: Suppose two customers simultaneously try to deposit money into the same account. The two ATMs handling the two customers could read the account balance at approximately the same time, compute new balances in parallel, and then store the new balances back into the database.

Take Note that: In response to customer requests, ATMs retrieve data from a database, perform computations, and store results back into the database.

Concurrent Transactions

The Lost Update Problem



Concurrent Transactions

The Lost Update Problem

Example I

Time	Transaction A	Transaction B
t1	Read (R)	~
t2	~	Read (R)
t3	Write (R)	~
t4	~	Write (R)

- Transaction A's update is lost at time t4 because transaction B overwrites it without even looking at it!

Concurrent Transactions

The Lost Update Problem

Example II: Transaction T_1 is executed concurrently with transaction T_2.

Time	T_1	T_2	Bal_x
t_1	-	begin_transaction	100
t_2	begin_transaction	read(bal_x)	100
t_3	read(bal_x)	$bal_x = bal_x + 100$	100
t_4	$bal_x = bal_x - 10$	write(bal_x)	200
t_5	write(bal_x)	Commmit	90
t_6	Commmit	-	90

- Transaction T_1 updates the same record updated earlier by T_2 at time t_5 on the basis of values read at t_3 .
- Question: How do we solve the lost update problem?

Concurrent Transactions

Uncommitted Dependency Problem

- A violation of the integrity constraints governing the database can arise when two transactions are allowed to execute concurrently without being synchronized.

- Uncommitted dependency Problem occurs when one transaction is allowed to see the intermediate results of another transaction before it has committed.

Concurrent Transactions

Uncommitted Dependency Problem

Example I: Consider this situation:

Time	T_3	T_4	Bal_x
t_1	-	begin_transaction	100
t_2	-	read(bal_x)	100
t_3	-	bal_x= bal_x +100	100
t_4	begin_transaction	write(bal_x)	200
t_5	read(bal_x)	:	200
t_6	bal_x= bal_x -10	rollback	100
t_7	write(bal_x)	-	190
t_8	commmit	-	190

Question: How do we solve the uncommitted dependency problem?

Concurrent Transactions

Uncommitted Dependency Problem

- Transaction T_3 reads on uncommitted update at time $t_{5.}$
- The update is then undone at time t_6
- Transaction T_3 is therefore operating on false assumption i.e.Transaction T_3 becomes dependent on an uncommitted update at time T_2

Concurrent Transactions

Inconsistent Analysis Problem

- Transactions that only read the database can obtain inaccurate results if they are allowed to read partial results of incomplete transactions, which are simultaneously updating the database

- Problem of inconsistent analysis occurs when a transaction reads several values from the database but a second transaction updates some of them during the execution of the first.

Concurrent Transactions

Inconsistent Analysis Problem

- Consider two transactions A and B operating on bank account records
- Transaction B is summing account balances.
- Transaction A is transferring an amount 10 from account 1 to account 3
- Account 1 = 100; Account 2 = 50; Account 3 = 25

Concurrent Transactions

Inconsistent Analysis Problem

Time	Transaction A	Transaction B	Acc1	Acc2	Acc3	Sum
t_1	Read Acc1	Read Acc1	100	50	25	0
t_2	Acc1 - 10	Sum + Acc1	100	50	25	100
t_3	Write Acc1	Read Acc2	90	50	25	100
t_4	Read Acc3	Sum + Acc2	90	50	25	150
t_5	Acc3 + 10	~	90	50	25	150
t_6	Write Acc3	~	90	50	35	150
t_6	Commit	Read Acc3	90	50	35	150
t_7	~	Sum + Acc3	90	50	35	185
t_8	~	Commit	90	50	35	185

🖂 Sum = 185 not 175!!

Question: How do we solve the inconsistent analysis problem?

How to solve Interference Between Concurrent Transactions

- The objective of concurrency control protocol is to schedule transactions in such a way as to avoid any interference between them.
- When there are multiple transactions that are running in a concurrent manner and the order of operation is needed to be set so that the operations do not overlap each other, Scheduling is brought into play and the transactions are timed accordingly.
  - A quick solution is to allow only one transaction to be executed and committed before another transaction begins to execute.
  - However, the aim of multi-user DBMS is to maximize the degree of concurrency or parallelism in the system without interference among the transactions.

References

- https://www.geeksforgeeks.org/types-of-schedules-in-dbms/
- http://www.ccs.neu.edu/home/kathleen/classes/cs3200/9-Transactions.pdf