Transaction Processing- I

Transaction Processing Systems

- **Transaction Processing Systems** are the systems with large databases and hundreds of concurrent users executing database transactions.
- For example airline reservations, banking, stock markets, etc.

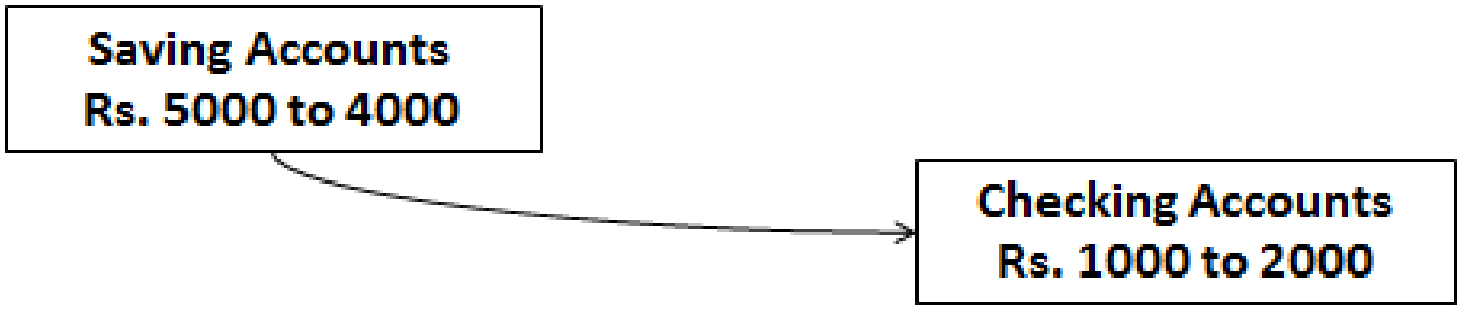
Transaction

- A transaction is an executing program that forms a logical unit of database processing.
- A transaction includes one or more database access operations- these can include insertion, deletion, modification, or retrieval operations.
- Transaction is executed as a single unit. It is a program unit whose execution may or may not change the contents of a database.

Example

- A transfer of money from one bank account to another requires two changes to the database both must succeed or fail together.
  - Subtracting the money from the savings account balance.
  - Adding the money to the checking account balance.

Example



Transaction

1. Subtract Rs. 1000 from Savings (Machine Crashes)
2. Add Rs. 1000 to Checking (Money Disappears)

Processes of Transaction

- Read Operation
- Write Operation

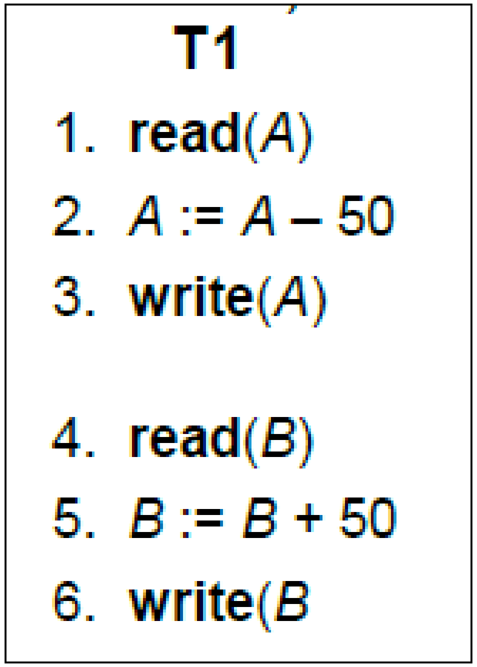
Desirable Properties of Transactions

ACID properties:

- **Atomicity**: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- **Consistency**: A correct execution of the transaction must take the database from one consistent state to another.
- **Isolation**: A transaction should appear as though it is being executed in isolation from other transactions, even if many transactions are executing concurrently. That is, the execution of a transaction should not be interfered with any other executing transactions.
- **Durability**: Once a transaction changes the database and the changes are committed, these changes must never be lost because of any failure.

Transaction Properties

- Let T1 be a transaction that transfers Rs 50 from account A to account B. This transaction can be defined as:



T1

```
1. read(A)
2. $A := A - 50$
3. write(A)

4. read(B)
5. $B := B + 50$
6. write(B
```

- Say value of A prior to transaction was 1000 and that of B was 2000

- **Atomicity:** Suppose that during the execution of T1, a power failure has occurred that prevented the T1 to complete successfully. The point of failure may be after the completion of Write(A) and before Write(B). It means that the changes in A are performed but not in B. Thus the values of account A and B are Rs.950 and Rs.2000 respectively. We have lost Rs.50 as a result of this failure.
- Now, our database is in inconsistent state.
- The reason for this inconsistent state is that our transaction is completed partially.
- In order to maintain atomicity of transaction, the database system keeps track of the old values of any write and if the transaction does not complete its execution, the old values are restored to make it appear as the transaction never executed.

- **Consistency:** The consistency requirement here is that the sum of A and B must be unchanged by the execution of the transaction. It can be verified easily that, if the database is consistent before an execution of the transaction, the database remains consistent after the execution of the transaction.
- Ensuring consistency for an individual transaction is the responsibility of the application programmer who codes the transaction.

- **Isolation:** If several transactions are executed concurrently (or in parallel), then each transaction must behave as if it was executed in isolation. It means that concurrent execution does not result an inconsistent state.
- For example, consider another transaction T2, which has to display the sum of account A and B. Then, its result should be Rs.3000.

- **Durability:** Once the execution of the transaction completes successfully, and the user who initiated the transaction has been notified that the transfer of funds has taken place, it must be the case that no system failure will result in a loss of data corresponding to this transfer of funds.

QUIZ

With regards to transaction processing, any DBMS should be capable of:

a. Ensuring that transactions are free from interference from other users.

b. Parts of a transaction are not lost due to a failure.

c. Transactions do not make the database inconsistent.

d. All of the above.

QUIZ

What is ACID properties of Transactions?

a. Atomicity, Consistency, Isolation, Database

b. Atomicity, Consistency, Isolation, Durability

c. Atomicity, Consistency, Inconsistent, Durability

d. Automatically, Concurrency, Isolation, Durability

QUIZ

Which of the following is not a property of a transaction?

a) Atomicity

b) Simplicity

c) Isolation

d) Durability

QUIZ

Collections of operations that form a single logical unit of work are called ___________

a) Views

b) Networks

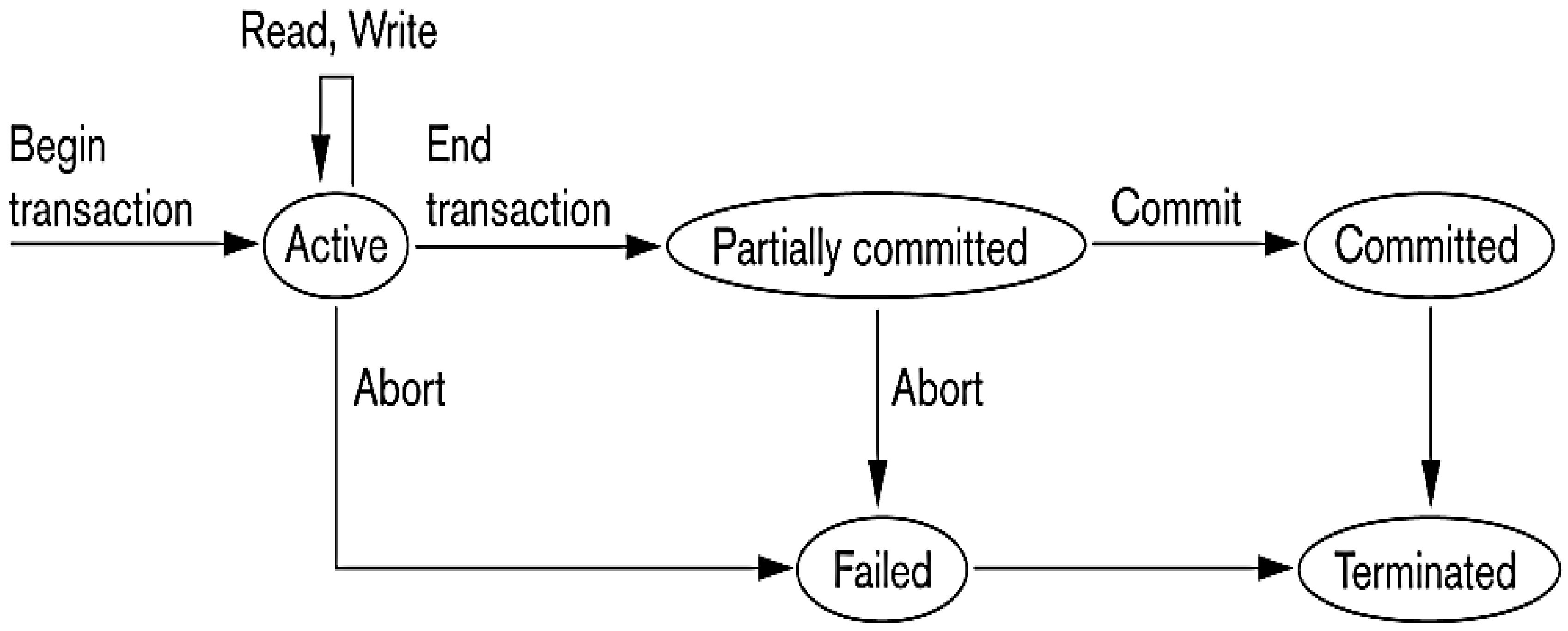
c) Units

d) Transactions

Transaction States

- **Active state–** the initial state; the transaction stays in this state while it is executing .
- **Partially committed state–** after the final statement has been executed.
- **Failed state --** after the discovery that normal execution can no longer proceed.
- **Aborted state–** after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted:
  - restart the transaction can be done only if no internal logical error
  - kill the transaction
- **Committed state –** after successful completion

State Transition Diagram



QUIZ

The “all-or-none” property is commonly referred to as

a) Isolation

b) Durability

c) Atomicity

d) None of the mentioned

QUIZ

Execution of transaction in isolation preserves the __________ of a database

a) Atomicity

b) Consistency

c) Durability

d) All of the mentioned

QUIZ

Which of the following is not a transaction state?

a) Active
b) Partially committed
c) Failed
d) Compensated

QUIZ

Consider money is transferred from (1)account-A to account-B and (2) account-B to account-A. Which of the following form a transaction?

a) Only 1

b) Only 2

c) Both 1 and 2 individually

d) Either 1 or 2

QUIZ

The database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements. This property is referred to as

a) Atomicity
b) Durability
c) Isolation
d) All of the mentioned

QUIZ

A transaction is delimited by statements (or function calls) of the form ___________

a) Begin transaction and end transaction

b) Start transaction and stop transaction

c) Get transaction and post transaction

d) Read transaction and write transaction

QUIZ

The property of a transaction that persists all the crashes is

a) Atomicity

b) Durability

c) Isolation

d) All of the mentioned

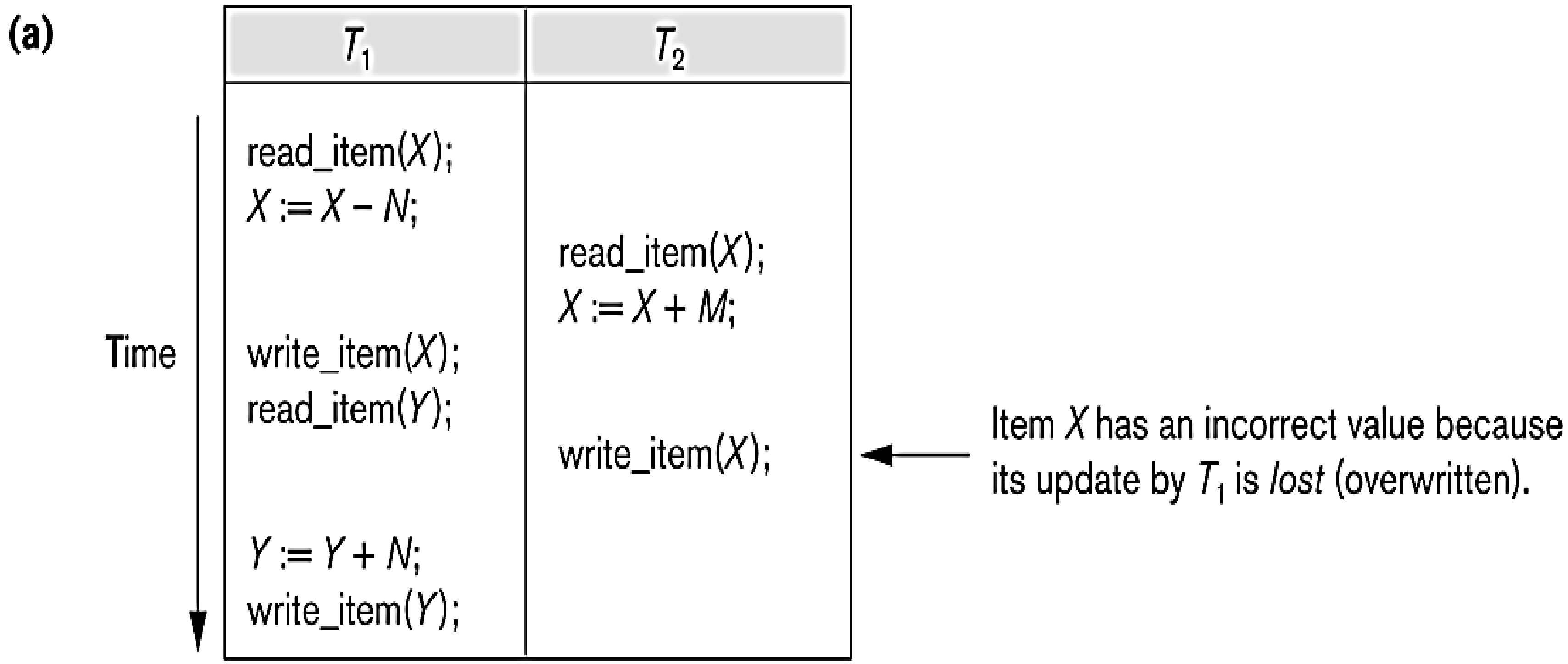
Concurrent Executions

- Multiple transactions are allowed to run concurrently in the system.
- Advantages are:
  - **increased processor and disk utilization**, leading to better transaction *throughput*
    - E.g. one transaction can be using the CPU while another is reading from or writing to the disk
  - **reduced average response time** for transactions: short transactions need not wait behind long ones.
- **Concurrency control schemes** – mechanisms to achieve isolation that is, to control the interaction among the concurrent transactions in order to prevent them from destroying the consistency of the database

Why Concurrency Control is needed?

- **The Lost Update Problem**
  - This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.
- **The Temporary Update (or Dirty Read) Problem**
  - This occurs when one transaction updates a database item and then the transaction fails for some reason
  - The updated item is accessed by another transaction before it is changed back to its original value.
- **The Incorrect Summary Problem**
  - If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.

Concurrent execution is uncontrolled: (a) The lost update problem.



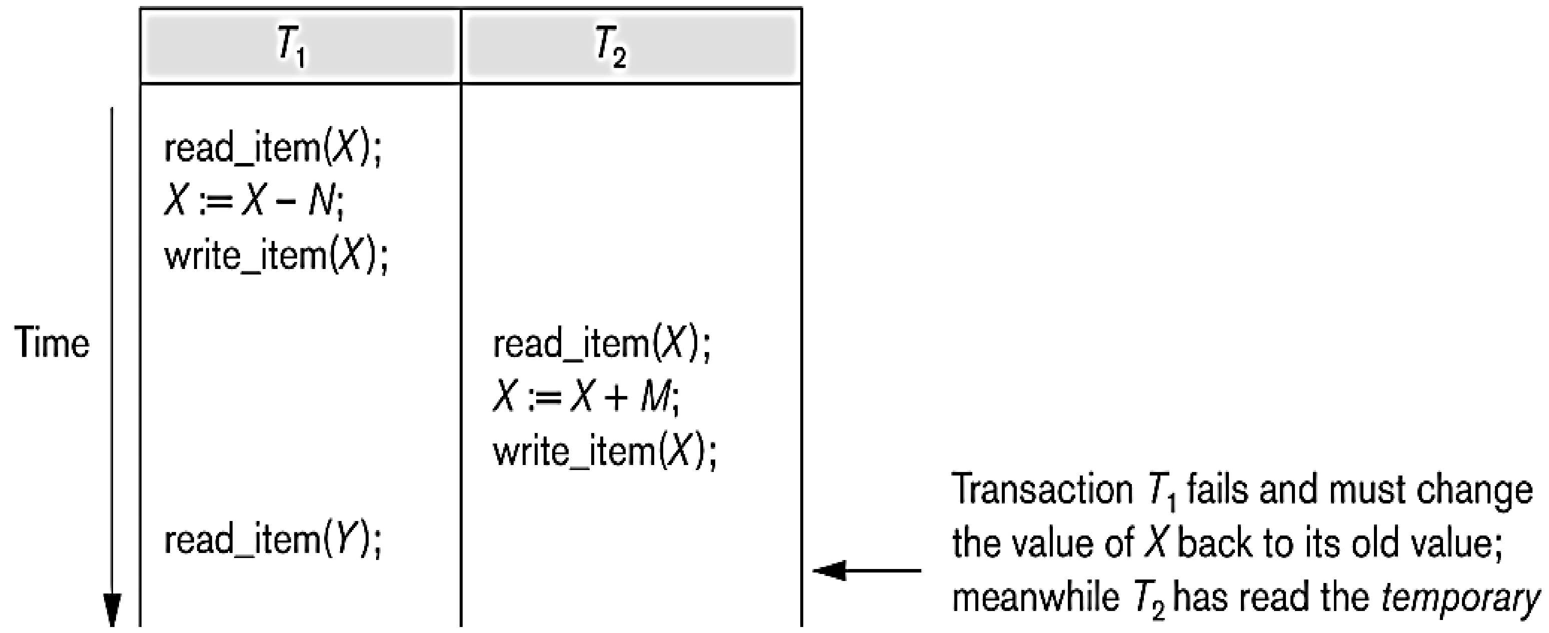
(a)

T_1	T_2
read_item(X);	
$X := X - N$;	
	read_item(X);
	$X := X + M$;
write_item(X);	
read_item(Y);	
	write_item(X);
$Y := Y + N$;	
write_item(Y);	

Item X has an incorrect value because its update by T_1 is *lost* (overwritten).

Concurrent execution is uncontrolled: (b) The temporary update problem.

(b)



T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);	

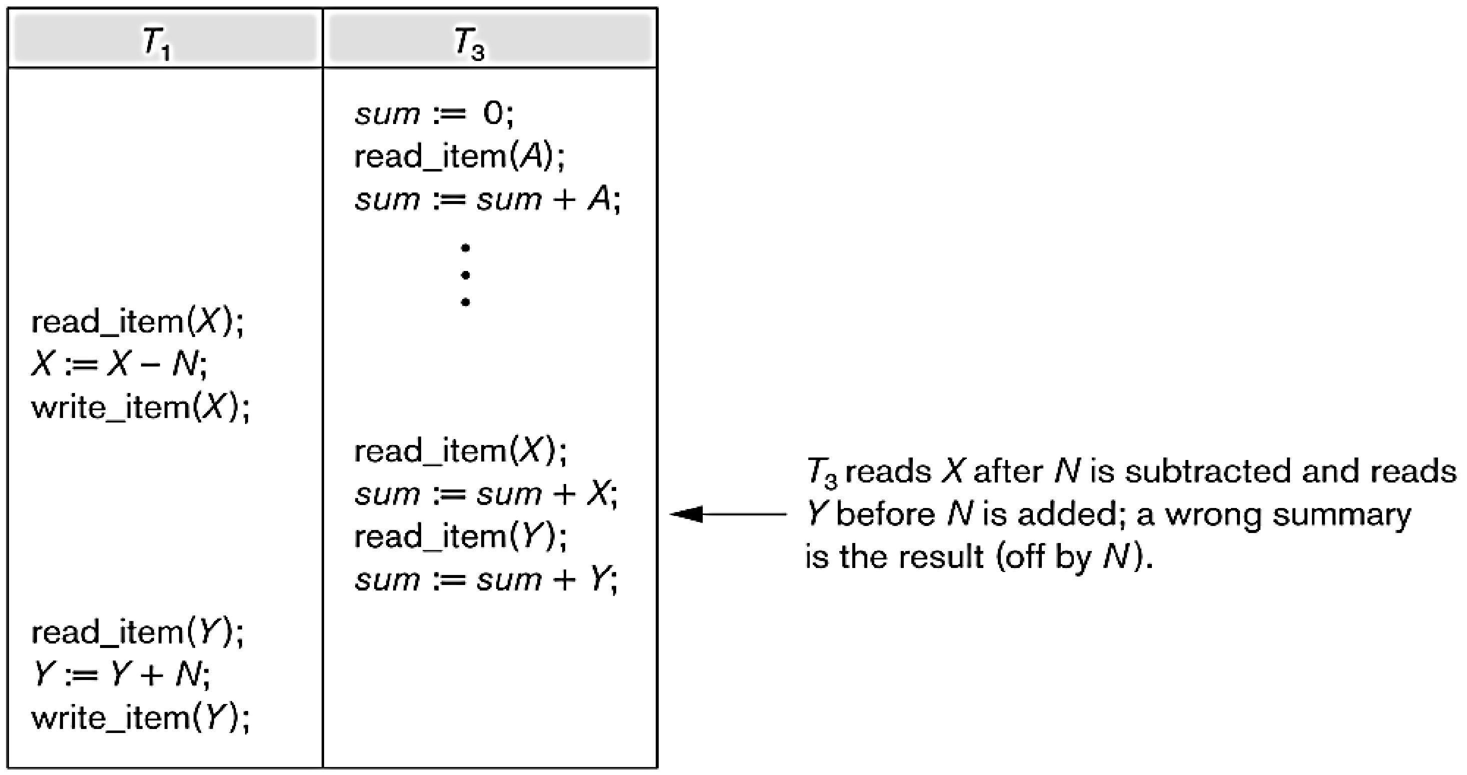
	read_item(X);
	$X := X + M$;
	write_item(X);

read_item(Y);	

Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the *temporary*

Concurrent execution is uncontrolled: (c) The incorrect summary problem.

(c)



T_1	T_3

	sum := 0;
	read_item(*A*);
	sum := *sum* + *A*;
	⋮

read_item(*X*);
X := *X* − *N*;
write_item(*X*);	

	read_item(*X*);
	sum := *sum* + *X*;
	read_item(*Y*);
	sum := *sum* + *Y*;

read_item(*Y*);
Y := *Y* + *N*;
write_item(*Y*);	

T_3 reads *X* after *N* is subtracted and reads *Y* before *N* is added; a wrong summary is the result (off by *N*).

Schedules

- **Schedule** – a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed
  - a schedule for a set of transactions must consist of all instructions of those transactions
  - must preserve the order in which the instructions appear in each individual transaction.
- A transaction that successfully completes its execution will have a commit instructions as the last statement
  - by default transaction assumed to execute commit instruction as its last step
- A transaction that fails to successfully complete its execution will have an abort instruction as the last statement

Scheduling of Transactions

- **Complete Schedule**: A schedule that contains either an abort or a commit for each transaction whose actions are listed in it is called a complete schedule.

- **Serial Schedule**: Each serial schedule consists of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule. If the actions of different transactions are not interleaved, we call that schedule a serial schedule.

Schedule 1

- Let T_1 transfer \$50 from A to B, and T_2 transfer 10% of the balance from A to B.
- A serial schedule in which T_1 is followed by T_2 :

T_1	T_2
read (A)	
$A := A - 50$	
write (A)	
read (B)	
$B := B + 50$	
write (B)	
commit	
	read (A)
	$temp := A * 0.1$
	$A := A - temp$
	write (A)
	read (B)
	$B := B + temp$
	write (B)
	commit

Schedule 2

- A serial schedule where T_2 is followed by T_1

T_1	T_2
	read (A)
	$temp := A * 0.1$
	$A := A - temp$
	write (A)
	read (B)
	$B := B + temp$
	write (B)
	commit
read (A)	
$A := A - 50$	
write (A)	
read (B)	
$B := B + 50$	
write (B)	
commit	

Schedule 3

- Let T_1 and T_2 be the transactions defined previously. The following schedule is not a serial schedule, but it is *equivalent* to Schedule 1.

T_1	T_2
read (A)	
$A := A - 50$	
write (A)	
	read (A)
	$temp := A * 0.1$
	$A := A - temp$
	write (A)
read (B)	
$B := B + 50$	
write (B)	
commit	
	read (B)
	$B := B + temp$
	write (B)
	commit

In Schedules 1, 2 and 3, the sum A + B is preserved.

Schedule 4

- The following concurrent schedule does not preserve the value of (*A* + *B*).

T_1	T_2
read (*A*)	
A := *A* – 50	
	read (*A*)
	temp := *A* * 0.1
	A := *A* - *temp*
	write (*A*)
	read (*B*)
write (*A*)	
read (*B*)	
B := *B* + 50	
write (*B*)	
commit	
	B := *B* + *temp*
	write (*B*)
	commit

QUIZ

The scheme that controls the interaction between executing transactions is called as _____

a) Concurrency control scheme

b) Multiprogramming scheme

c) Serialization scheme

d) Schedule scheme

QUIZ

The execution sequences in concurrency control are termed as ________

a) Serials

b) Schedules

c) Organizations

d) Time tables