TRANSACTION

- Collections of operations that form a single logical unit of work are called **transactions.**
- A transaction is a unit of program execution that accesses and possibly updates various data items.
- The transaction consists of all operations executed between the **begin transaction** and **end transaction.**

OPERATIONS OF TRANSACTION

A transaction is an atomic unit of work that is either completed in its entirety or not done at all. For recovery purposes, the system needs to keep track of when the transaction starts, terminates, and commits or aborts (see below). Hence, the recovery manager keeps track of the following operations:

- **BEGIN_TRANSACTION:** This marks the beginning of transaction execution.
- **READ or WRITE:** These specify read or write operations on the database items that are executed as part of a transaction.
- **COMMIT_TRANSACTION:** This signals a successful end of the transaction so that any changes (updates) executed by the transaction can be safely committed to the database and will not be undone.
- **ROLLBACK (or ABORT):** This signals that the transaction has ended unsuccessfully, so that any changes or effects that the transaction may have applied to the database must be undone.
- **END_TRANSACTION:** This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution. However, at this point it may be necessary to check whether the changes introduced by the transaction can be permanently applied to the database (committed) or whether the transaction has to be aborted because it violates serializability (see Document 23) or for some other reason.

TRANSACTION STATES

A transaction must be in one of the following,

- **Active,** the initial state; the transaction stays in this state while it is executing.
- **Partially committed,** after the final statement has been executed.
- **Failed,** after the discovery that normal execution can no longer proceed.
- **Aborted,** after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- **Committed,** after successful completion.

STATE DIAGRAM OF TRANSACTION

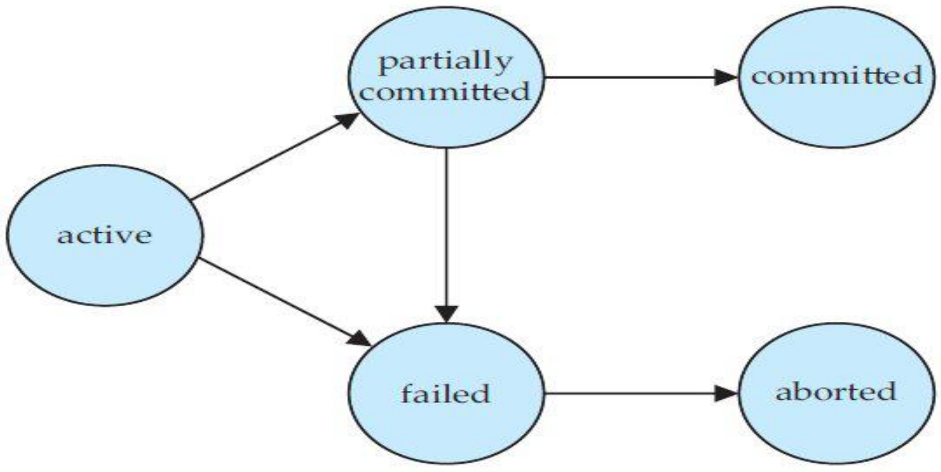
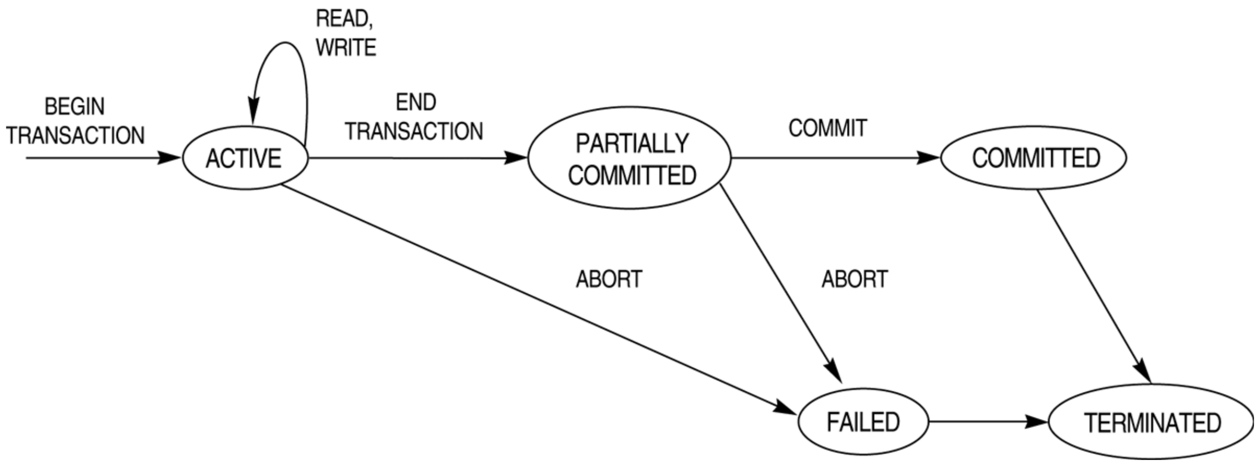


Figure 14.1 State diagram of a transaction.



DESIRABLE PROPERTIES OF TRANSACTION

A transaction is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data the database system must ensure:

- **Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency:** Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
- **Durability.** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

These properties are often called the **ACID properties**; the acronym is derived from the first letter of each of the four properties.

NEED FOR CONCURRENCY CONTROL

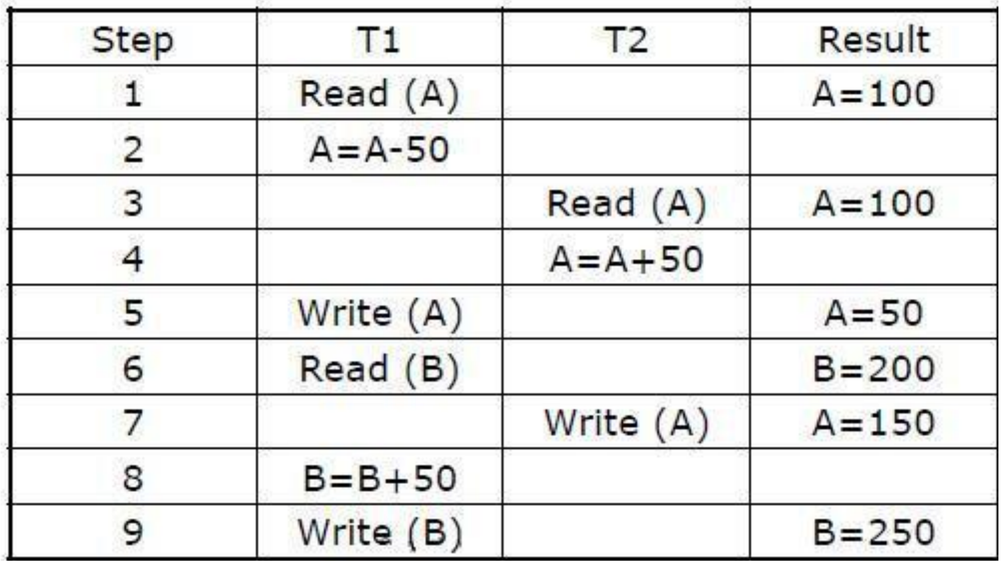
- The concurrent execution of transactions may lead to problems such as an inconsistent database.
- Concurrency Control techniques are used to ensure that multiple transactions submitted by various users do not interfere with one another in a way that produces incorrect results.
- The effect on a database of any number of transactions executing in parallel must be the same as if they were executed one after another.

The types of problems we may encounter with the two transactions if they run concurrently.

- The Lost Update Problem
- The Temporary Update (or Dirty Read) Problem
- The Incorrect Summary Problem

THE LOST UPDATE PROBLEM

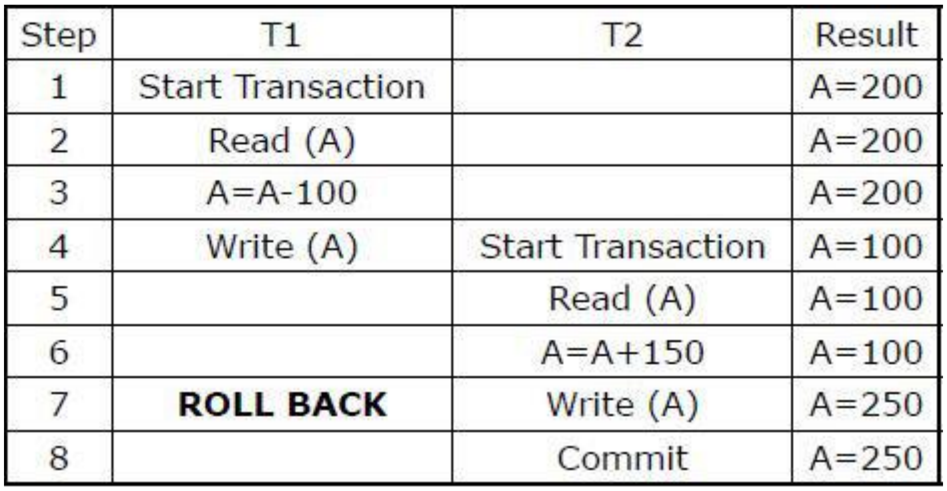
This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.



Step	T1	T2	Result
1	Read (A)		A=100
2	A=A-50		
3		Read (A)	A=100
4		A=A+50	
5	Write (A)		A=50
6	Read (B)		B=200
7		Write (A)	A=150
8	B=B+50		
9	Write (B)		B=250

THE TEMPORARY UPDATE (OR DIRTY READ) PROBLEM

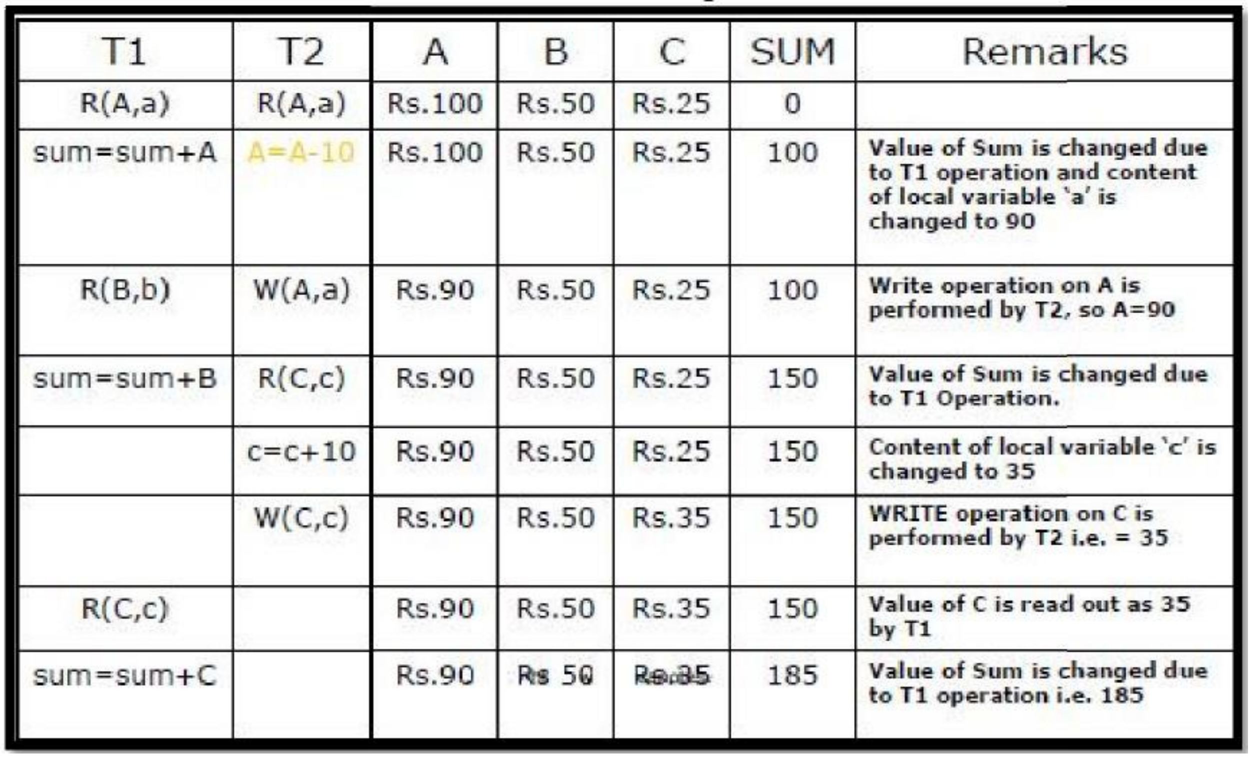
This problem occurs when one transaction updates a database item and then the transaction fails for some reason .The updated item is accessed by another transaction before it is changed back to its original value.



Step	T1	T2	Result
1	Start Transaction		A=200
2	Read (A)		A=200
3	A=A-100		A=200
4	Write (A)	Start Transaction	A=100
5		Read (A)	A=100
6		A=A+150	A=100
7	**ROLL BACK**	Write (A)	A=250
8		Commit	A=250

THE INCORRECT SUMMARY PROBLEM

This problem occurs when a transaction reads several values from a database while a second transaction updates some of them.



T1	T2	A	B	C	SUM	Remarks
R(A,a)	R(A,a)	Rs.100	Rs.50	Rs.25	0	
sum=sum+A	A=A-10	Rs.100	Rs.50	Rs.25	100	**Value of Sum is changed due to T1 operation and content of local variable 'a' is changed to 90**
R(B,b)	W(A,a)	Rs.90	Rs.50	Rs.25	100	**Write operation on A is performed by T2, so A=90**
sum=sum+B	R(C,c)	Rs.90	Rs.50	Rs.25	150	**Value of Sum is changed due to T1 Operation.**
	c=c+10	Rs.90	Rs.50	Rs.25	150	**Content of local variable 'c' is changed to 35**
	W(C,c)	Rs.90	Rs.50	Rs.35	150	**WRITE operation on C is performed by T2 i.e. = 35**
R(C,c)		Rs.90	Rs.50	Rs.35	150	**Value of C is read out as 35 by T1**
sum=sum+C		Rs.90	Rs 50	Rs.35	185	**Value of Sum is changed due to T1 operation i.e. 185**