SALALE UNIVERSITY

COMPUTER SCIENCE PROGRAM

ADVANCED DATABASE SYSTEMS

CHAPTER ONE

Transaction Processing Concepts

Chapter outline

- Introduction
- Transaction and System Concepts
- Properties of Transaction
- Schedules and Recoverability
- Serializability of Schedules
- Transaction Definition in SQL

INTRODUCTION

- A transaction is a unit of program execution that accesses and possibly updates various data items.
- Transaction processing systems are systems with large databases and hundreds of concurrent users executing database transactions.
- Examples of such systems include
  - ✓ airline reservations
  - ✓ banking,
  - ✓ credit card processing
  - ✓ online retail purchasing
  - ✓ stock markets
  - ✓ supermarket checkouts, and
  - ✓ many other applications.

CONT…

- To ensure integrity of the data, we require that the database system maintain the following properties of the transactions: These properties are often called the **ACID properties**
- **Atomicity**. Either all operations of the transaction are reflected properly in the database, or none are.
- **Consistency**. Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database.
- **Isolation**. Even though multiple transactions may execute concurrently, the system guarantees that, each transaction is unaware of other transactions executing concurrently in the system.
- **Durability**. After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

CONT…

- The database permanently resides on disk, but that some portion of it is temporarily residing in main memory. **Transactions access data using two operations:**
- **read(*X*)**, which transfers the data item *X* from the database to a local buffer belonging to the transaction that executed the read operation.
- **write(*X*)**, which transfers the data item *X* from the the local buffer of the transaction that executed the write back to the database.
- Let *Ti* be a transaction that transfers $50 from account *A* to account *B*. This transaction
- can be defined as
- *Ti*:
- read(*A*);
- $A := A - 50$;
- write(*A*);
- read(*B*);
- $B := B + 50$;
- write(*B*).

CONT…

- The transaction must either fully happen, or not happens at all. It must not complete partially. It is handled by a component called the **transaction-management component**.
- Ensuring durability is the responsibility of a component of the database system called the **recovery-management component**.
- Ensuring the isolation property is the responsibility of a component of the database system called the **concurrency-control component**.

TRANSACTION STATES

- All transactions complete successfully in the absence of failures.
- When a transaction does not always complete its execution successfully it is termed as **aborted**.
- To maintain the atomicity once the changes caused by an aborted transaction have been undone, then the transaction has been **rolled back**.
- A transaction that completes its execution successfully is said to be **committed**.

CONT…

- **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
- The transaction is in **committed** state only if it has entered the committed state. Similarly, the transaction has been **aborted** only if it has entered the aborted state.
- A transaction is said to have **terminated** if has either committed or aborted.

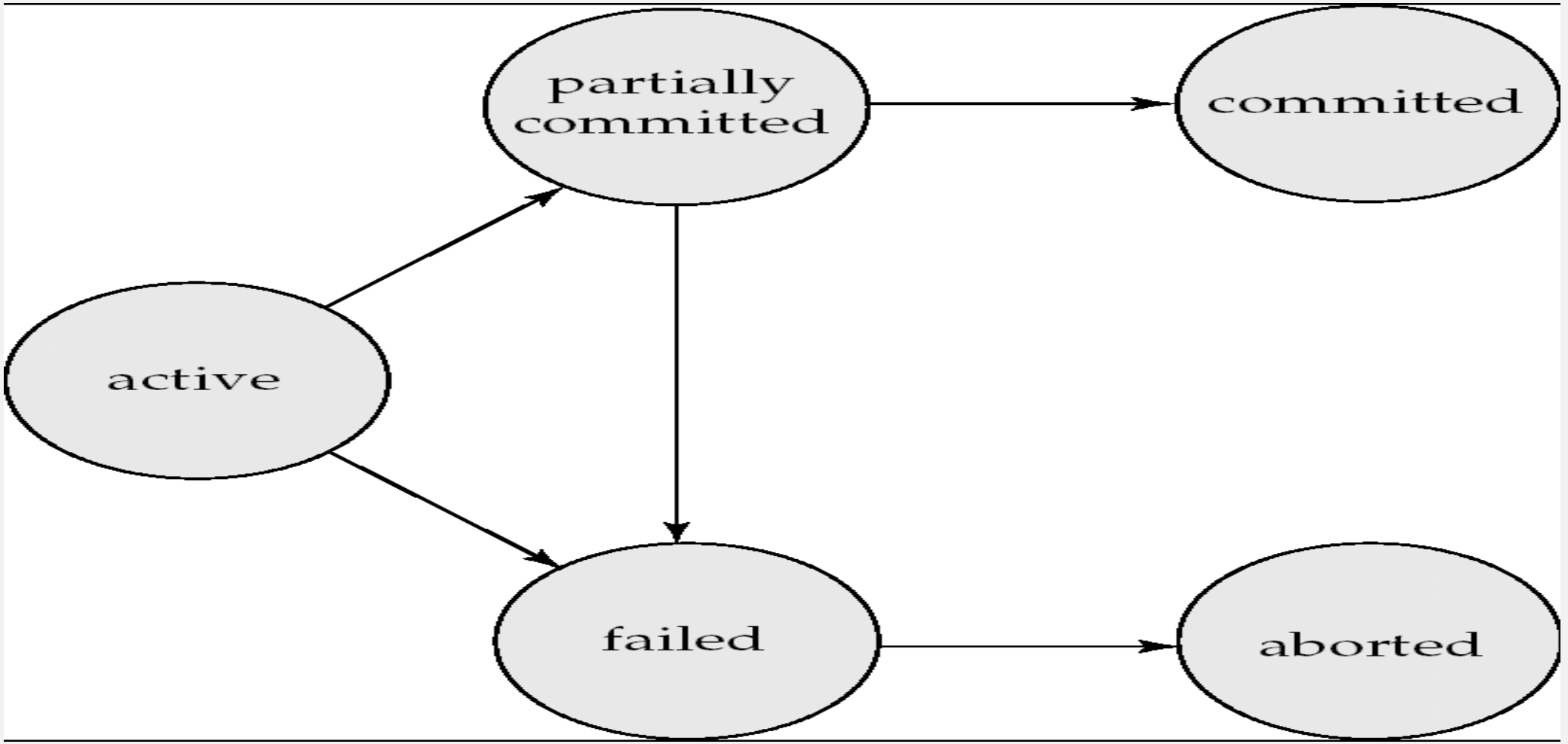
CONT…

- A transaction enters the failed state after the system determines that the transaction can no longer proceed with its normal execution (for example, because of hardware or logical errors). Such a transaction must be rolled back. Then, it enters the aborted state.
- At this point, the system has two options:

• It can **restart** the transaction, but only if the transaction was aborted as a result of some hardware or software error that was not created through the internal logic of the transaction.

• It can **kill** the transaction. It usually does so because of some internal logical error that can be corrected only by rewriting the application program, or because the input was bad, or because the desired data were not found in the database.

CONT…



IMPLEMENTATION OF ATOMICITY AND DURABILITY

- The **recovery-management** component of a database system implements the support for atomicity and durability.
  - E.g. the ***shadow-database*** scheme
- In the shadow-copy scheme, a transaction that wants to update the database first creates a complete copy of the database. All updates are done on the new database copy, leaving the original copy, the **shadow copy**, untouched.
- The shadow-database scheme:
  - Assumes that only one transaction is active at a time.
  - Assumes disks do not fail
  - Useful for text editors
  - extremely inefficient for large databases
  - Does not handle concurrent transactions

CONCURRENT EXECUTION

- **Transaction-processing systems** usually allow multiple transactions to run concurrently.
- Allowing multiple transactions to update data concurrently **causes several complications with consistency of the data**.
- We can **ensure consistency** by insisting the transactions to run **serially**—that is, one at a time, each starting only after the previous one has completed.
- However, there are two good reasons for allowing concurrency:
- **Improved throughput and resource utilization**.
  - **Throughput** of the system - the number of transactions executed in a given amount of time. Correspondingly, the processor and disk **utilization** also increase
- **Reduced waiting time**. There may be a mix of transactions running on a system, some short and some long. If transactions run serially, a short transaction may have to wait for a preceding long transaction to complete, which can lead to unpredictable delays in running a transaction.

SCHEDULES AND RECOVERABILITY

Transaction schedule or history:

- When transactions are executing concurrently in an interleaved fashion, the order of execution of operations from the various transactions forms what is known as a transaction schedule (or history).

A **schedule(or history) S of n transactions T1, T2, …, Tn:**

- It is an ordering of the operations of the transactions subject to the constraint that, for each transaction Ti that participates in S, the operations of Ti in S must appear in the same order in which they occur in Ti.
- Note, however, that operations from other transactions Tj can be interleaved with the operations of Ti in S.

CONT…

Schedules classified on recoverability:

Recoverable schedule:

- One where no transaction needs to be rolled back.
- A schedule S is recoverable if no transaction T in S commits until all transactions T’ that have written an item that T reads have committed.

Cascade less schedule:

- One where every transaction reads only the items that are written by committed transactions.

Schedules requiring cascaded rollback:

- A schedule in which uncommitted transactions that read an item from a failed transaction must be rolled back.

Strict Schedules:

- A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.

Serializability of Schedules:

Serial schedule:

- A schedule S is serial if, for every transaction T participating in the schedule, all the operations of T are executed consecutively in the schedule.
- Otherwise, the schedule is called non serial schedule.

SERIALIZABLE SCHEDULE:

- A schedule S is serializable if it is equivalent to some serial schedule of the same n transactions.

Result equivalent:

- Two schedules are called result equivalent if they produce the same final state of the database.

Conflict equivalent:

- Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

Conflict serializable:

- A schedule S is said to be conflict serializable if it is conflict equivalent to some serial schedule S’.

Being serializable is not the same as being serial

- Being serializable implies that the schedule is a correct schedule.
- It will leave the database in a consistent state.
- The interleaving is appropriate and will result in a state as if the transactions were serially executed, yet will achieve efficiency due to concurrent execution

CONT…

Serializability is hard to check.

- Interleaving of operations occurs in an operating system through some scheduler
- Difficult to determine beforehand how the operations in a schedule will be interleaved

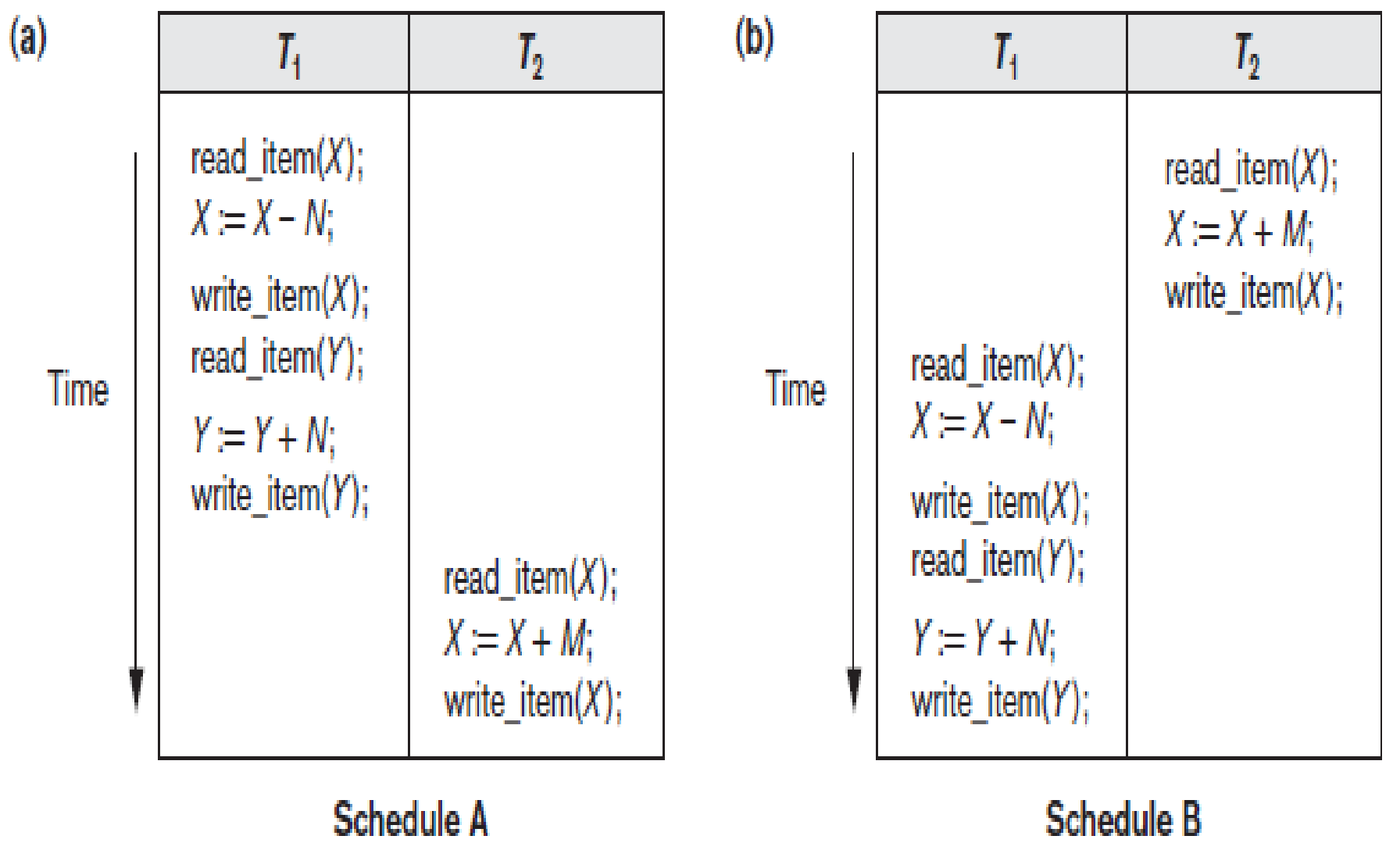
Examples of serial and nonserial schedules involving transactions *T1 and T2.*

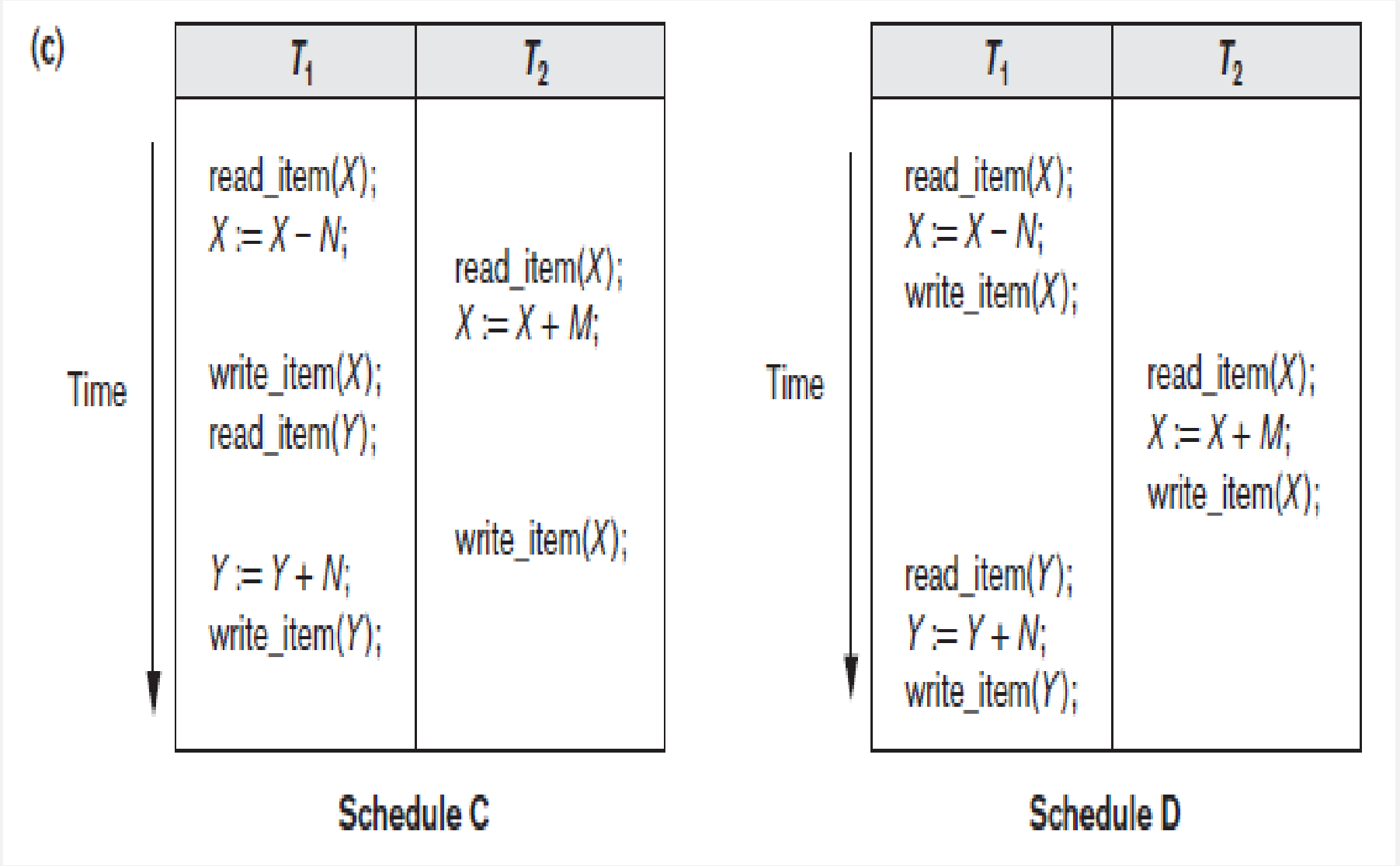
*(a)*Serial schedule A: *T1 followed by T2.*

(b) Serial schedule B: T2 followed by T1.

(c) Two nonserial schedules C and D with interleaving of operations.

EXAMPLE


(a)
T_1
T_2
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
read_item(X);
X := X + M;
write_item(X);
Time
Schedule A
(b)
T_1
T_2
read_item(X);
X := X + M;
write_item(X);
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
Time
Schedule B


(c)
Time
T_1
T_2
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
read_item(X);
X := X + M;
write_item(X);
Schedule C
Time
T_1
T_2
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
read_item(X);
X := X + M;
write_item(X);
Schedule D

CONT…

- Consider the schedules in Figure above , and assume that the initial values of database items are X = 90 and Y = 95 and that N = 3 and M = 2.
- After executing transactions T1 and T2, we would expect the database values to be X= 89 and Y = 98, according to the meaning of the transactions.

Executing either of the serial schedules A or B gives the correct results.

Now consider the nonserial schedules C and D. Schedule C gives the results X = 92 and Y = 98, in which the X value is erroneous, whereas schedule D gives the correct results.

TRANSACTION DEFINITION IN SQL

- A data-manipulation language must include a construct for specifying the set of actions that constitute a transaction.
- The SQL standard specifies that a transaction begins implicitly.
- Transactions are ended by one of these SQL statements:
  - **Commit work** commits the current transaction and begins a new one.
  - **Rollback work** causes the current transaction to abort.
- *The keyword **work** is optional in both the statements. If a program terminates without either of these commands, the updates are either committed or rolled back*

THANK YOU!