INTRODUCTION TO TRANSACTION PROCESSING

CHAPTER 9

Introduction

- One criterion for classifying a database system is according to the number of users who can use the system concurrently-that is, at the same time.
- A DBMS is single-user if at most one user at a time can use the system, and it is multiuser if many users can use the system-and hence access the database-concurrently.
- Single-user DBMSs are mostly restricted to personal computer systems; most other DBMSs are multiuser.

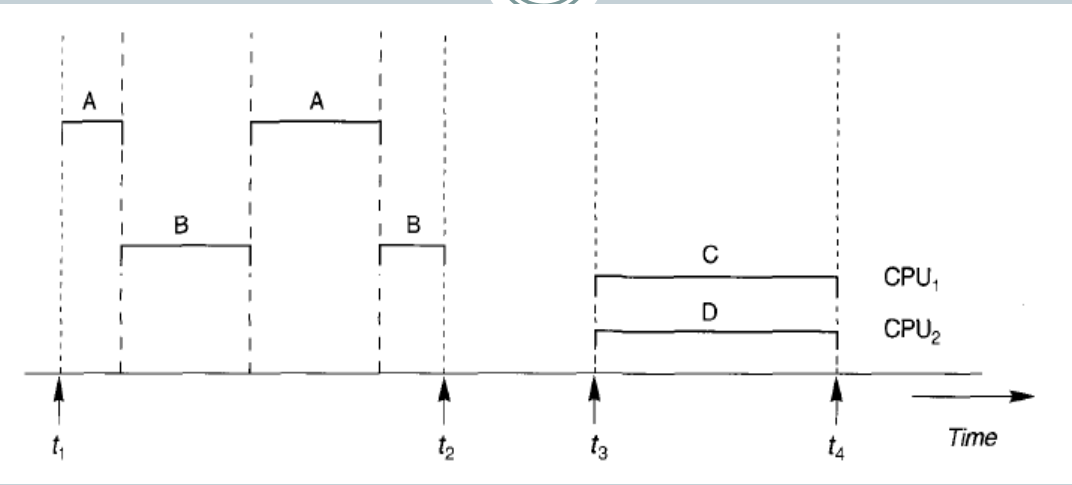
Introduction

- Multiple users can access databases-and use computer systems-simultaneously because of the concept of multiprogramming, which allows the computer to execute multiple programs-or processes-at the same time.
- If only a single central processing unit (CPU) exists, it can actually execute at most one process at a time.

Introduction

- However, multiprogramming operating systems execute some commands from one process, then suspend that process and execute some commands from the next process, and so on.
- A process is resumed at the point where it was suspended whenever it gets its turn to use the CPU again.
- Hence, concurrent execution of processes is actually interleaved.

Introduction



- Figure , shows two processes A and B executing concurrently in an interleaved fashion.
- Interleaving keeps the CPU busy when a process requires an input or output (r/o) operation, such as reading a block from disk.

Introduction

- The CPU is switched to execute another process rather than remaining idle during r/o time.
- Interleaving also prevents a long process from delaying other processes.
- If the computer system has multiple hardware processors (CPUs), parallel processing of multiple processes is possible, as illustrated by processes C and D in Figure.

Transactions

- 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.
- The database operations that form a transaction can either be embedded within an application program or they can be specified interactively via a high-level query language such as SQL.

- One way of specifying the transaction boundaries is by specifying explicit **begin transaction** and **end transaction** statements in an application program; in this case, all database access operations between the two are considered as forming one transaction.

Read and Write Operations

- The basic database access operations that a transaction can include are as follows:
- **read_item(X):** Reads a database item named X into a program variable.
- **write_item(X):** Writes the value of program variable X into the database item namedX.

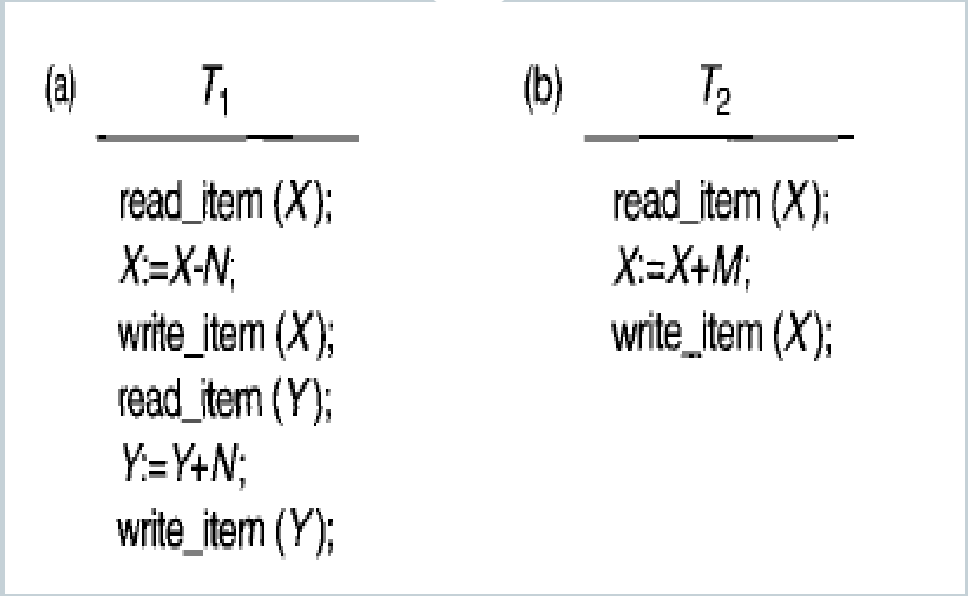
DBMS Buffers

- The DBMS will generally maintain a number of buffers in main memory that hold database disk blocks containing the database items being processed.
- When these buffers are all occupied, and additional database blocks must be copied into memory, some buffer replacement policy is used to choose which of the current buffers is to be replaced.
- If the chosen buffer has been modified, it must be written back to disk before it is reused.

Why Concurrency Control Is Needed

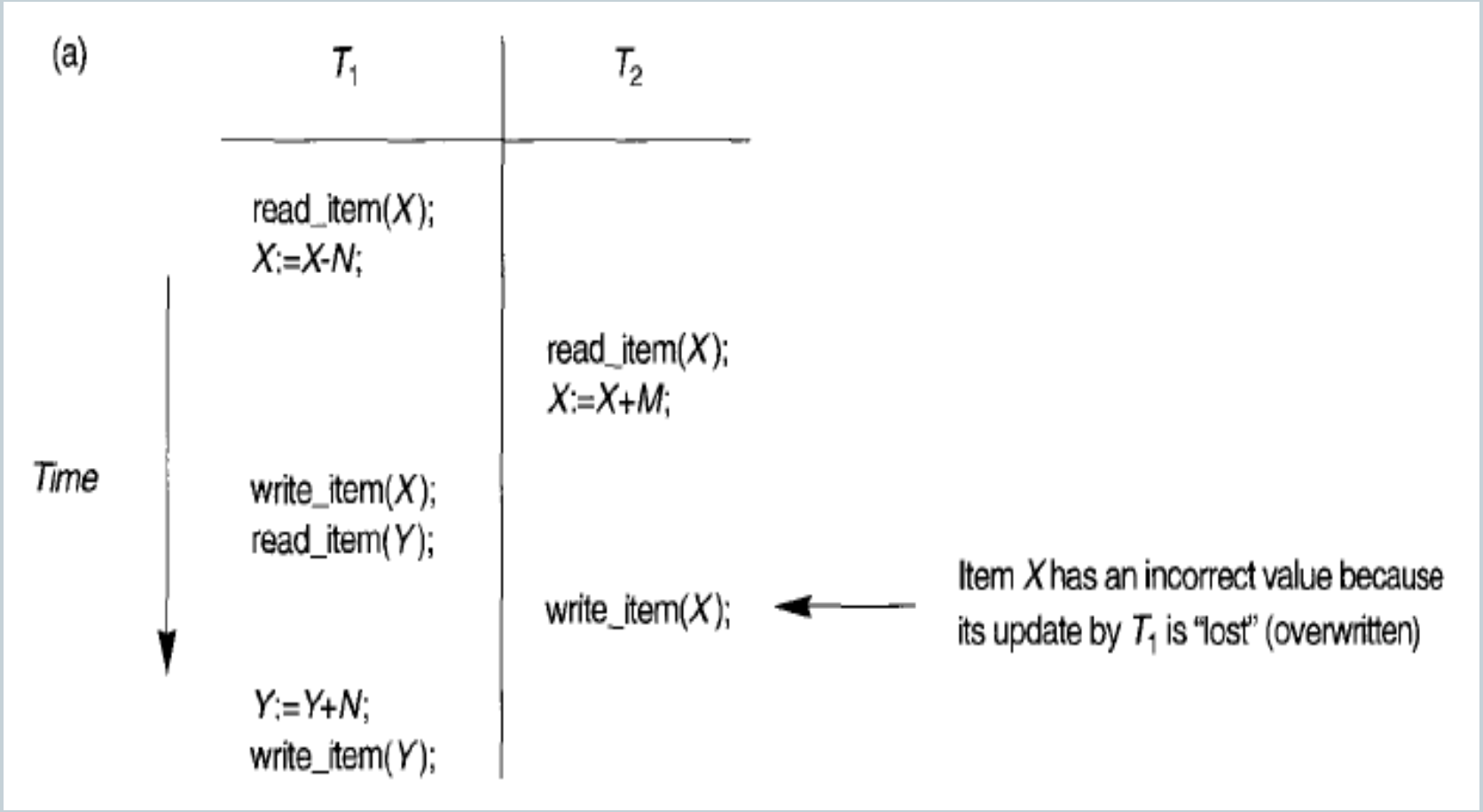
- Several problems can occur when concurrent transactions execute in an uncontrolled manner.
- Figure 2(a) shows a transaction T1 that *transfers* N reservations from one flight whose number of reserved seats is stored in the database item named X to another flight whose number of reserved seats is stored in the database item named Y.
- Figure 2(b) shows a simpler transaction T2 that just *reserves* M seats on the first flight (X) referenced in transaction T1.

Why Concurrency Control Is Needed



- **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 items incorrect.

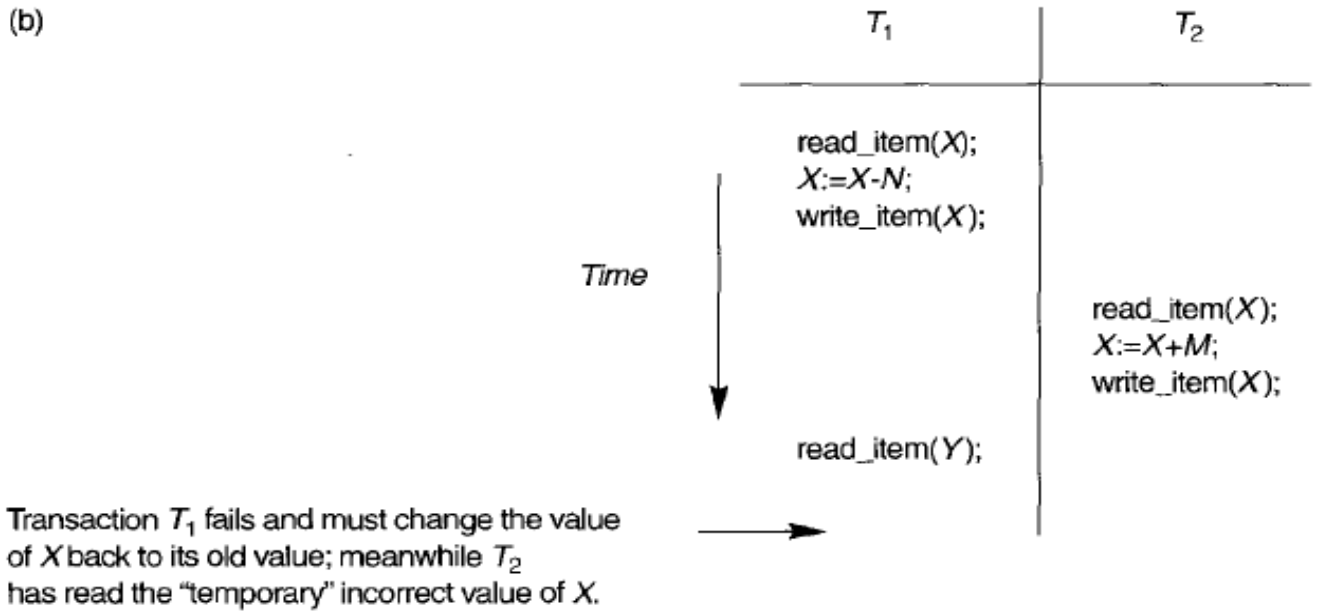
Why Concurrency Control Is Needed



Why Concurrency Control Is Needed

- **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.

Why Concurrency Control Is Needed



The value of item X that is read by T_2 is called ***dirty data***, because it has been created by a transaction that has not completed and committed yet; hence, this problem is also known as the ***dirty read problem.***

Why Concurrency Control Is Needed

- **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.

Why Concurrency Control Is Needed

- **For example:** Suppose that a transaction T3 is calculating the total number of reservations on all the flights; meanwhile, transaction T1 is executing.
- If the interleaving of operations occurs, the result of *T3* will be off by an amount N because T3 reads the value of X *after* N seats have been subtracted from it but reads the value of *Y before* those N seats have been added to it.

Why Concurrency Control Is Needed

- Another problem that may occur is called unrepeatable read, where a transaction T reads an item twice and the item is changed by another transaction T' between the two reads.
- Hence, T receives *different values* for its two reads of the same item.

Why Recovery Is Needed

- Whenever a transaction is submitted to a DBMS for execution, the system is responsible for making sure that either
- all the operations in the transaction are completed successfully and their effect is recorded permanently in the database, or
- the transaction has no effect whatsoever on the database or on any other transactions.
- The DBMS must not permit some operations of a transaction *T* to be applied to the database while other operations of T are not.

Types of Failures

- ***A computer failure (system crash):***
  - A hardware, software, or network error occurs in the computer system during transaction execution.
  - Hardware crashes are usually media failures-for example, main memory failure.
- ***A transaction or system error:***
  - Some operation in the transaction may cause it to fail, such as integer overflow or division by zero.
  - Transaction failure may also occur because of erroneous parameter values or because of a logical programming error.
  - In addition, the user may interrupt the transaction during its execution

Types of Failures

- ***Local errors or exception conditions detected by the transaction:***
  - During transaction execution, certain conditions may occur that necessitate cancellation of the transaction.
  - For example, data for the transaction may not be found. Notice that an exception condition," such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be canceled.
  - This exception should be programmed in the transaction itself, and hence would not be considered a failure.

Types of Failures

- ***Concurrency control enforcement:***
  - The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serializability or because several transactions are in a state of deadlock.
- ***Disk failure:***
  - Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.
- ***Physical problems and catastrophes:***
  - This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

TRANSACTION AND SYSTEM CONCEPTS

- 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.

Transaction Operations

- BEGIN_TRANSACTION: This marks the beginning of transaction execution.
- READ DR WRITE: These specify read or write operations on the database items that are executed as part of a transaction.
- END_TRANSACTION: This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution.

Transaction Operations

- 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*.

Transaction States

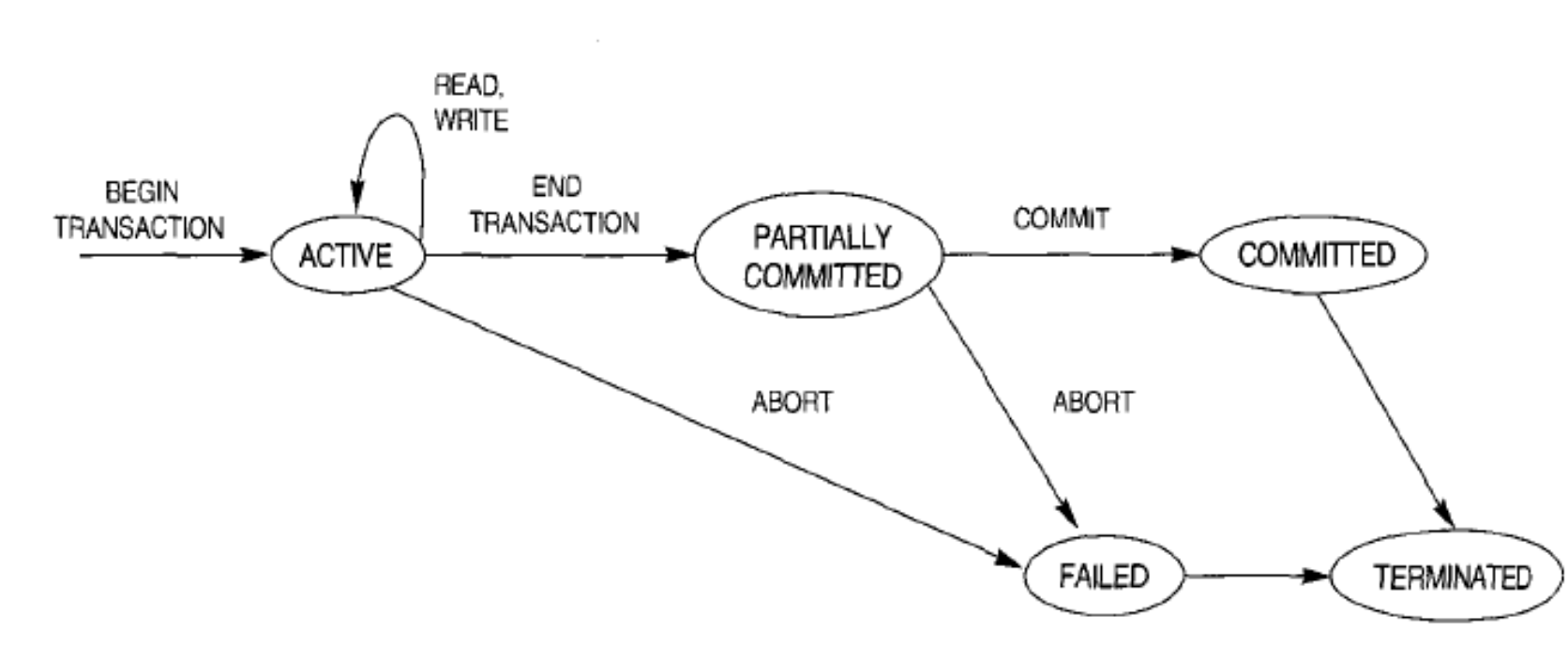


Figure shows a state transition diagram that describes how a transaction moves through its execution states.

Transaction States

- A transaction goes into an active state immediately after it starts execution, where it can issue READ and WRITE operations.
- When the transaction ends, it moves to the partially committed state.
  At this point, some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the transaction permanently.
- Once this check is successful, the transaction is said to have reached its commit point and enters the committed state.

Transaction States

- Once a transaction is committed, it has concluded its execution successfully and all its changes must be recorded permanently in the database.
- However, a transaction can go to the failed state if one of the checks fails or if the transaction is aborted during its active state.
- The transaction may then have to be rolled back to undo the effect of its WRITE operations on the database.
- The terminated state corresponds to the transaction leaving the system.

The System Log

- To be able to recover from failures that affect transactions, the system maintains a log to keep track of all transaction operations that affect the values of database items.
- This information may be needed to permit recovery from failures.

Commit Point of a Transaction

- A transaction T reaches its **commit point** when all its operations that access the database have been executed successfully *and* the effect of all the transaction operations on the database have been recorded in the log.
- Beyond the commit point, the transaction is said to be **committed,** and its effect is assumed to be *permanently recorded* in the database. The transaction then writes a commit record [commit,T] into the log.

DESIRABLE PROPERTIES OF TRANSACTIONS

- ***Atomicity:*** A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- ***Consistency preservation:*** A transaction is consistency preserving if its complete execution takes the database from one consistent state to another.

DESIRABLE PROPERTIES OF TRANSACTIONS

- ***Isolation:*** A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.
- ***Durability or permanency:*** The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.