COS221 - L30 - Transaction processing 1

Linda Marshall

18 May 2023

Transaction processing systems

- A **transaction** describes a unit of work (database processing). It comprises of a sequence of database actions (operations).
- **Transaction processing systems** have large databases and have multiple concurrent users executing transactions. These systems require high response for large volumes of users.
- Transactions need to execute correctly and therefore keep the database in a valid state.

Introduction to Transaction Processing

- **Single-user** database systems are typically restricted to one computer and do not have multiple users accessing the database. It therefore does not suffer from concurrent access to data and what happens after transaction failure when multiple users have either read from or written to the database.
- In a **multiuser system**, transactions are processed concurrently. Concurrently processed transactions are either executed in an *interleaved* fashion or in *parallel*.

Introduction to Transaction Processing

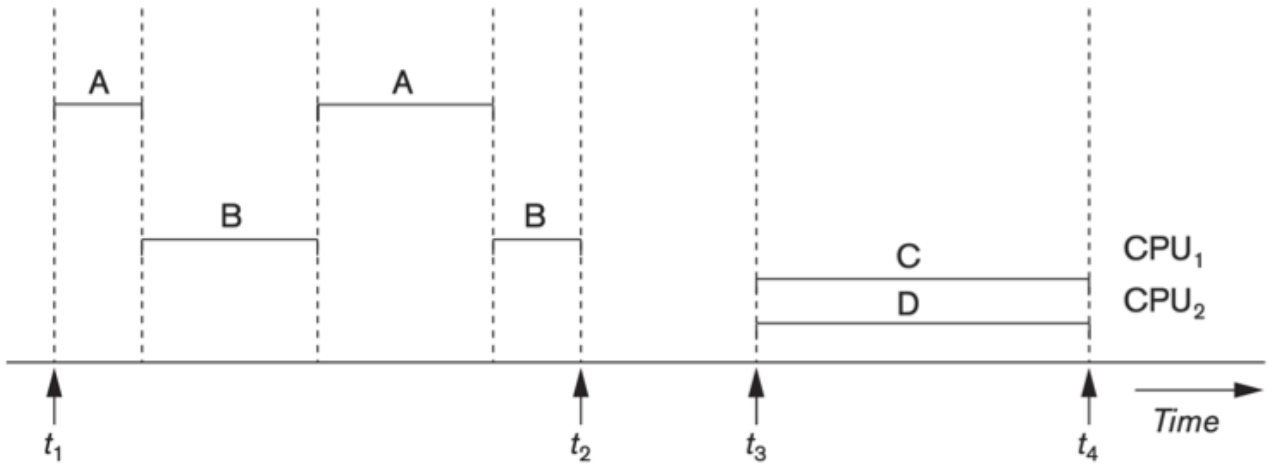


Figure 20.1 Interleaved processing versus parallel processing of concurrent transactions.

Introduction to Transaction Processing

- A transaction is bounded and contains one or more database access operations. A database program may comprise of one or more transactions. Transactions with operations that only read values from the database are referred as *read-only transactions*. All other transactions are referred to as *read-write transactions*.
- To simplify the model for transaction processing, reference will be made to a *data item*. Data items can be of varying granularities and are uniquely identifiable. In this simplified model, a transaction has two database operations:
  - `read_item(X)` - reads database item `X` into program variable `X`
  - `write_item (X)` - writes program variable `X` into database item `X`

Introduction to Transaction Processing

`read_item(X)` includes the following process:

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer). The size of the buffer is the same as the disk block size.
3. Copy item X from the buffer to the program variable named X.

Introduction to Transaction Processing

`write_item(X)` requires the following steps to be executed:

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
3. Copy item X from the program variable named X into its correct location in the buffer.
4. Store the updated disk block from the buffer back to disk (either immediately or at some later point in time).

Introduction to Transaction Processing

The database buffers are managed by the database cache. Usually a disk block is loaded into memory. When the data is written out to disk is dependent on the buffer replacement policy. LRU (least recently used) is an example of such a policy.

Introduction to Transaction Processing

Why is it necessary to have concurrency control (and recovery)?

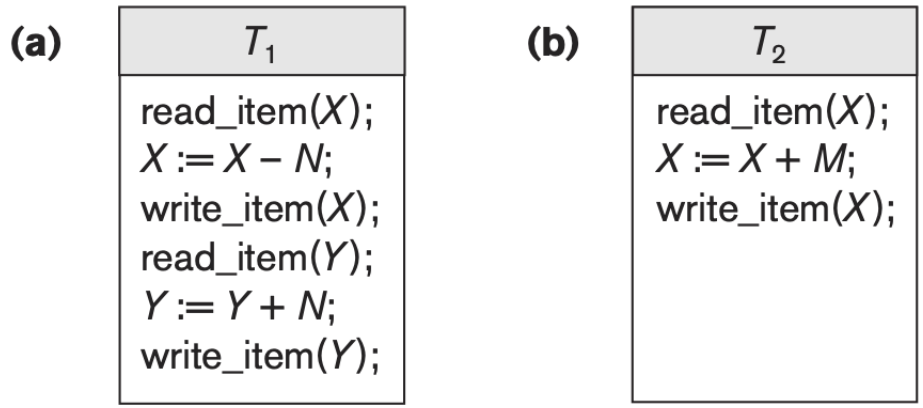


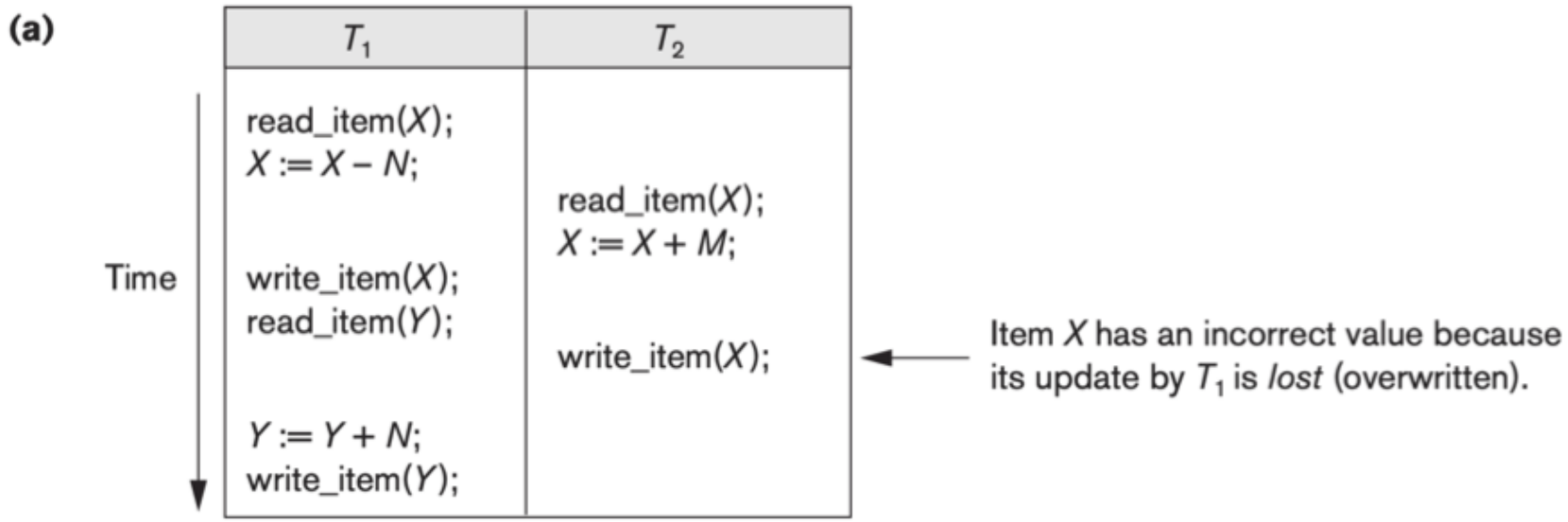
Figure 20.2
Two sample transactions.
(a) Transaction T_1.
(b) Transaction T_2.

Need for concurrency control and recovery

- Lost Update Problem
- Temporary Update Problem
- The Incorrect Summary Problem
- The Unrepeatable Read Problem

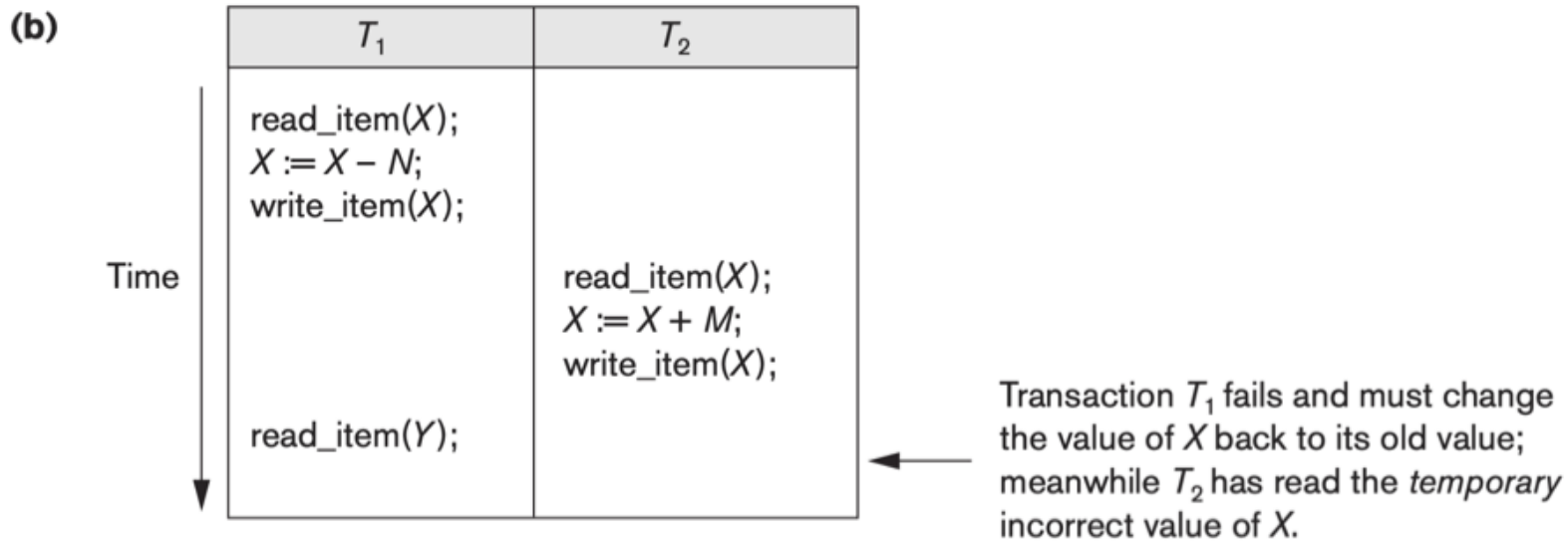
Need for concurrency control and recovery

Lost Update Problem (write-write conflict problem) - Occurs when two transactions update the same data item, but both read the same original value before update



Need for concurrency control and recovery

The Temporary Update (or Dirty Read) Problem - This occurs when one transaction T_1 updates a database item X, which is accessed (read) by another transaction T_2; then T_1 fails for some reason; X was (read) by T_2 before its value is changed back (rolled back or UNDONE) after T_1 fails



Need for concurrency control and recovery

The Incorrect Summary Problem - One transaction is calculating an aggregate summary function on a number of records (for example, sum (total) of all bank account balances) while other transactions are updating some of these records (for example, transferring a large amount between two accounts; the aggregate function may read some values before they are updated and others after they are updated.

(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).

Need for concurrency control and recovery

The Unrepeatable Read Problem - A transaction T_1 may read an item (say, available seats on a flight); later, T_1 may read the same item again and get a different value because another transaction T_2 has updated the item (reserved seats on the flight) between the two reads by T_1

Need for concurrency control and recovery

Why is recovery needed?

- A transaction that has been completed and all its changes have been permanently written to the database is said to be **committed**.
- A transaction that has not been committed after it has executed in its entirety is **aborted**.

Need for concurrency control and recovery

Transaction failure may be as a results of:

1. **A computer failure (system crash)**
   A hardware or software error occurs during transaction execution. If the hardware crashes, the contents of the computer's internal main memory may be lost.
2. **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.

Need for concurrency control and recovery

Transaction failure may be as a results of: (cont.)

3. **Local errors or exception conditions detected by the transaction**
   Certain conditions necessitate cancellation of the transaction. For example, data for the transaction may not be found. A condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be cancelled - a programmed abort causes the transaction to fail.
4. **Concurrency control enforcement**
   The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serialisability or because several transactions are in a state of deadlock (see Chapter 22).

Need for concurrency control and recovery

Transaction failure may be as a results of: (cont.)

5. **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 kind of failure and item 6 are more severe than items 1 through 4.
6. **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.