Transaction Management

SCS2209- Database II

Ms. Hiruni Kegalle

Transactions

- Transactions provide a mechanism to describe ***logical units*** of database processing.
- A transaction is an executing program that includes some database operations: any number of retrieval operations and any number of update operations.
- The operations in the transactions will together form an atomic unit of work against the database.
- At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema.

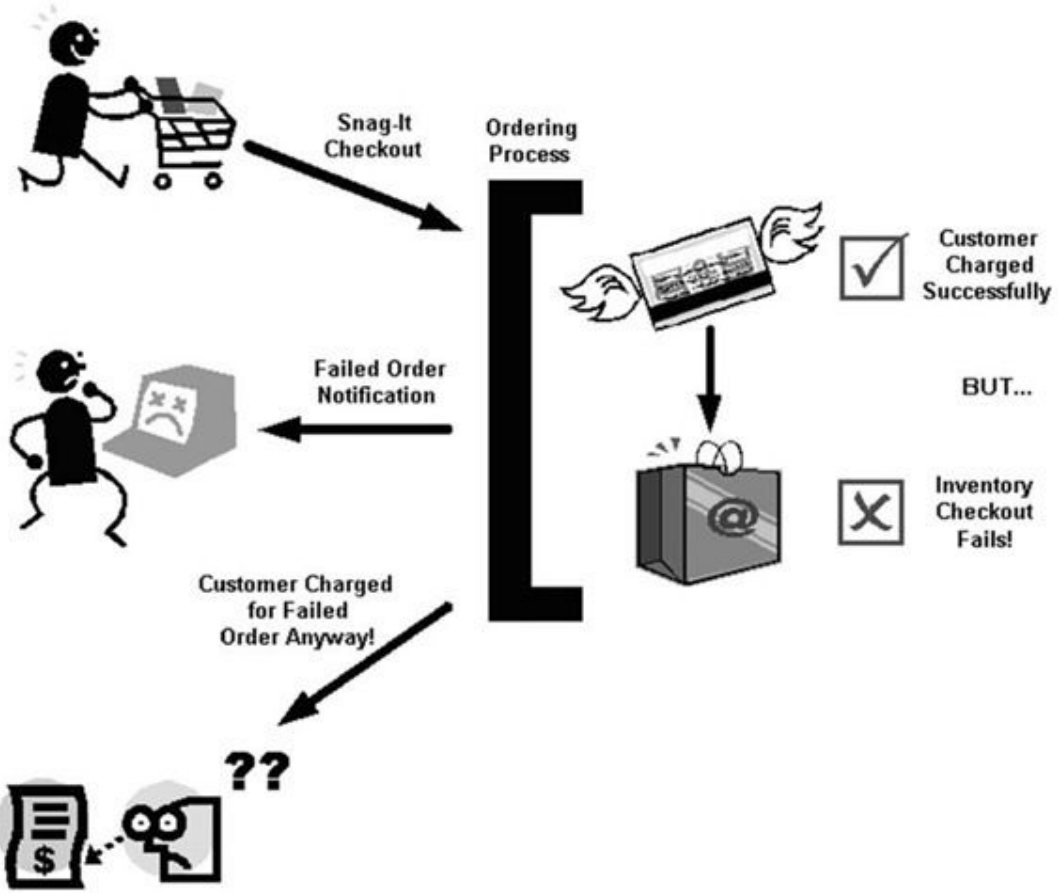


Image source: http://what-when-how.com/wp-content/uploads/2011/06/tmp2F410_thumb.jpg

Transactions

A transaction is an atomic unit of work that should either be completed in its entirety or not done at all.

A transaction should either:

- ***COMMITTED***: Effect is recorded permanently in the database if successful

- ***ABORTED***: Does not have any effect in the database .
  - If some operations are completed before the transaction failed, those operations should be undone (or ROLLBACKed).

Types of Failures

Transactions may fail due to various types of issues

- A computer failure
- A transaction or system error
- Local errors or exception conditions decided by the transactions
- Concurrency control enforcement
- Disk failure
- Physical problems and catastrophes

Transaction Processing Systems

Systems with large databases and hundreds of concurrent users executing database transactions.

Ex:

- Airline reservations
- Banking
- Credit card processing
- Online retail purchasing
- Stock markets
- Supermarket checkouts

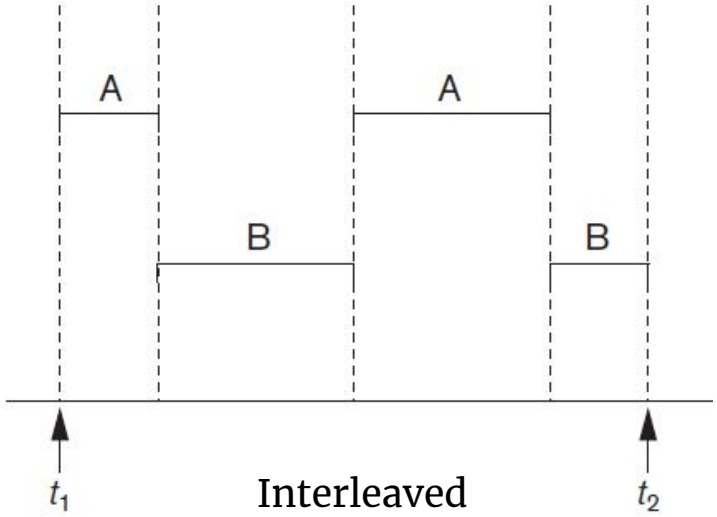
High Availability

Fast Response Time

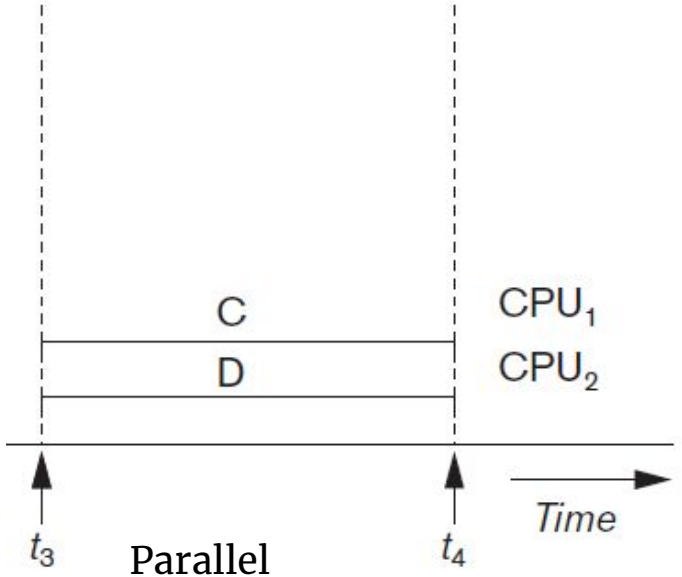
Large No. of Concurrent Users

Concurrent Transaction Processing

In a multi user DBMS, the stored data items are the primary resources that may be accessed concurrently by interactive users (or application programs) which are constantly retrieving information from and modifying the database.



Interleaved



Parallel

Database Access Operations

A single transaction may consist of any number of database operations.

Depending on the operations involved, transactions can be ***read-only transactions*** or ***read-write transactions***.

Transaction processing concepts are independent of the data item granularity.

- read_item(X): Reads a database item named X into a program variable X.
- write_item(X): Writes the value of program variable X into the database item named X.

Concurrent Transactions – Example

X and Y are number of bookings of two flights.

Transaction 1: Transfer N seat booking from the flight X to Y

Transaction 2: Reserve M seats from the flight X

T1

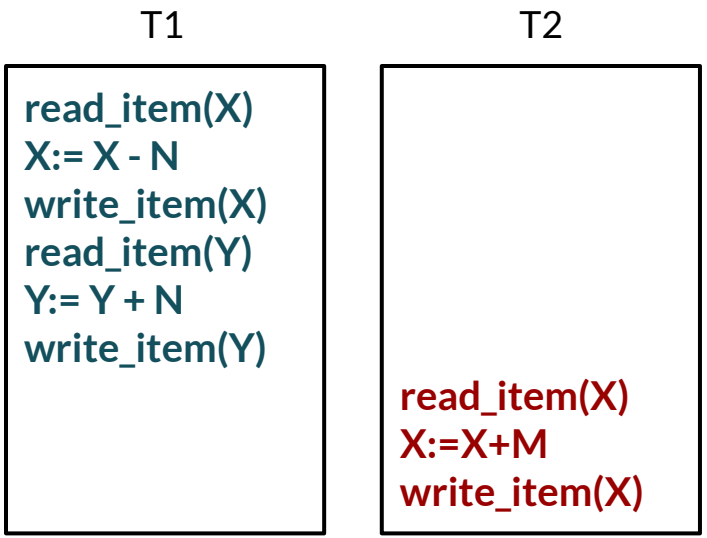
```
read_item(X)
X:= X - N
write_item(X)
read_item(Y)
Y:= Y + N
write_item(Y)
```

T2

```
read_item(X)
X:=X+M
write_item(X)
```

Concurrent Transactions

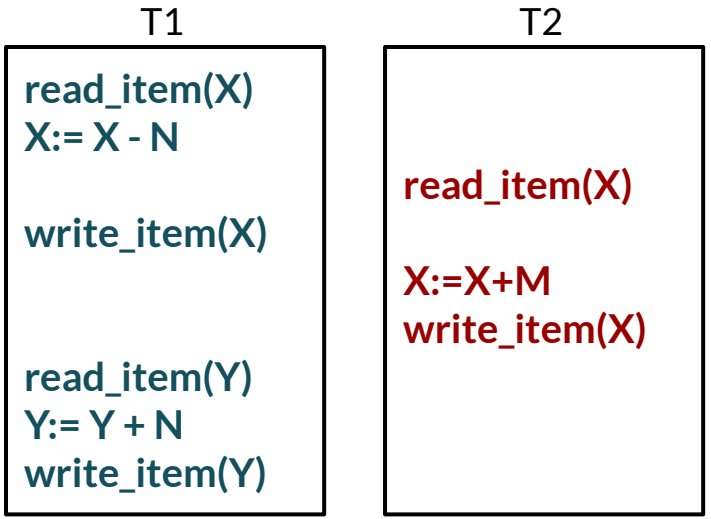
X = 80, Y =100, N = 5 and M = 4



X = ?
Y = ?

Concurrent Transactions – Lost Update Problem

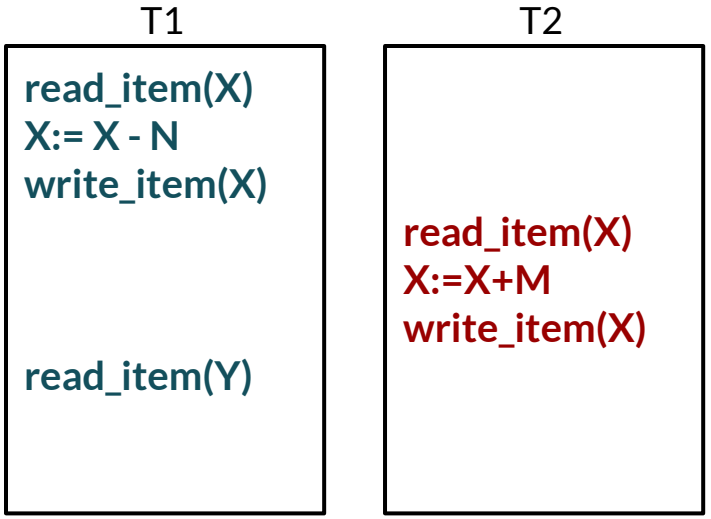
X = 80, Y =100, N = 5 and M = 4



X = ?
Y = ?

Concurrent Transactions – Temporary Update Problem

X = 80, Y =100, N = 5 and M = 4

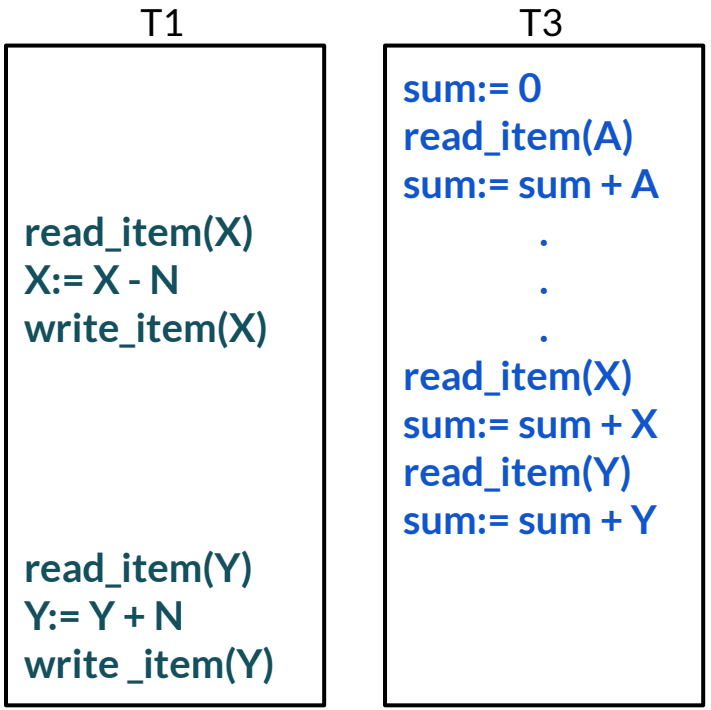


X = ?
Y = ?

T1 fails when performing operation read_item(Y)

Concurrent Transactions – Incorrect Summary Problem

X = 80, Y =100, N = 5 and M = 4



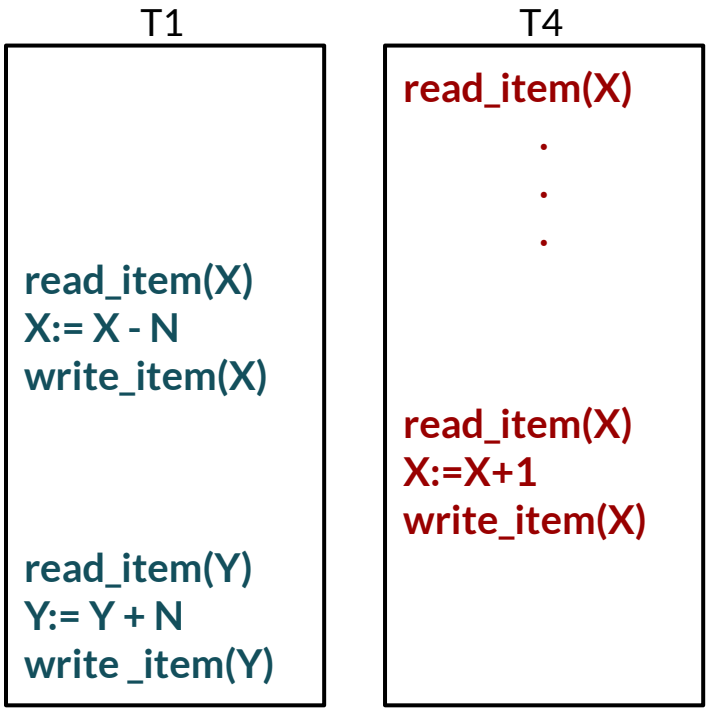
T3 calculate total number of reservations on all flights.

X = ?

Y = ?

Concurrent Transactions – Unrepeatable Read Problem

X = 80, Y =100, N = 5 and M = 4

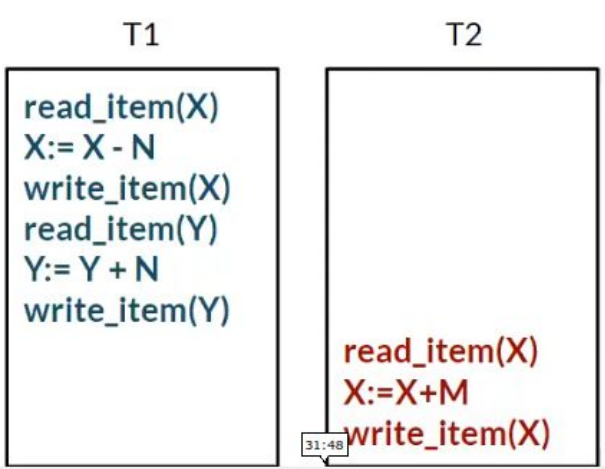


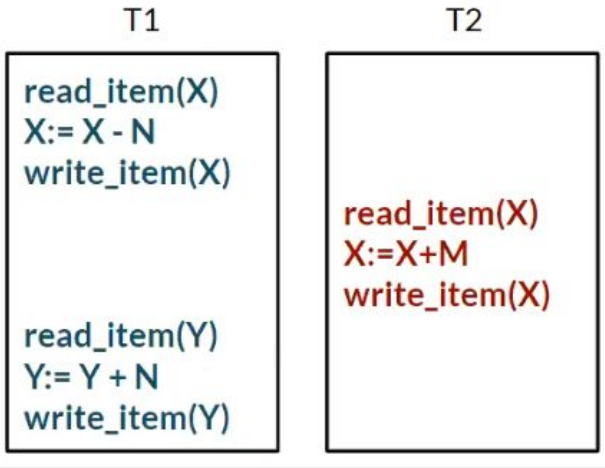
T4 transaction reads X and do some operations and reserves a seat in X

X = ?
Y = ?

Concurrency Goal

- Execute sequence of SQL statements in a way to appear as they are run in isolation
  - Method 1-Run in isolation
  - Method 2- Allow concurrency when possible





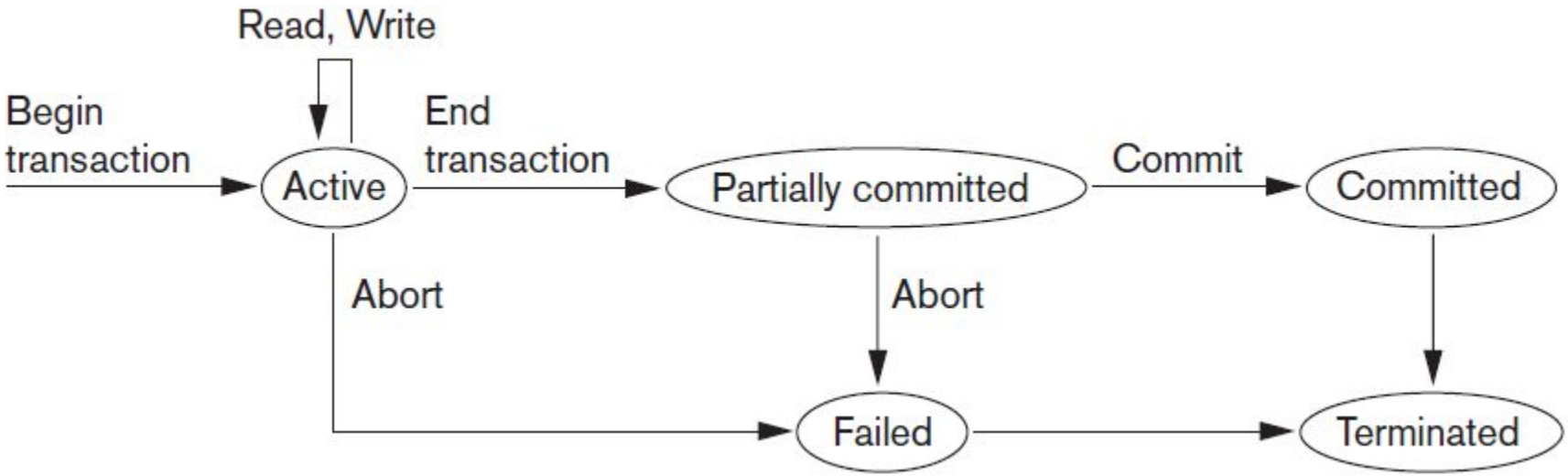
Transaction States

For recovery purposes system DBMS needs to keep track of operations:

- BEGIN_TRANSACTION
- READ or WRITE
- END_TRANSACTION
- COMMIT_TRANSACTION
- ROLLBACK (or ABORT)

The transaction information that is maintained in system tables while the transaction has been running is removed when the transaction terminates.

Transaction States



Logging Transactions

System log keep track of all transaction operations that affect the values of database items and other transaction information.

Depending on the recovery protocols, required log entries and their contents will be different.

Examples of log records (t is the transaction id) :

- [start-transaction, T]
- [write_item, T, X, old-valus, new_value]
- [read_item, T, X]
- [commit, T]
- [abort, T]

Commit Point

A transaction 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.

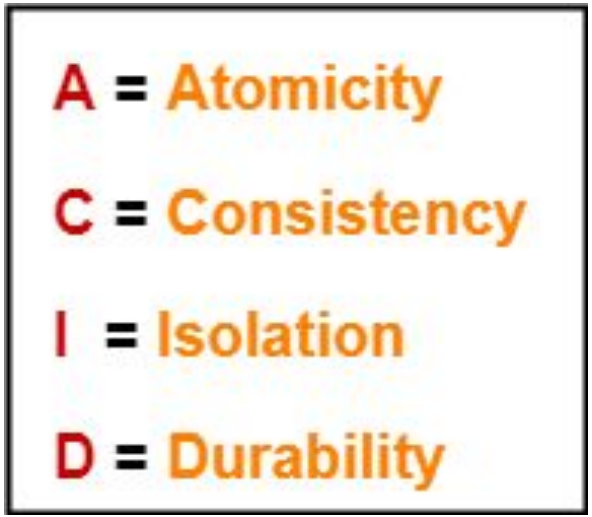
Beyond the commit point, the transaction is said to be committed, and its effect must be permanently recorded in the database.

The transaction then writes a commit record [commit, T] into the log.

To ensure recoverability, in case of main memory contents are lost, force-writing the log buffer to disk before committing a transaction.

properties of transactions

properties of transactions



Levels of Isolation

	Dirty Read	Non-repeatable read	phantoms
Read Uncommitted			
Read Committed			
Repeatable Read			
Serializable			

Problem of Dirty read

T1	T2
w(x)	
	R(x)
commit	

Read from an uncommitted transaction

Problem of non-repeatable read

T1	T2
R(x) =100	
	w(x)=200
R(x) =200	

Do not get the value repeated

Problem of phantoms

e_id	name	dept_id	salary
122	Sirimalee	21	23000
534	Sanath	43	66000
533	Fathima	5	97000
446	Krishanth	15	43000

Select * from employee where salary>45000;

Insert into employee (e_id, name, dept_id, salary) values (‘233’, ‘natali’,45,69000);

Levels of Isolation

	Dirty Read	Non-repeatable read	phantoms
Read Uncommitted			
Read Committed			
Repeatable Read			
Serializable