Topic:

Transaction Processing Simulation in ATM

Group:

1. Lê Thị Minh Nhật – 09070455
2. Lê Hoàng Ngọc Quỳnh – 09070460
3. Hà Lê Hoài Trung – 09070473

HCMC, Jan 2010

Outline

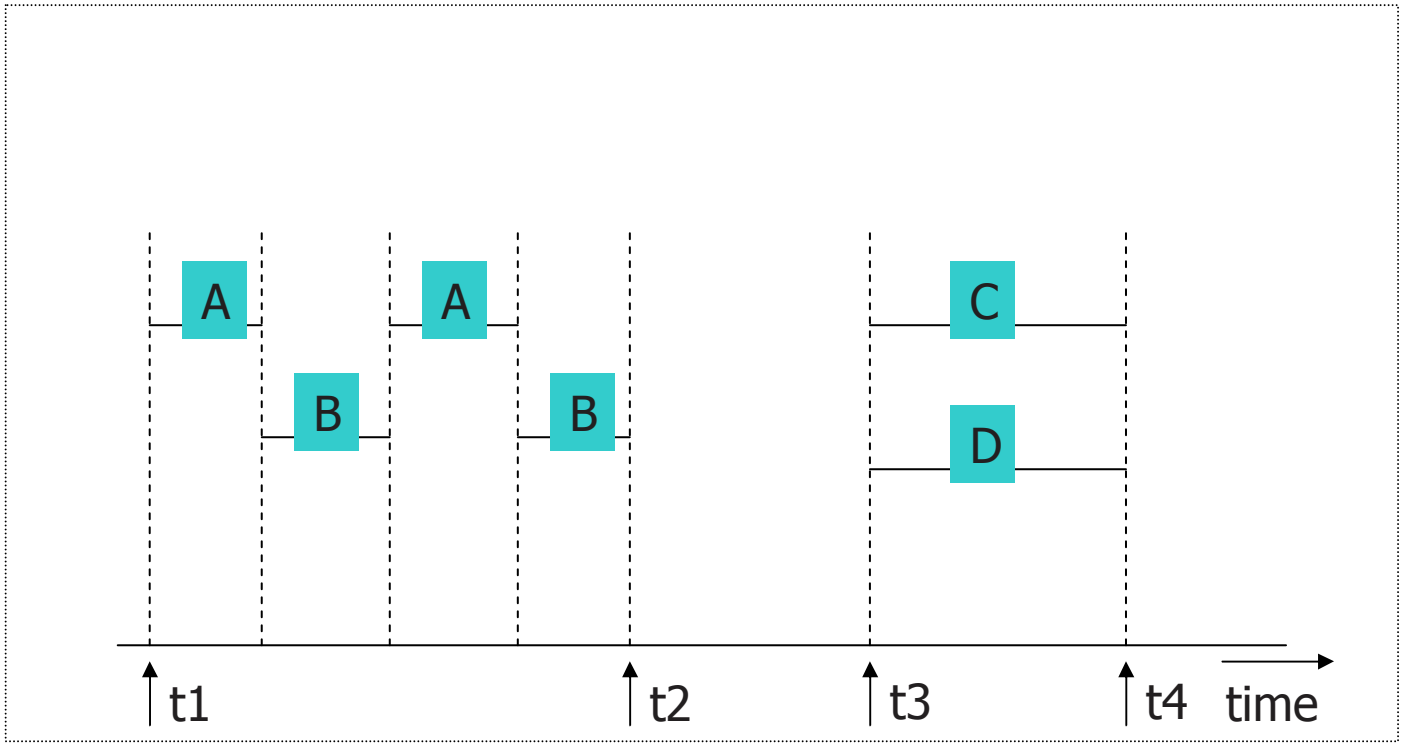
- Introduction to Transaction Processing
- Concurrency Control
- TP in action
- Q & A

Introduction to Transaction Processing

- **Single-User System:** At most one user at a time can use the system.
- **Multi-user System**: Many users can access the system concurrently.
- **Concurrency**
  - **Interleaved processing**: concurrent execution of processes is interleaved in a single CPU
  - **Parallel processing**: processes are concurrently executed in multiple CPUs.

Introduction to Transaction Processing

Interleaved processing and parallel processing

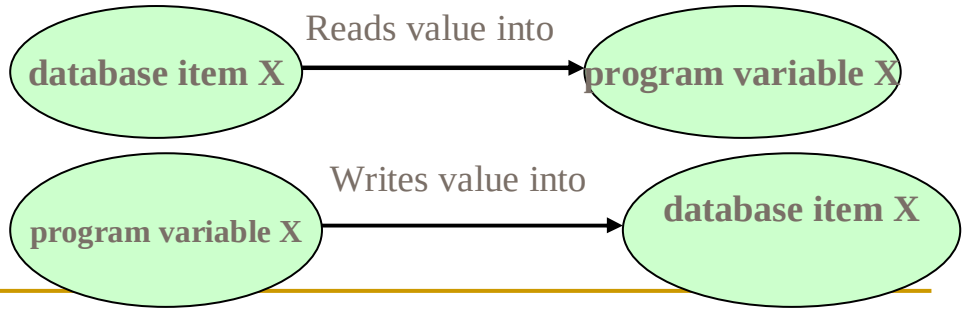


Interleaved processing Parallel processing

Introduction to Transaction Processing

- **A transaction:** logical unit of database processing that includes one or more access operations (read -retrieval, write - insert or update, delete).
- **The structure of a transaction:**

  - begin transaction
  - body: includes one or more operations
  - end transaction :
    - + Commit : successful
    - + Abort: unsuccessful (fail).
- **Basic operations are read and write**
  - **read_item(X)**:
  - **write_item(X)**:



Introduction to Transaction Processing

(a) T_1

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

(b) T_2

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

Two sample transactions.
(a) Transaction T1.
(b) Transaction T2.

Introduction to Transaction Processing

Basic characteristics of a transaction (ACID)

- Atomicity (nguyên tố)
- Consistency (nhất quán)
- Isolation (cô lập)
- Durability (bền vững)

Introduction to Transaction Processing

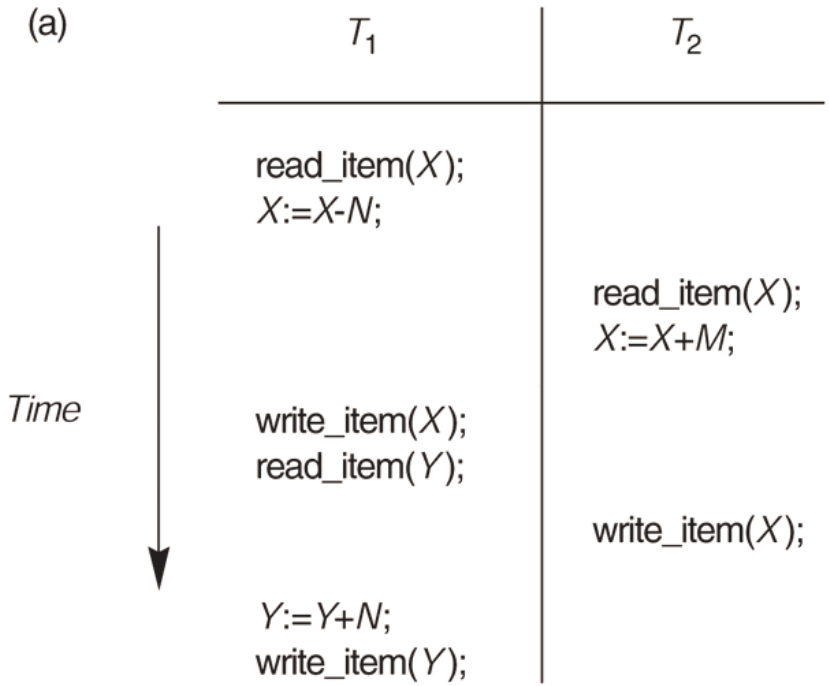
Common problems

1. The Lost Update Problem

This 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.

Introduction to Transaction Processing

The lost update problem example



Ex: X=100; N=10;M=20
X=110 OR 120?
WHY?

X has an incorrect value
Because its update by T1
is “lost”.

Introduction to Transaction Processing

Common problems (cont)

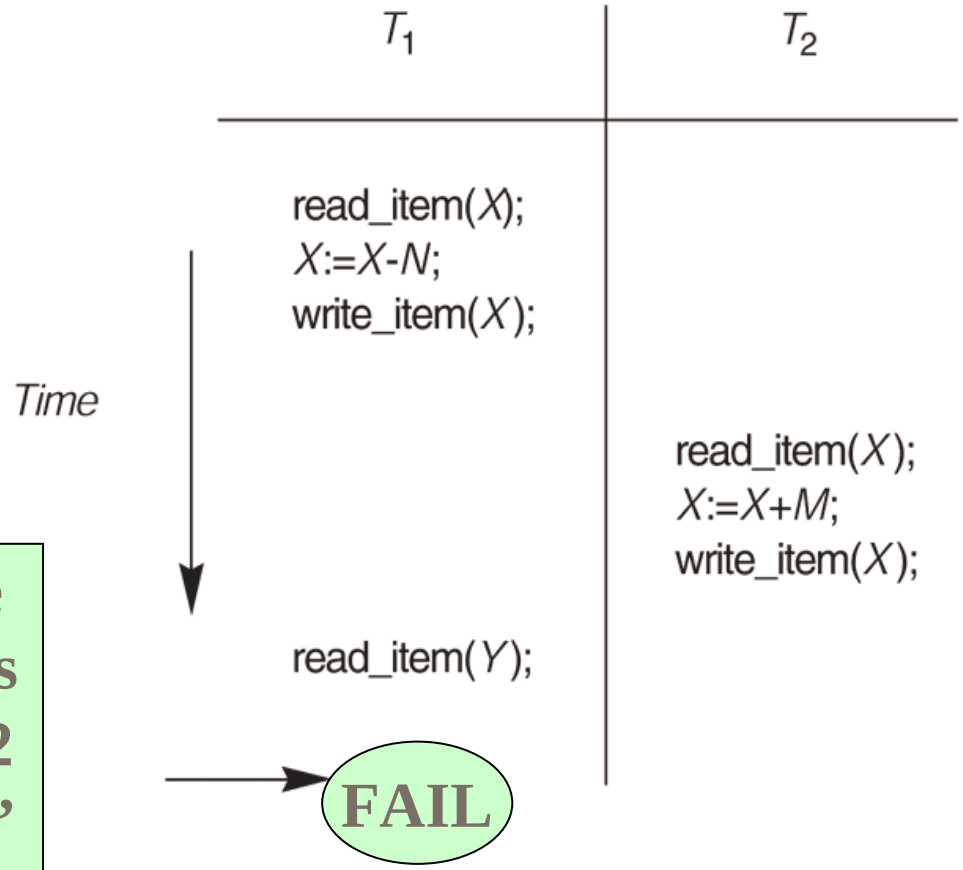
2. The Temporary Update (or Dirty Read)

This occurs when one transaction updates a database item and then the transaction fails for some reasons (see Section 17.1.4). The updated item is accessed by another transaction before it is changed back to its original value.

Introduction to Transaction Processing

The temporary update problem example

Ex: X=100; N=10;M=20
X=110 OR 120?



T1 fails and must change The value of X back to its Old value; meanwhile T2 has read the "temporary" Incorrect value of X

Introduction to Transaction Processing

Common problems (cont)

3. 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.

Introduction to Transaction Processing

The incorrect summary problem example

(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*);	

Value of Sum is incorrect
Why?

T3 reads X after N is subtracted
And reads Y before N is added

Introduction to Transaction Processing

Some reasons cause a transaction to fail

1. **A computer failure (system crash):** A hardware or software error occurs in the computer system during transaction execution. =>the contents of the computer's internal memory may be lost.

2. **A transaction error:** Some operations in the transaction may cause it to fail, such as integer overflow or division by zero.

`

3. **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.

Introduction to Transaction Processing

Some reasons cause a transaction to fail

4. **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 from that account, to be canceled.

   - a programmed abort in the transaction causes it to fail.

5. **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

Outline

- Introduction to Transaction Processing
- Concurrency Control
- TP in action
- Q & A

Concurrency Control

- Purpose of Concurrency Control
- Two-Phase Locking Techniques
- Types of lock
- Deadlock

Concurrency Control

Purpose of Concurrency Control

- Example: 1 account, in same time:
  - Withdraw
  - Add

X = 100, N = 20, M = 30.

X = ? 80, 130, 110

(a)

T_1
read_item (X);
X:=X-N;
write_item (X);

(b)

T_2
read_item (X);
X:=X+M;
write_item (X);

Concurrency Control

Purpose of Concurrency Control

- To enforce Isolation (through mutual exclusion) among conflicting transactions.
- To preserve database consistency through consistency preserving execution of transactions.
- To resolve read-write and write-write conflicts.

Concurrency Control

- Purpose of Concurrency Control
- Two-Phase Locking Techniques
- Types of lock
- Deadlock

Concurrency Control

Two-Phase Locking Techniques

(a)

T_1
read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);

(b)

T_2
read_item (X);
X:=X+M;
write_item (X);

Concurrency Control

Two-Phase Locking Techniques

- Locking is an operation which secures permission to Read or permission to Write a data item for a transaction.
  - *Example*: Lock (X). Data item X is locked in behalf of the requesting transaction.
- Unlocking is an operation which removes these permissions from the data item.
  - *Example*: Unlock (X). Data item X is made available to all other transactions.
- Lock and Unlock are Atomic operations.

Concurrency Control

Two-Phase Locking Techniques

- Lock Manager: Managing locks on data items.
- Lock table: Lock manager uses it to store the identify of transaction locking a data item, the data item, lock mode and pointer to the next data item locked. One simple way to implement a lock table is through linked list.

Transaction ID	Data item id	lock mode	Ptr to next data item
T1	X1	Read	Next

Concurrency Control

- Purpose of Concurrency Control
- Two-Phase Locking Techniques
- Types of lock
- Deadlock

Concurrency Control

Types of lock

- Binary locks
- Shared/Exclusive (or Read/ Write) Locks
- Conversion of lock

Concurrency Control

Binary locks

The following code performs the lock operation:

```
B:if LOCK (X) = 0 (*item is unlocked*)
  then LOCK (X) ← 1 (*lock the item*)
  else
    wait (until lock (X) = 0) and
         the lock manager wakes up the
         transaction);
    goto B
  end;
```

Concurrency Control

Binary locks

The following code performs the unlock operation:

```
LOCK (X) ← 0 (*unlock the item*)
  if any transactions are waiting then
    wake up one of the waiting the
  transactions;
```

Concurrency Control

Types of lock

- Binary locks
- Shared/Exclusive (or Read/Write) Locks
- Conversion of lock

Concurrency Control

Shared/Exclusive Locks

- Two locks modes (a) shared (read) and (b) exclusive (write)
  - Shared mode: shared lock (X). More than one transaction can apply share lock on X for reading its value but no write lock can be applied on X by any other transaction.
  - Exclusive mode: Write lock (X). Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X.

Concurrency Control

Shared/Exclusive Locks

	Read	Write
Read	Y	N
Write	N	N

Concurrency Control

Shared/Exclusive Locks

- Clock table : Each record in the lock table have 5 fields :

data item name	LOCK	no_of_read	locking_transactior	pointer

Concurrency Control

Shared/Exclusive Locks

The following code performs the read lock operation:

```
B: if LOCK (X) = “unlocked” then
begin LOCK (X) ← “read-locked”;
  no_of_reads (X) ← 1;
end
else if LOCK (X) ← “read-locked” then
        no_of_reads (X) ← no_of_reads (X) +1
    else begin wait (until LOCK (X) = “unlocked”
  and
       the lock manager wakes up the transaction);
       go to B
    end;
```

Concurrency Control

Shared/Exclusive Locks

The following code performs the write lock operation:

```
B: if LOCK (X) = “unlocked”
     then LOCK (X) ← “write-locked”;
    else
      begin wait (until LOCK (X) = “unlocked”
      and the lock manager wakes up the
  transaction);
       go to B
    end;
```

Concurrency Control

Shared/Exclusive Locks

The following code performs the unlock operation:

```
if LOCK (X) = “write-locked” then
begin LOCK (X) ← “unlocked”;
   wakes up one of the transactions, if any
end
else if LOCK (X) ← “read-locked” then
  begin
        no_of_reads (X) ← no_of_reads (X) -1
        if  no_of_reads (X) = 0 then
        begin
            LOCK (X) = “unlocked”;
         wake up one of the transactions, if any
        end
  end;
```

Concurrency Control

- Purpose of Concurrency Control
- Two-Phase Locking Techniques
- Types of lock
- Deadlock

Concurrency Control

Dealing with Deadlock

T'1	**T'2**	
read_lock (Y);		*T1 and T2 did follow two-phase*
read_item (Y);		*policy but they are deadlock*
	read_lock (X);	
	read_item (X);	
write_lock (X);		
(waits for X)	write_lock (Y);	
	(waits for Y)	

Deadlock (T'1 and T'2)

Concurrency Control

Deadlock prevention

A transaction locks all data items it refers to before it begins execution. This way of locking prevents deadlock since a transaction never waits for a data item. The conservative two-phase locking uses this approach.

Concurrency Control

Deadlock detection and resolution

In this approach, deadlocks are allowed to happen. The scheduler maintains a wait-for-graph for detecting cycle. If a cycle exists, then one transaction involved in the cycle is selected (victim) and rolled-back.

A wait-for-graph is created using the lock table. As soon as a transaction is blocked, it is added to the graph. When a chain like: Ti waits for Tj waits for Tk waits for Ti or Tj occurs, then this creates a cycle. One of the transaction of the cycle is selected and rolled back.

Concurrency Control

Deadlock avoidance

There are many variations of two-phase locking algorithm. Some avoid deadlock by not letting the cycle to complete. That is as soon as the algorithm discovers that blocking a transaction is likely to create a cycle, it rolls back the transaction. Wound-Wait and Wait-Die algorithms use timestamps to avoid deadlocks by rolling-back victim.

Outline

- Introduction to Transaction Processing
- Concurrency Control
- TP in action
- Q & A

TP in action

- TP concepts review
- Transaction in MySQL
- ATM simulation

TP in action

TP Concepts review

- Common issues:
  - Lost update
  - Uncommitted dependency
  - The Incorrect Summary
- Transaction definition (ACID)
- Concurrency control
  - Locking
  - Timestamp
  - Multiversion
  - Validation/certification (optimistic)
- Recovery

TP in action

Transaction in MySQL

- ACID rules
- Isolation levels
  - READ UNCOMMITTED
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE
- Locking mechanism: Table/Page/Row lock

TP in action

Transaction in MySQL (cont'd)

- Concurrency control
  - Full support with InnoDB
  - Locking (row level – reduce memory usage)
    - Shared (read)
    - Exclusive (write)
    - Intention
      - Intention shared (IS): **Select…lock in share mode**
      - Intention exclusive (IX): **Select …for update**

    (SELECT structure)
  - Multiversion

TP in action

Transaction in MySQL (cont'd)

- Notes:
  - Autocommit (default), or
  - Specify explicitly
    - `START TRANSACTION/BEGIN`
    - `COMMIT/ROLLBACK`

TP in action

ATM (automated teller machine)

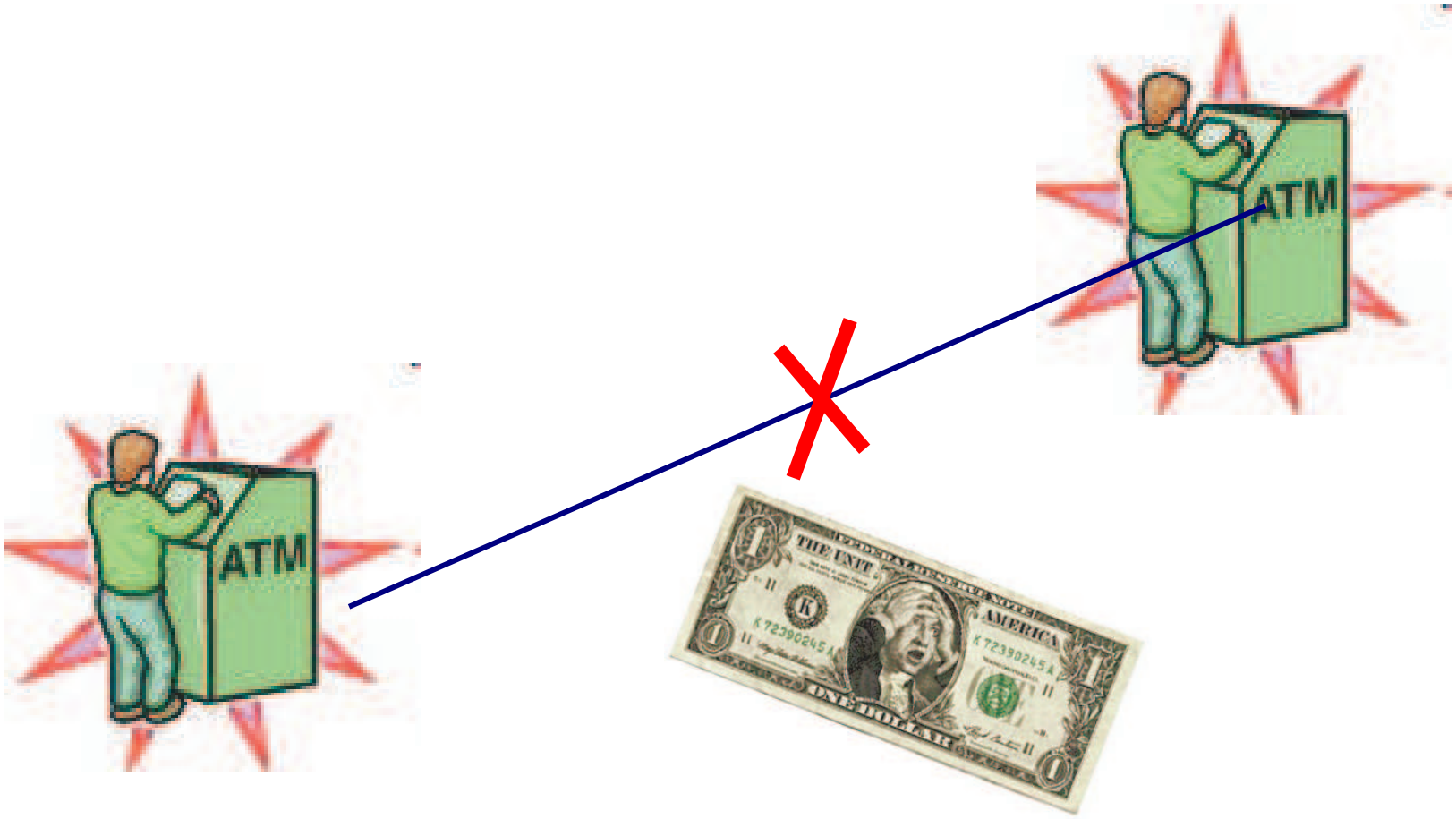
TP in action

ATM activities

- Query
- Withdraw
- Transferring
  - Account
  - Online payment: shopping, booking, etc.
  - And so on

TP in action

Risks in transaction



TP in action

Solution: Transaction

- Ensure the user got the money (withdraw)
- Both involved accounts balanced (credited/deducted in transferring)
- Moreover: concurrency control with multi-user' activities

TP in action

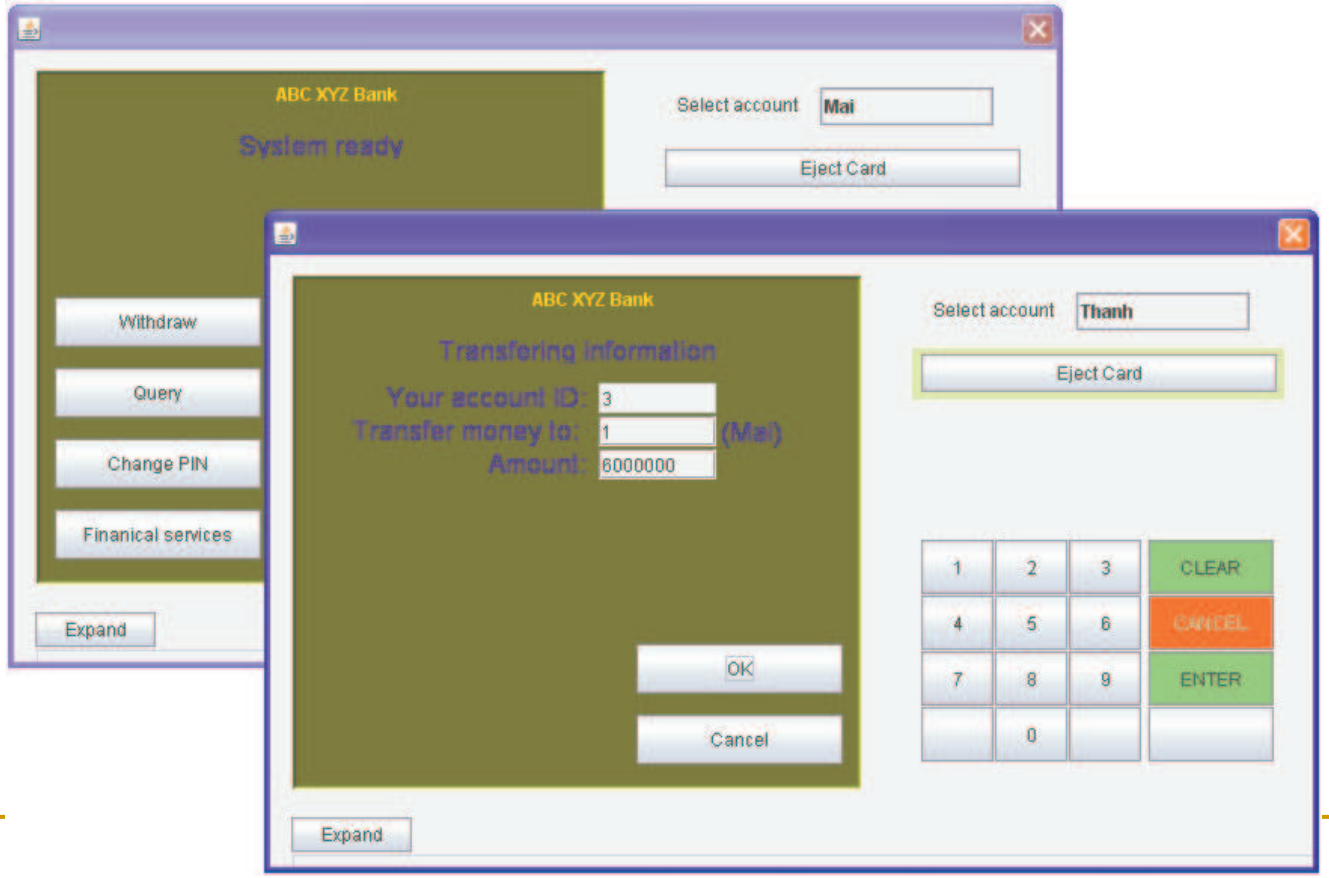
Demo

- Withdraw money

TP in action

Demo

- Transferring between 2+ ATM clients



TP in action

ATM #1	ATM #2
START TRANSACTION	…
Lock in shared mode	START TRANSACTION
…	Possible to query
…	Wait for withdraw…
If Withdraw -> deadlock	…
COMMIT to end tst.	Lock taken. Withdraw
	COMMIT

TP in action

- **Notes**
  - Invisible during transaction
  - While ATM #1 takes the lock, all other ATMs have to wait for it to release if they need to update the account in use (or timed out)
  - Deadlock occurs, then transaction gets terminated.
  - Must COMMIT to ensure all changes saved.

References

- Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database systems, 4th edition, chapter 17,18,19
- Concurrency Control: How It Really Works, MySQL Conference, April 2009
- Philip A. Bernstein, Eric Newcomer, Principles of Transaction Processing, 2nd edition
- http://en.wikipedia.org/wiki/Transaction_processing

Q & A

Thanks for your attention!