



# HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
# SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY


HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# Lesson10
# Transaction Management

## Concurrence Control

# Learning objectives

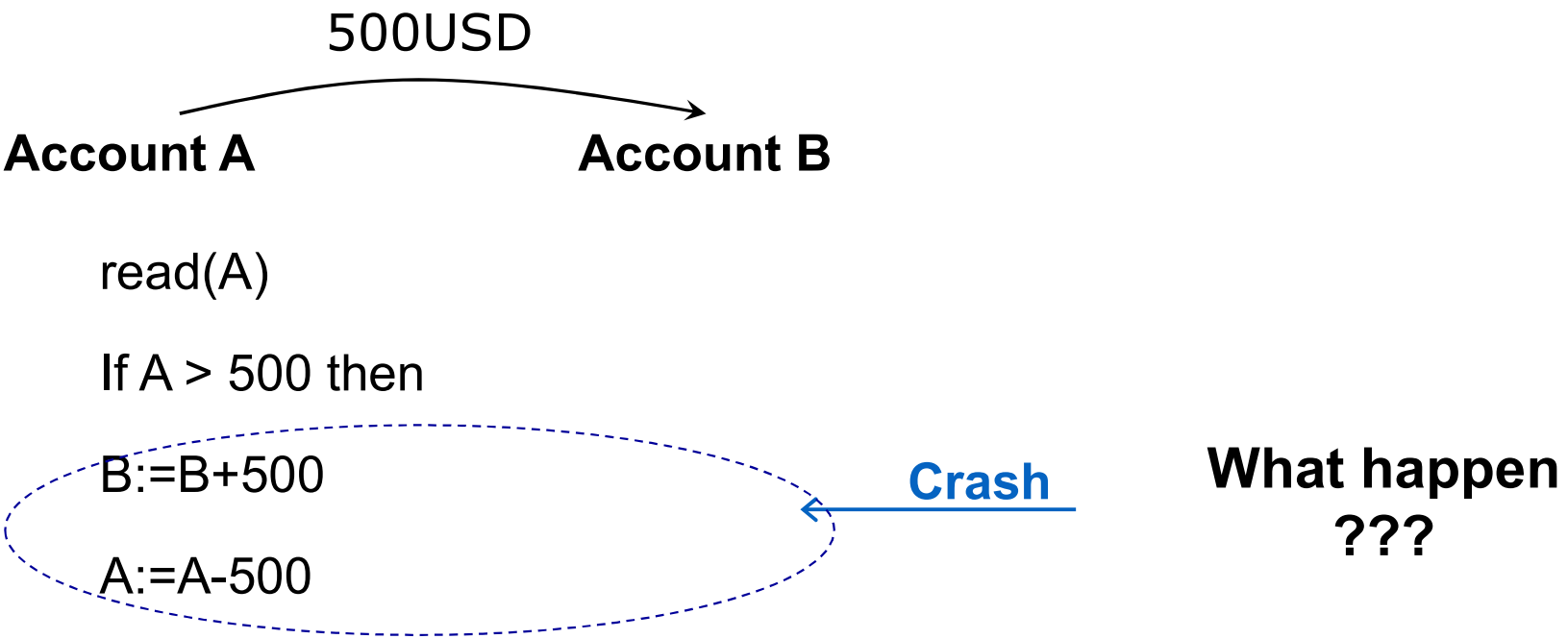
- ***Upon completion of this lesson, students will be able to:***

1. Understand main concepts of transaction
2. Be able to select a suitable transaction management strategy

# Outline

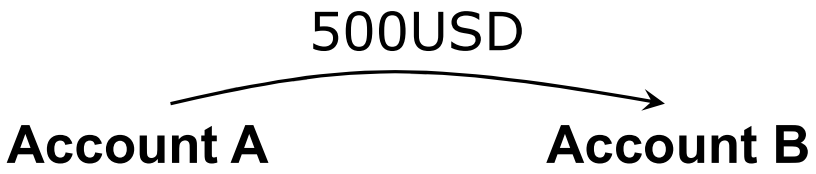
1. Transaction
2. ACID properties
3. Transaction Management Interface
4. Concurrence control
5. Isolation levels

# Example


500USD
Account A
Account B
read(A)
If A > 500 then
B:=B+500
A:=A-500
Crash
What happen
???

# 1. Transaction

- A sequence of read and write operations on data items that logically functions as one unit of work
  - Ensuring data integrity and correctness




read(A)

If A > 500 then

B:=B+500

A:=A-500

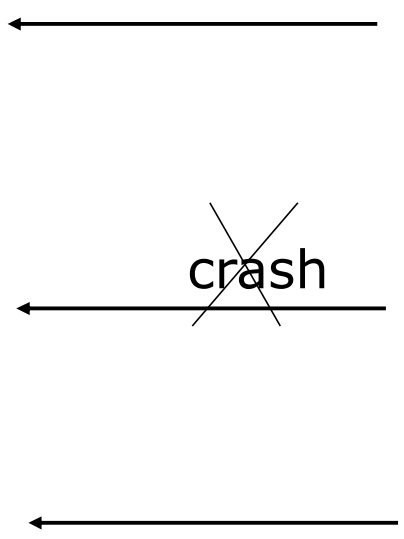
# 2. ACID Properties

2.1. Atomicity
2.2. Consistency
2.3. Isolation
2.4. Durability

# 2.1. Atomicity

- Guarantee that either all of the tasks of a transaction are performed or none of them are.

```
T: Read(A,t1);
   If t1 > 500 {
    Read(B,t2);
    t2:=t2+500;
    Write(B,t2);
    t1:=t1-500;
    Write(A,t1);
   }
```



# 2.2. Consistency

- Ensures that the DB remains in a consistent state before the start of the transaction and after the transaction is over.

```
T: Read(A,t1);                    <---- A+B = C
   If t1 > 500 {
    Read(B,t2);
    t2:=t2+500;
    Write(B,t2);
    t1:=t1-500;
    Write(A,t1);
   }                              <---- A+B = C
```

# 2.3. Isolation

- Ability of the application to make operations in a transaction appear isolated from all other operations.

A= 5000, B= 3000

```
T: Read(A,t1);
   If t1 > 500 {
    Read(B,t2);
    t2:=t2+500;
    Write(B,t2);
    t1:=t1-500;
    Write(A,t1);
   }
```

← *T': A+B (= 5000+3500)*

← *(A+B = 4500+3500)*

# 2.4. Durability

- Guarantee that once the user has been notified of success, the transaction will persist, and not be undone.

A= 5000, B= 3000

```
T: Read(A,t1);
   If t1 > 500 {
    Read(B,t2);
    t2:=t2+500;
    Write(B,t2);
    t1:=t1-500;
    Write(A,t1);
   }
```

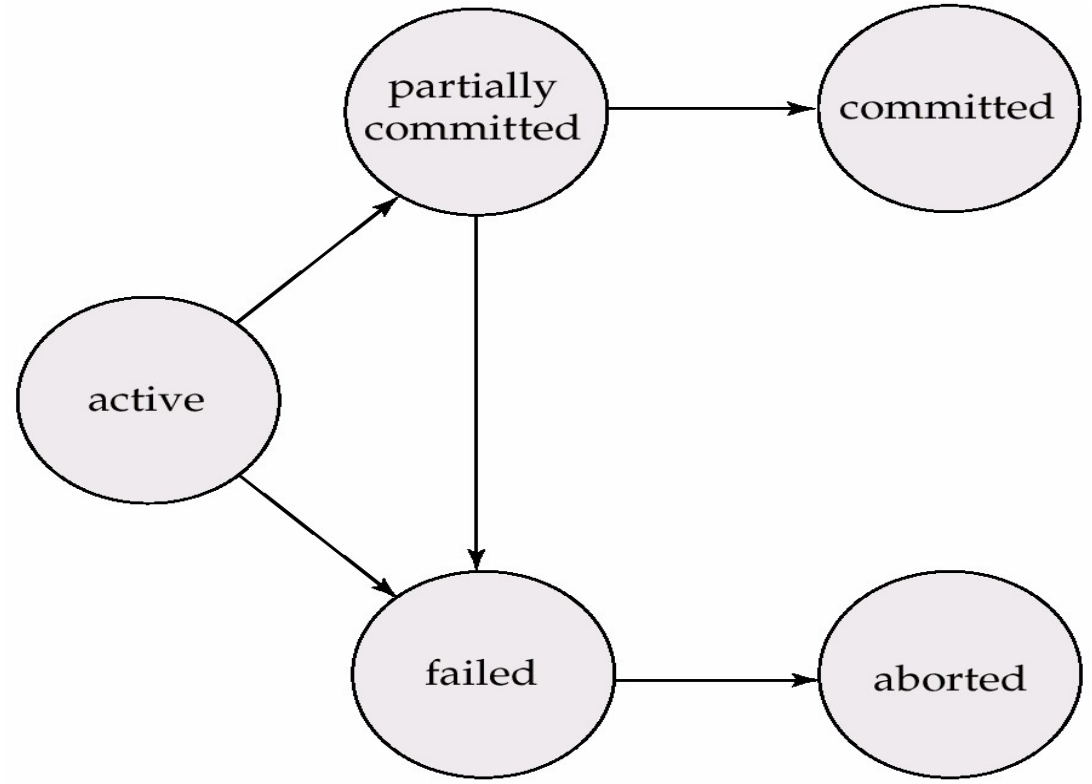
crash

A= 4500, B=3500

# 3. Transaction Management Interface

# 3.1. Transaction States

- Active
- Failed
- Aborted
- Commited
- Partially commited



# 3.1. Transaction States

- Active
- Begin Trans
- Commit ()
- Abort()
- Savepoint Save()
- Rollback (savepoint)

  (savepoint = 0 ==> Abort)

# 4. Concurrent control

4.1. Objective

4.2. Scheduling

4.3. Lock

# 4.1. Objective

- Ensures that database transactions are performed concurrently without the concurrency violating the data integrity.
- Guarantees that no effect of committed transactions is lost, and no effect of aborted (rolled back) transactions remains in the related database.
- Example

```
T0:  read(A);
     A := A -50;
     write(A);
     read(B);
     B := B + 50;
     write(B);
```

```
T1:  read(A);
     temp := A *0.1;
     A := A -temp;
     write(A);
     read(B);
     B := B + temp;
     write(B);
```

# 4.2. Scheduling

| $T_0$ | $T_1$ |
|---|---|
| read(A) | |
| A := A - 50 | |
| write(A) | |
| read(B) | |
| B := B + 50 | |
| write(B) | |
| | read(A) |
| | temp := A * 0.1 |
| | A := A -temp |
| | write(A) |
| | read(B) |
| | B := B + temp |
| | write(B) |

(1)

| $T_0$ | $T_1$ |
|---|---|
| | read(A) |
| | temp := A * 0.1 |
| | A := A -temp |
| | write(A) |
| | read(B) |
| | B := B + temp |
| | write(B) |
| read(A) | |
| A := A - 50 | |
| write(A) | |
| read(B) | |
| B := B + 50 | |
| write(B) | |

(2)

| $T_0$ | $T_1$ |
|---|---|
| read(A) | |
| A := A - 50 | |
| | read(A) |
| | temp := A * 0.1 |
| | A := A -temp |
| | write(A) |
| | read(B) |
| write(A) | |
| read(B) | |
| B := B + 50 | |
| write(B) | |
| | B := B + temp |
| | write(B) |

(3)

# 4.2. Scheduling

- A schedule of a set of transactions is a linear ordering of their actions
  - e.g. for the simultaneous deposits example:

    R1(X) R2(X) W1(X) W2(X)
- A serial schedule is one in which all the steps of each transaction occur consecutively
- A serializable schedule is one which is equivalent to some serial schedule

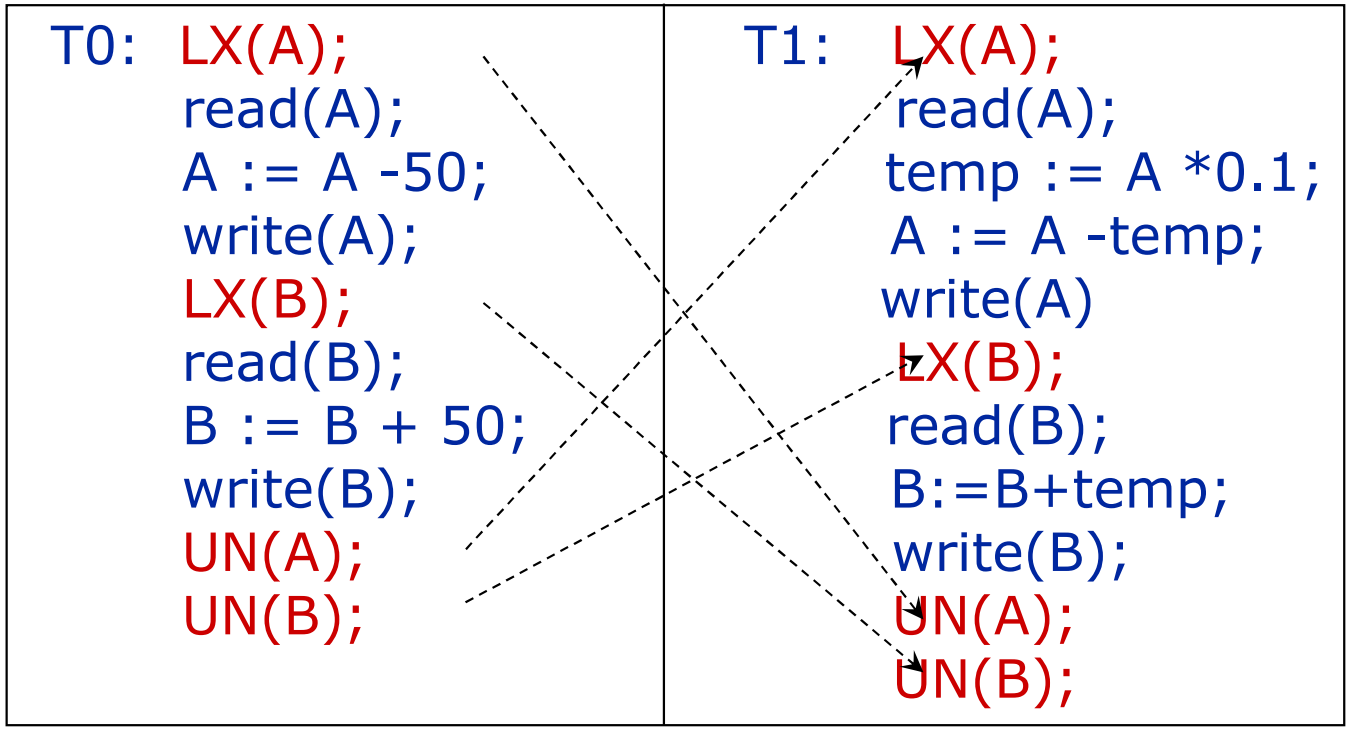
# 4.3. Lock

- Lock: a synchronization mechanism for enforcing limits on access to DB in concurrent way.
  - one way of enforcing concurrency control policies
- Lock types
  - Shared lock (LS) readable but can not write
  - Exclusive lock (LX): read and write
  - UN(D): unlock
- Compatibility

| | LS | LX |
|---|---|---|
| LS | true | false |
| LX | false | false |

# 4.3. Lock

- Example



# 5. Isolation levels

```
Set isolation level  <level>
```

- Read Uncommitted (No lost update)
- Exclusive locks for write operations are held for the duration of the transactions
- No locks for read
- Read Committed (No inconsistent retrieval)
- Shared locks are released as soon as the read operation terminates.
- Repeatable Read (no unrepeatable reads)
- Strict two phase locking

- Serializable (no phantoms)
  - Table  locking or index locking to avoid phantoms

# Summary

- Transaction
  - Sequence of actions
- ACID
  - Properties of a transaction
- Concurrence control
  - Mechanism allows multiple transactions accessing the same resource in the same time
- Isolation level
  - Defining the level DBMS must ensure data integrity and correctness in processing concurrent accesses

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict