

[illegible]

# Database System Concept (CSE 3103)

Lecture 08-Day 01

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

# Transaction Concept

- A **transaction** is a *unit* of program execution that accesses and possibly updates various data items.
- E.g., transaction to transfer \$50 from account A to account B:
  1. **read**(A)
  2.  $A := A - 50$
  3. **write**(A)
  4. **read**(B)
  5.  $B := B + 50$
  6. **write**(B)
- Two main issues to deal with:
  - Failures of various kinds, such as hardware failures and system crashes
  - Concurrent execution of multiple transactions

# Required Properties of a Transaction

- Transaction to transfer \$50 from account A to account B:
  1. **read**(A)
  2.  $A := A - 50$
  3. **write**(A)
  4. **read**(B)
  5.  $B := B + 50$
  6. **write**(B)
- **Atomicity requirement**
  - If the transaction fails after step 3 and before step 6, money will be “lost” leading to an inconsistent database state
    - Failure could be due to software or hardware
  - The system should ensure that updates of a partially executed transaction are not reflected in the database
- **Durability requirement** — once the user has been notified that the transaction has completed (i.e., the transfer of the \$50 has taken place), the updates to the database by the transaction must persist even if there are software or hardware failures.

## Required Properties of a Transaction (Cont.)

- **Consistency requirement** in above example:
  - The sum of A and B is unchanged by the execution of the transaction
- In general, consistency requirements include
  - Explicitly specified integrity constraints such as primary keys and foreign keys
  - Implicit integrity constraints
    - e.g., sum of balances of all accounts, minus sum of loan amounts must equal value of cash-in-hand
- A transaction, when starting to execute, must see a consistent database.
- During transaction execution the database may be temporarily inconsistent.
- When the transaction completes successfully the database must be consistent
  - Erroneous transaction logic can lead to inconsistency

## Required Properties of a Transaction (Cont.)

- **Isolation requirement** — if between steps 3 and 6 (of the fund transfer transaction) , another transaction **T2** is allowed to access the partially updated database, it will see an inconsistent database (the sum  $A + B$  will be less than it should be).

<b>T1</b>	<b>T2</b>
1. <b>read</b> (A)	
2. $A := A - 50$	
3. <b>write</b> (A)	
	read(A), read(B), print(A+B)
4. <b>read</b> (B)	
5. $B := B + 50$	
6. <b>write</b> (B)	

- Isolation can be ensured trivially by running transactions **serially**
  - That is, one after the other.
- However, executing multiple transactions concurrently has significant benefits, as we will see later.

# ACID Properties

A **transaction** is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data the database system must ensure:

- **Atomicity.** Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency.** Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation.** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
  - That is, for every pair of transactions  $T_i$  and  $T_j$ , it appears to  $T_i$  that either  $T_j$  finished execution before  $T_i$  started, or  $T_j$  started execution after  $T_i$  finished.
- **Durability.** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.