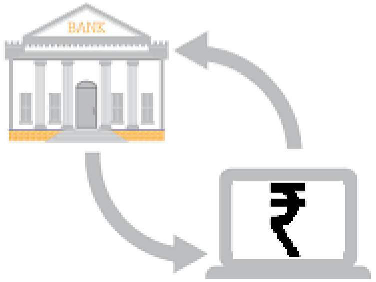
Transactions

Database Management Systems

Amrita Vishwa Vidyapeetham, Amritapuri

Transaction Concept

- **Transactions**: Collections of operations that form a single logical unit.
- A **transaction** is a *unit* of program execution that accesses and possibly updates various data items.
- Examples
  - Transfer money between bank accounts
  - ATM transactions
  - Book flight/ train via online reservations systems



- Application View (SQL View):

  Begin transaction

  Sequence of SQL statements (database access operations)

  End transaction

Transaction Example

Transaction to transfer ₹500 from account R to account S:

1. **read(*R*)**
2. *R* = *R* – 500
3. **write(*R*)**
4. **read(*S*)**
5. *S* = *S* + 500
6. **write(*S*)**

- Two main issues to deal with:
  - **Failures** of various kinds, such as hardware failures and system crashes
  - **Concurrent** execution of multiple transactions

Example of Fund Transfer

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**

1. **read**(*R*)
2. *R* := *R* – 500
3. **write**(*R*)
4. **read**(*S*)
5. *S*:= *S* + 500
6. **write**(*S*)

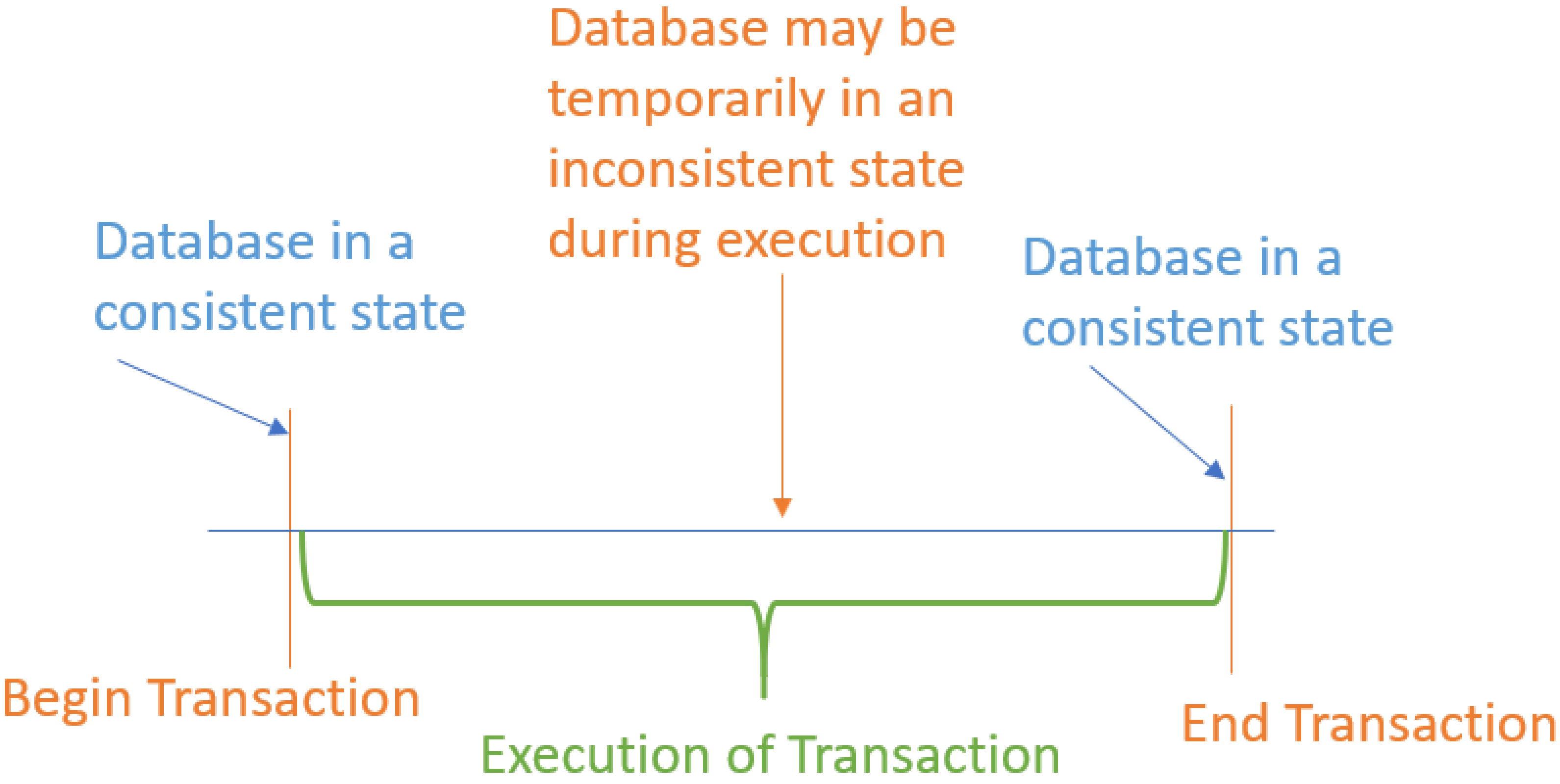
Example of Fund Transfer (Cont.)

Consistency requirement

- In example, the **sum of R and S is unchanged** by the execution of the transaction.
- In general, consistency requirements include
  - **Explicitly** specified **integrity constraints**
    - primary keys and foreign keys
  - **Implicit** integrity constraints
    - sum of balances of all accounts, minus sum of loan amounts must equal value of cash-in-hand
  - A transaction 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

1. **read**(*R*)
2. *R* := *R* – 500
3. **write**(*R*)
4. **read**(*S*)
5. *S*:= *S* + 500
6. **write**(*S*)

Consistency requirement



Example of Fund Transfer (Cont.)

Isolation requirement

- if between steps 3 and 6, another transaction *T2* is allowed to access the partially updated database, it will see an inconsistent database (the sum *R* + *S* will be less than it should be).

T1	T2
1. **read**(R)	
2. $R := R - 500$	
3. **write**(R)	
	read(R), read(S), print(R+S)
4. **read**(S)	
5. $S := S + 500$	
6. **write**(S)	

- Isolation can be ensured trivially by running transactions serially
  - that is, one after the other.
- However, executing multiple transactions concurrently has significant benefits

Example of Fund Transfer (Cont.)

Durability requirement

- once the user has been notified that the transaction has completed (i.e., the transfer of the ₹500 has taken place), the **updates to the database by the transaction must persist** even if there are software or hardware failures.

1. **read**(R)
2. *$R := R - 500$*
3. **write**(R)
4. **read**(S)
5. *$S := S + 500$*
6. **write**(S)

ACID Properties

- 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.