TRANSACTION

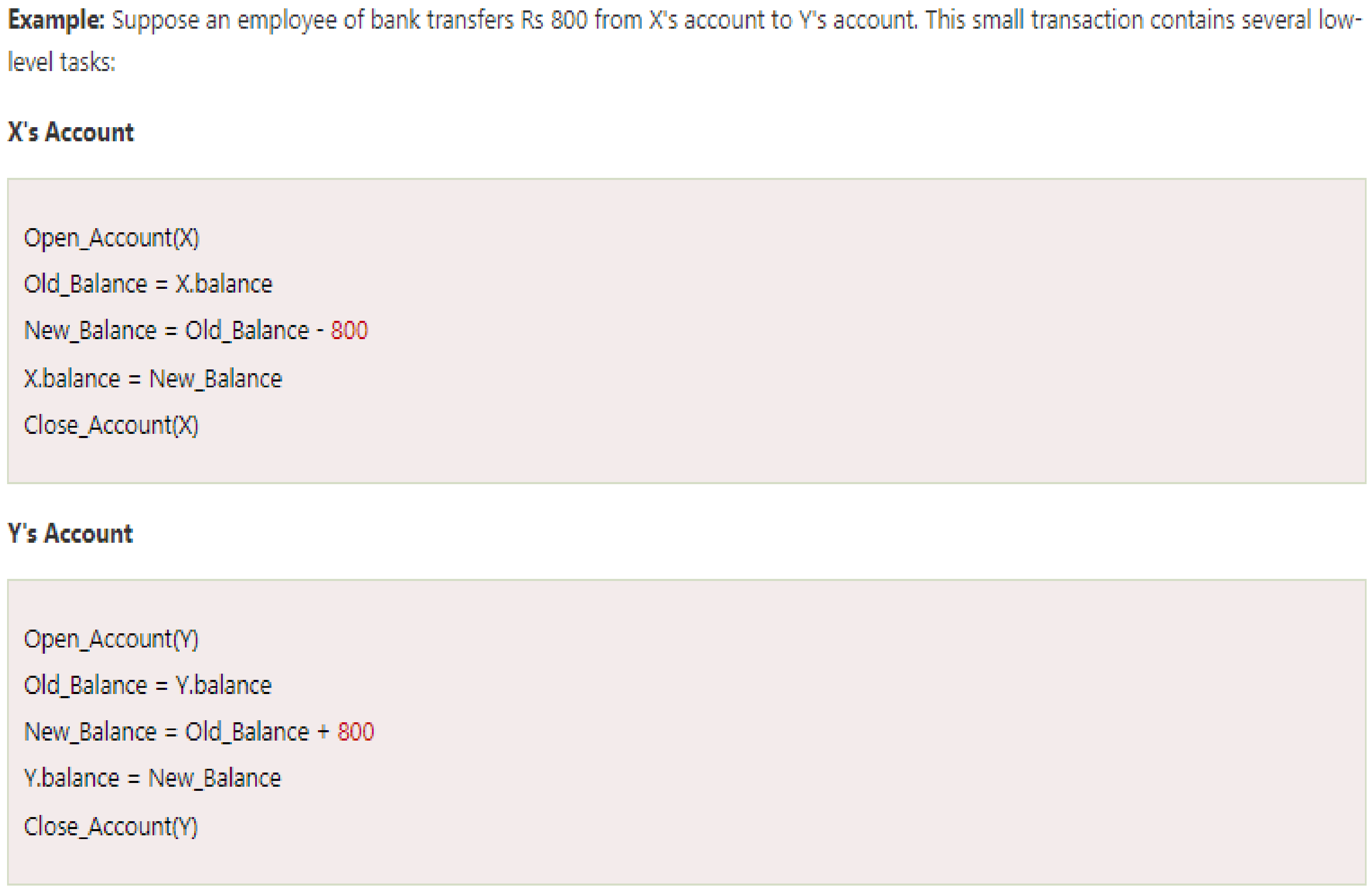
What is a transaction???

- A **transaction** is a logical unit of work of database processing that includes one or more database access operations.
- A transaction can be defined as an action or series of actions that is carried out by a single user or application program to perform operations for accessing the contents of the database.
- The operations can include retrieval(Read), insertion (Write), deletion and modification.
- A transaction must be either completed or aborted.

Different operations performed in a transaction

Operations	Descriptions
Read(x)	Read operation is used to read the value of X from the database and stores it in a buffer in main memory.
Write(X)	Write operation is used to write the value back to the database from the buffer.
Commit	It is used to save the work done permanently.
Rollback	It is used to undo the work done.

Example of a transaction



Example: Suppose an employee of bank transfers Rs 800 from X's account to Y's account. This small transaction contains several low-level tasks:

X's Account

```
Open_Account(X)
Old_Balance = X.balance
New_Balance = Old_Balance - 800
X.balance = New_Balance
Close_Account(X)
```

Y's Account

```
Open_Account(Y)
Old_Balance = Y.balance
New_Balance = Old_Balance + 800
Y.balance = New_Balance
Close_Account(Y)
```

Properties of Transaction

A transaction is having four properties. Such as-

i) Atomicity
ii) Consistency
iii) Isolation
iv) Durability

Atomicity

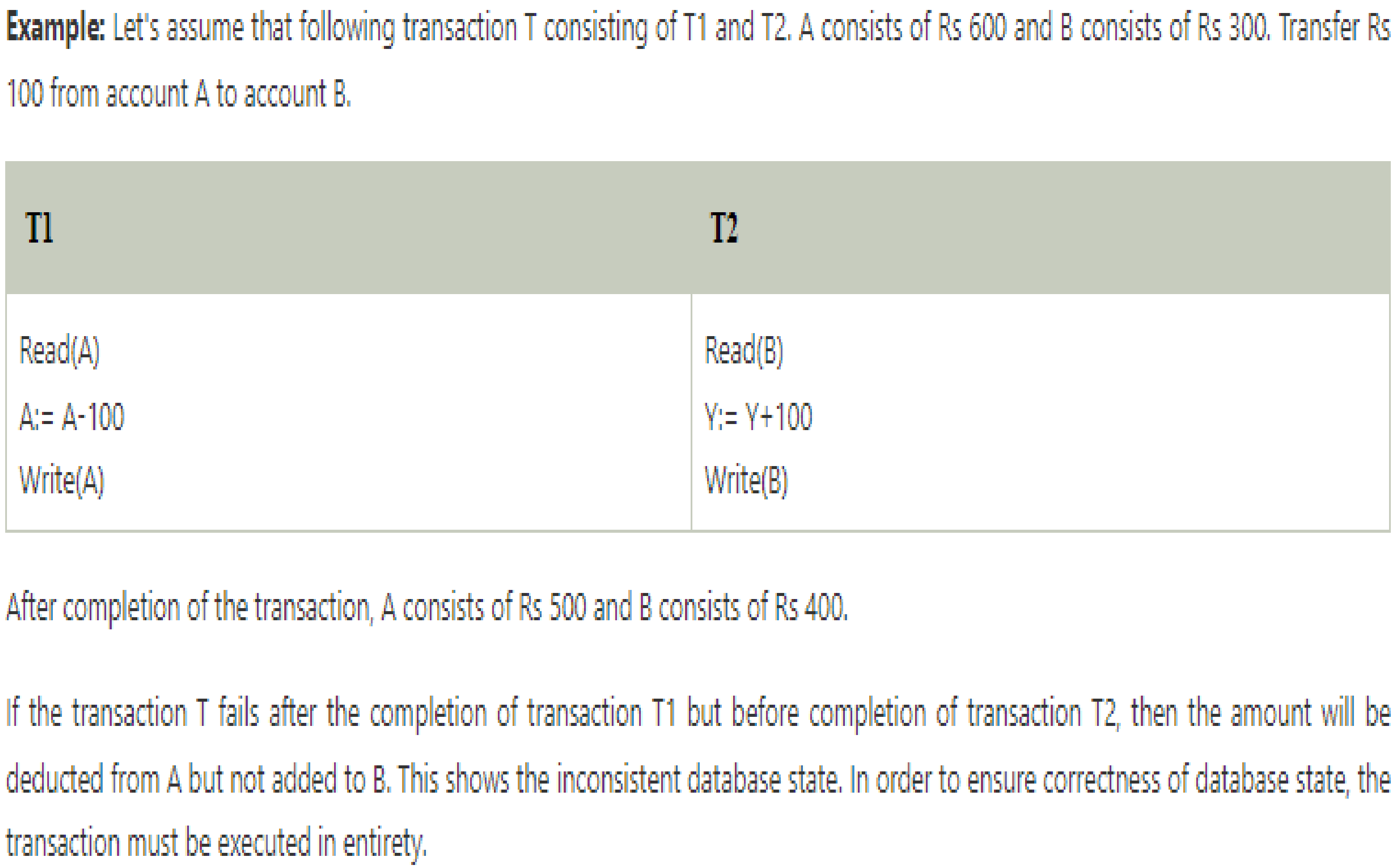
- Atomicity means all successful or none.
- It states that all operations of the transaction take place at once if not, the transaction is aborted.
- There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

Atomicity

Atomicity involves the following two operations.

- **Abort:** If a transaction aborts then all the changes made are not visible.
- **Commit:** If a transaction commits then all the changes made are visible.

Example of Atomicity



Example: Let's assume that following transaction T consisting of T1 and T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

T1	T2

Read(A)
A:= A-100
Write(A)
	Read(B)
	Y:= Y+100
	Write(B)

After completion of the transaction, A consists of Rs 500 and B consists of Rs 400.

If the transaction T fails after the completion of transaction T1 but before completion of transaction T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure correctness of database state, the transaction must be executed in entirety.

Consistency

- The integrity constraints are maintained so that the database is consistent before and after the transaction.
- The execution of a transaction will leave a database in either its prior stable state or a new stable state.
- The consistent property of database states that every transaction sees a consistent database instance.
- The transaction is used to transform the database from one consistent state to another consistent state.

Consistency

In the previous example, the total balance of A and B should be same before and after T occurs.

Total before T occurs = 600+300=900

Total after T occurs= 500+400=900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

Isolation

- It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
- In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.
- The concurrency control subsystem of the DBMS enforced the isolation property.

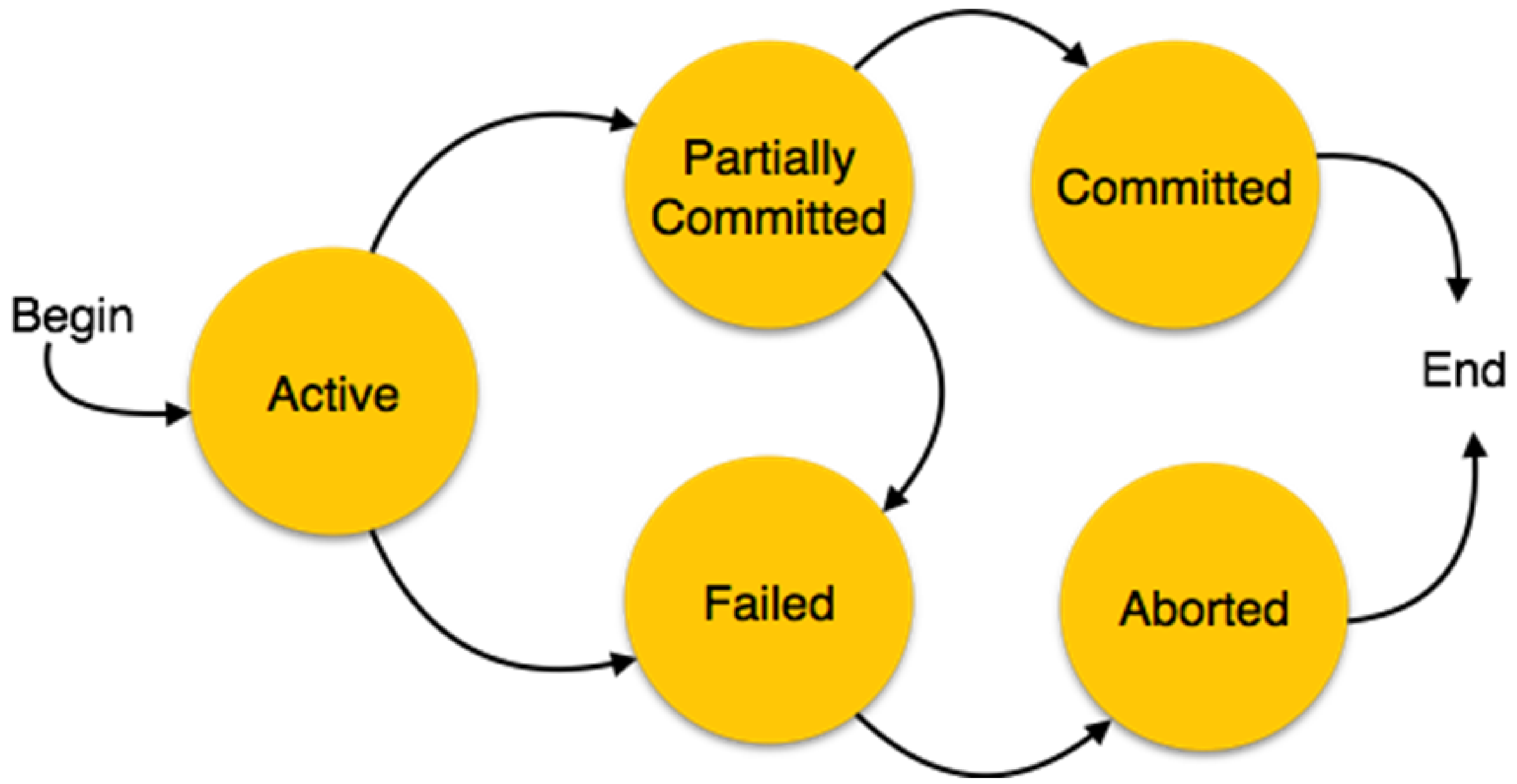
Isolation

- 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.
- No transaction will affect the existence of any other transaction.

Durability

- The database should be durable enough to hold all its latest updates even if the system fails or restarts.
- If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data.
- If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

States of Transaction



States of Transaction

- **Active State:**In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, but the data is not saved in disk, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

States of Transaction

- **Aborted** – If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts –
    - Re-start the transaction
    - Kill the transaction
- **Committed** – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Thank You