

# *Database Management System*

## *Transaction & Distributed Concurrency Control*

RAMNA SATTAR



# *Transaction*

## **Transaction:**

- The transaction is a set of logically related operation. It contains a group of tasks.
- A transaction is an action or series of actions. It is performed by a single user to perform operations for accessing the contents of the database.
- A transaction is a program including a collection of database operations, executed as a logical unit of data processing. The operations performed in a transaction include one or more of database operations like insert, delete, update or retrieve data. It is an atomic process that is either performed into completion entirely or is not performed at all.
- A transaction involving only data retrieval without any data update is called read-only transaction.



# *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)



# *Transaction*

## Operations of Transaction:

- **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.

Let's take an example to debit transaction from an account which consists of following operations:

1. **R(X);**
2. **X = X - 500;**
3. **W(X);**

Let's assume the value of X before starting of the transaction is 4000.

The first operation reads X's value from database and stores it in a buffer.

The second operation will decrease the value of X by 500. So buffer will contain 3500.

The third operation will write the buffer's value to the database. So X's final value will be 3500.



# Transaction

## Property of Transaction:

- ☐ Atomicity
- ☐ Consistency
- ☐ Isolation
- ☐ Durability

### Atomicity

means either all successful or none.

### Consistency

ensures bringing the database from one consistent state to another consistent state.  
ensures bringing the database from one consistent state to another consistent state.

### Isolation

ensures that transaction is isolated from other transaction.

### Durability

means once a transaction has been committed, it will remain so, even in the event of errors, power loss etc.

# *Transaction*

## **Atomicity:**

- 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 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.



# *Transaction*

## **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.



# *Transaction*

## **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.





# *Transaction*

## **Durability:**

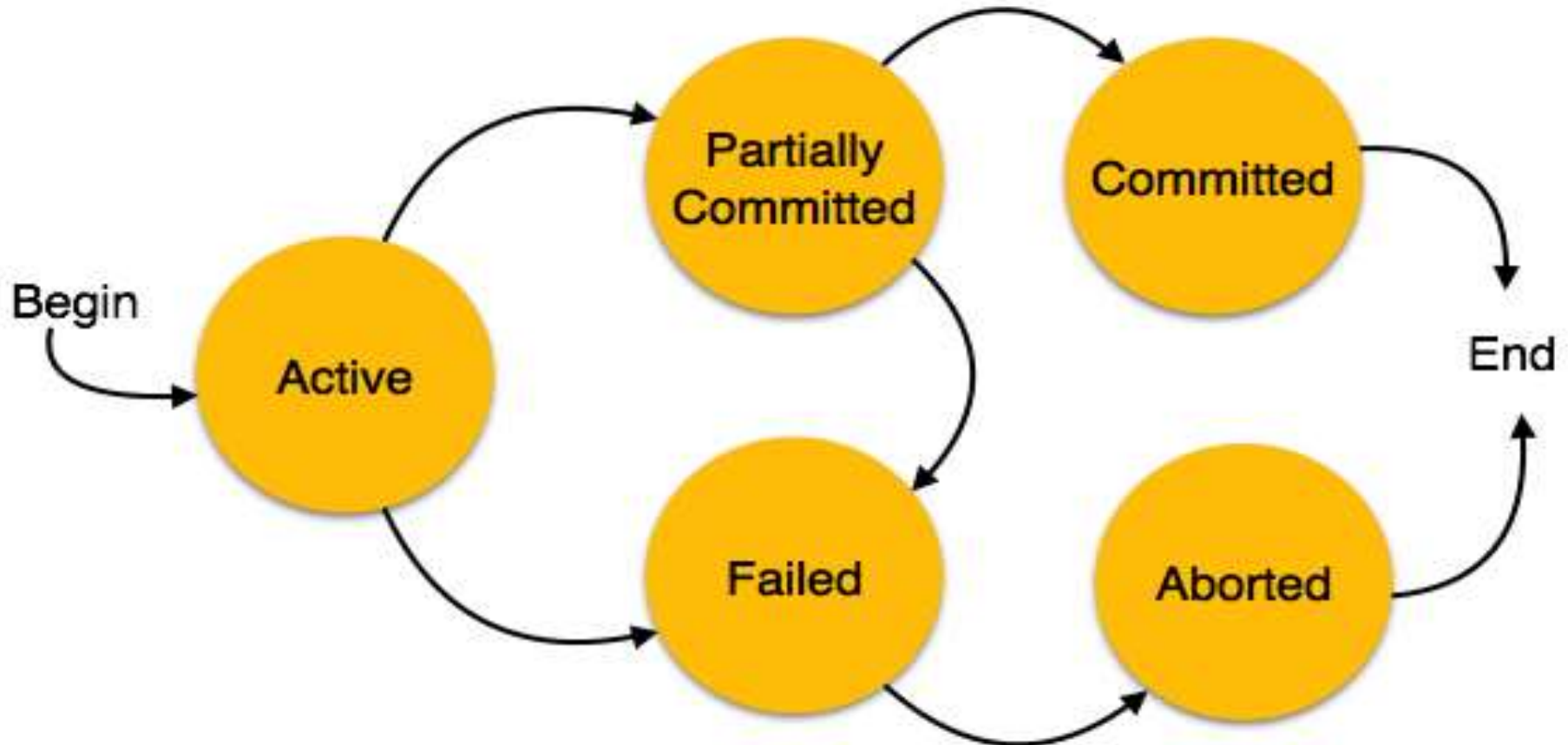
- The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure.
- The recovery subsystem of the DBMS has the responsibility of Durability property.



# *Transaction*

## States of Transaction:

In a database, the transaction can be in one of the following states.



# *Transaction*

## **States of Transaction:**

### ➤ **Active state**

The active state is the first state of every transaction. In this state, the transaction is being executed.

For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

### ➤ **Partially committed**

In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.

In the total mark calculation example, a final display of the total marks step is executed in this state.

### ➤ **Committed**

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.



# *Transaction*

## States of Transaction:

### ➤ Failed state:

- If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
- In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

### ➤ Aborted:

- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two operations:
  - Re-start the transaction
  - Kill the transaction



# *Concurrency Control*

## **Concurrency Control:**

- In the concurrency control, the multiple transactions can be executed simultaneously.
- It may affect the transaction result. It is highly important to maintain the order of execution of those transactions.

## **Problems of Concurrency Control:**

Several problems can occur when concurrent transactions are executed in an uncontrolled manner.

Following are the three problems in concurrency control.

1. **Lost updates**
2. **Dirty read**
3. **Unrepeatable read**



# *Concurrency Control*

## **The Need for Concurrency Control:**

A key purpose in developing a database is to facilitate multiple users to access shared data in parallel (i.e., at the same time). Concurrent accessing of data is comparatively easy when all users are only reading data, as there is no means that they can interfere with one another. However, when multiple users are accessing the database at the same time, and at least one is updating data, there may be the case of interference which can result in data inconsistencies.



# Distributed Concurrency Control

## Distributed Concurrency Control:

Concurrency control deals with the isolation and consistency properties of transactions. The distributed concurrency control mechanism of a distributed DBMS ensures that the consistency of the database.

The concurrency control protocol can be divided into mainly two categories:

- **Lock based Protocol**
- **Time-stamp Protocol**



*THANK YOU*

ANY QUERY???

