Transaction Processing Concepts

Transaction Processing

- A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Transaction Processing

- Let’s take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

Transaction Processing

A's Account

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

B's Account

```
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

Transaction Processing

- **The concept of *transaction*** :
  - provides a mechanism for describing logical units of database processing.

- **Transaction processing systems** :
  - systems with large databases and hundreds of concurrent users that are executing database transactions**.**

Singile user Versus Multiuser Systems

- **Single-User** : at most one user at a time can use the system
- **Multiuser** : many users can use the system concurrently.
- **Multiprogramming** : allows the computer to execute multiple programs at the same time.
- **Interleaving** : keeps the CPU busy when a process requires an input or output operation, the CPU switched to execute another process rather than remaining idle during I/O time .
- Most of the theory concerning concurrency control in databases is developed in terms of interleaved concurrency.

Transactions, Read and Write Operations, and DBMS Buffers

- **A Transaction** : is a logical unit of database processing that includes one or more database access operation.
- All database access operations between **Begin Transaction** and **End Transaction** statements are considered one logical transaction.
- If the database operations in a transaction do not update the database but only retrieve data , the transaction is called a **read-only transaction.**
- **Basic database access operations** :
  - **read_item(X)** : reads a database item **X** into program variable.
  - **Write_item(X)** : Writes the value of program variable **X** into the database item **X**.

Why Recovery Is Needed

- **There several possible reasons for a transaction to fail**
  - ***A computer failure*** : A hardware, software, or network error occurs in the computer system during transaction execution.
  - ***A transaction or system error*** : Some operations in the transaction may cause it to fail.
  - **Local errors or exception conditions** detected by the transaction.

Why Recovery Is Needed (continued)

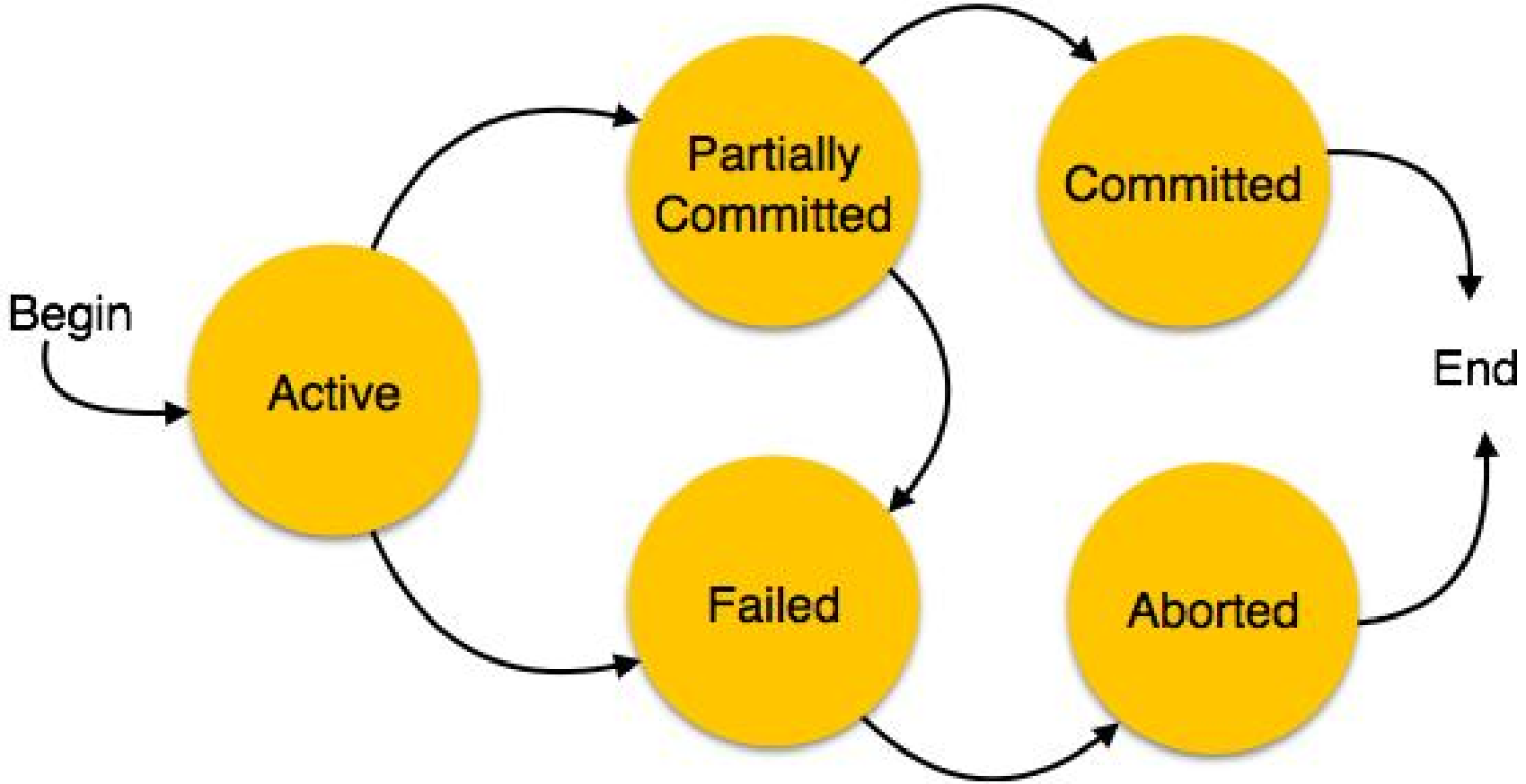
- ***Concurrency control enforcement*** : The concurrency control method may decide to abort the transaction.
- ***Disk failure*** : all disk or some disk blocks may lose their data
- ***Physical problems*** : Disasters,theft, fire, etc.

- **The system must keep sufficient information to recover from the failure.**

Transaction states and additional operations

- A transaction is an atomic unit of work that is either completed in its entirety or not done at all.
- For recovery purposes the system needs to keep track of when the transaction starts, terminates, and commits or aborts.
- The recovery manager keeps track of the following operations :
  - BEGIN_TRANSACTION
  - READ OR WRITE
  - END_TRANSACTION
  - COMMIT_TRANSACTION
  - ROLLBACK

Transaction States



Transaction States

- **Active** – 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, 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.

Transaction States

- **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

Transaction States

- **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.

Transaction states and additional operations (continued)

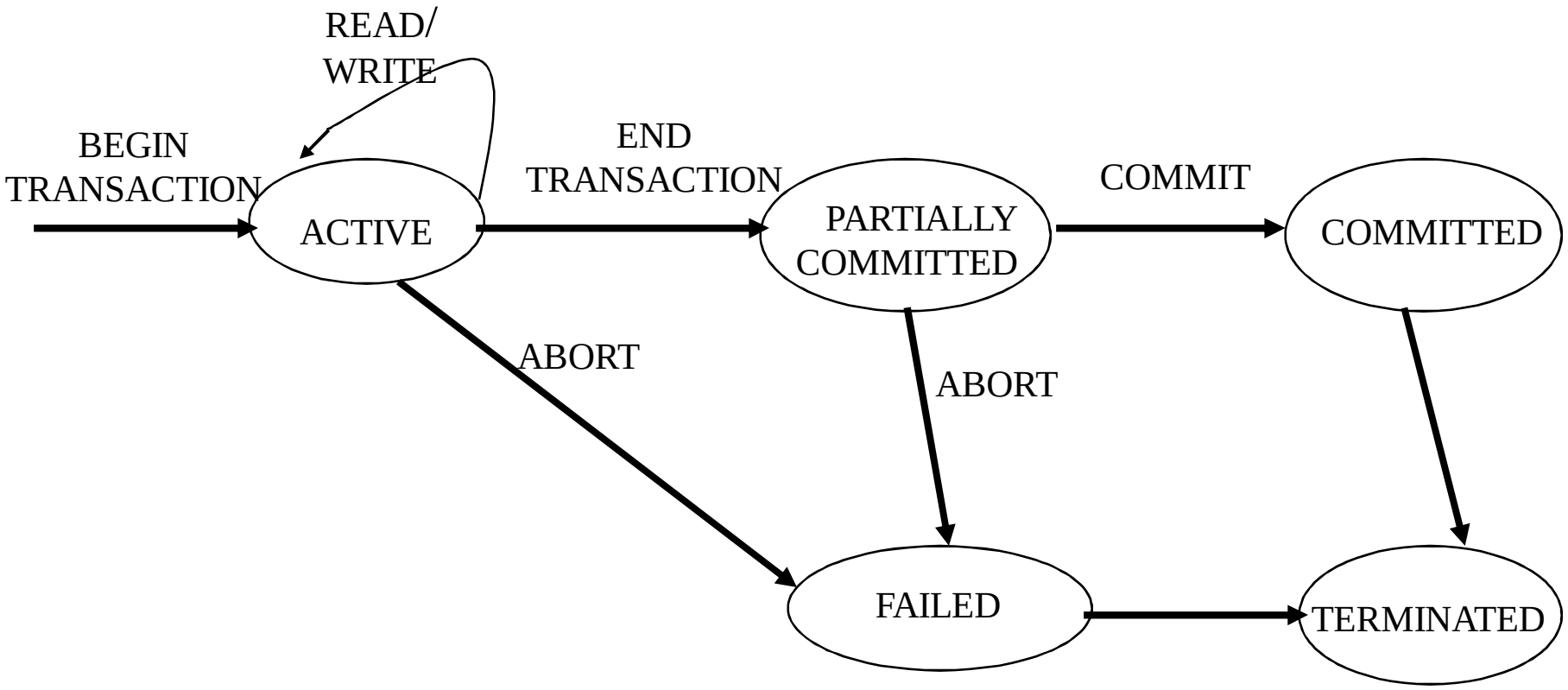


Figure 19.4 State transition diagram illustrating the states for transaction execution

The System Log

- The system maintains a log to keep track of all transaction operations that affect the values of database items.
- This log may be needed to recover from failures.
- Types of log records :
  - **[start_transaction,T] :** indicates that transaction T has started execution.
  - **[write_item,T,X,old_value,new_value] :** indicates that transaction T has changed the value of database item X from old_value to new_value.

    (new_value may not be recorded)

The System Log (continued)

- **[read_item,T,X]:** indicates that transaction T has read the value of database item X.

  (read_item may not be recorded)

- **[commit,T]:** transaction T has recorded permanently .

- **[abort,T]:** indicates that transaction T has been aborted.

Desirable Properties of transactions

- ACID should be enforced by the concurrency control and recovery methods of the DBMS.
- ACID properties of transactions :
  - ***Atomicity*** : a transaction is an atomic unit of processing; it is either performed entirely or not performed at all.

  (It is the responsibility of recovery)

  - ***Consistency*** : transfer the database from one consistent state to another consistent state

  (It is the responsibility of the applications and DBMS to maintain the constraints)

Desirable Properties of transactions (continued)

- ***Isolation*** : the execution of the transaction should be isolated from other transactions (Locking)

(It is the responsibility of concurrency)

* Isolation level:

-Level 0 (no dirty read) **-Level 1 (no lost update)**

-Level 2 (no dirty+ no lost) **-Level 3 (level 2+repeatable reads)**

- ***Durability*** : committed transactions must persist in the database ,i.e. those changes must not be lost because of any failure.

(It is the responsibility of recovery)