UNIT 3:
TRANSACTION PROCESSING, CONCURRENCY CONTROL, AND RECOVERY

Introduction - 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.

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

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

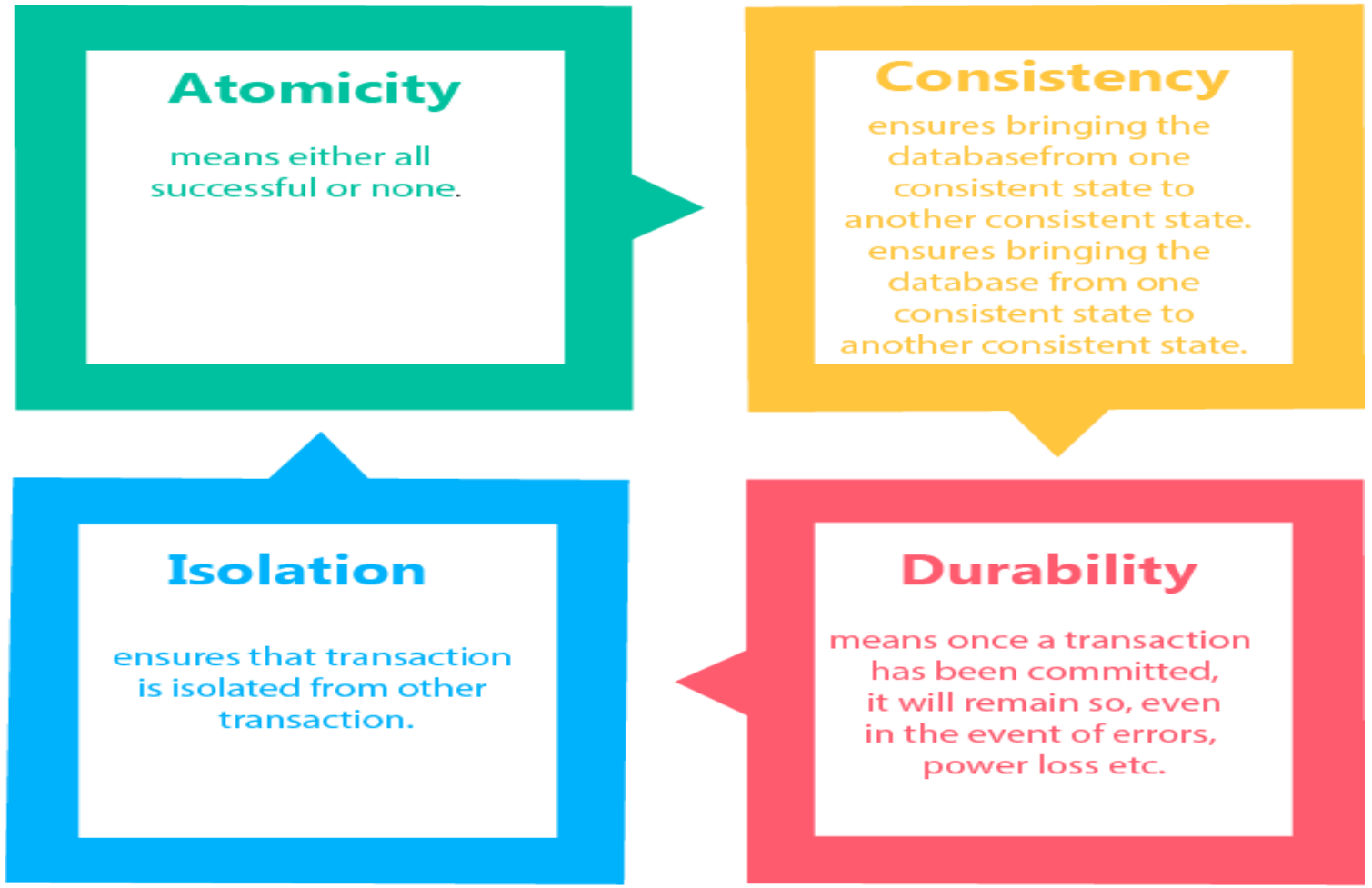
Operations of Transaction

- **Read(A):** Read operation is used to read the value of X from the database and stores it in a buffer in main memory.
- **Write(A):** Write operation is used to write the value back to the database from the buffer.

- The first operation reads A's value from database and stores it in a buffer.
- The second operation will decrease the value of A by 500. So buffer will contain 3500.
- The third operation will write the buffer's value to the database. So A's final value will be 3500.

```
R(A);
A = A - 500;
W(A);
```

Property of Transaction



Before: X : 500	Y : 200
Transaction T	
T1	T2
Read (X) X: = X - 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

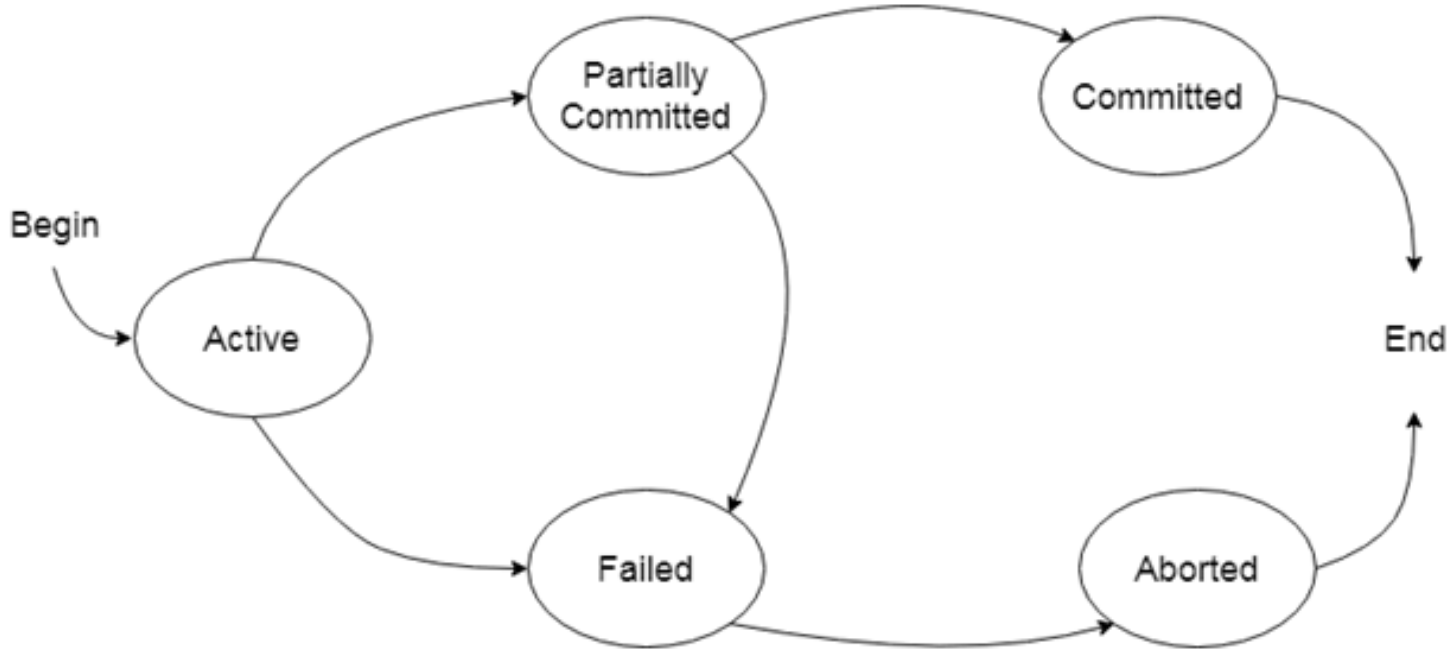
X = 500 Rs Y = 500 Rs

T	T''
Read (X) X :=X*100 Write (X) Read (Y) Y:= Y - 50 Write (Y)	Read (X) Read (Y) Z:= X + Y Write (Z)

X = 500 Rs Y = 500 Rs

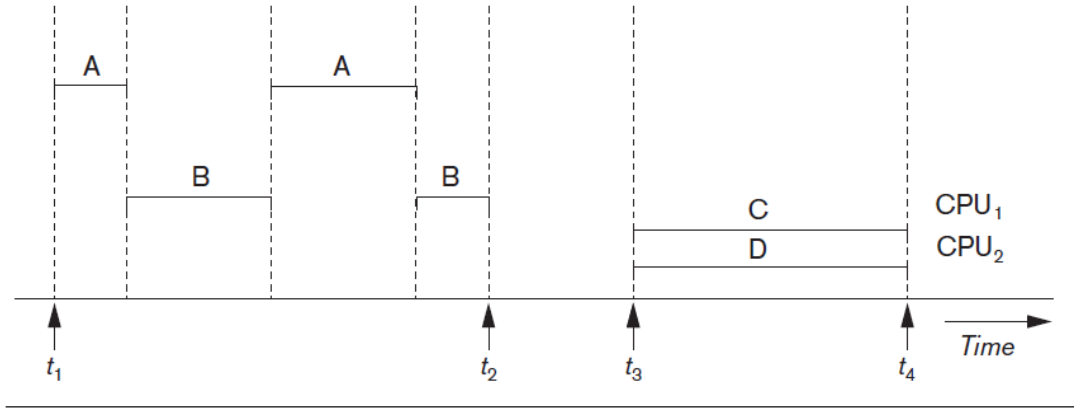
T	T''
Read (X) X :=X*100 Write (X) Read (Y) Y:= Y - 50 Write (Y)	Read (X) Read (Y) Z:= X + Y Write (Z)

States of Transaction



- **Single-User versus Multiuser Systems**

  Interleaved processing versus parallel processing of concurrent transactions.



Interleaved

- Interleaving also prevents a long process from delaying other processes.

- The database model that is used to present transaction processing concepts is quite simple when compared to the data models.
- A database is basically represented as a **collection of named data items**.
- The size of a data item is called its **granularity.**

Transaction Support in SQL

The characteristics are the

- access mode
- diagnostic area size
- isolation level

Access mode

- The **access mode** can be specified as **READ ONLY or READ WRITE**. The default is **READ WRITE**, unless the isolation level of READ UNCOMMITTED is specified, in which case READ ONLY is assumed.
- A mode of **READ WRITE** allows **select, update, insert, delete, and create commands to be executed.**
- A mode of **READ ONLY**, as the name implies, is simply for **data retrieval.**

Diagnostic area size

- The **diagnostic area size** option, DIAGNOSTIC SIZE n**, specifies an integer value n**, which indicates the **number of conditions** that can be held simultaneously in the diagnostic area.
- These conditions supply feedback information (errors or exceptions)to the user or program on the n most recently executed SQL statement.

Isolation level

- The **isolation level option** is specified using the statement **ISOLATION LEVEL <isolation>**, where the value for <isolation> can be **READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE**.
- The **default isolation level is SERIALIZABLE**, although some systems use READ COMMITTED as their default.
- The use of the term SERIALIZABLE here is based on not allowing violations that cause dirty read, unrepeatable read, and phantoms, and it is thus not identical to the way serializability was defined earlier.
- If a transaction executes at a lower isolation level than SERIALIZABLE, then one or more of the following three violations may occur:

1. **Dirty read:** A transaction T1 may read the update of a transaction T2, which has not yet committed. If T2 fails and is aborted, then T1 would have read a value that does not exist and is incorrect.

2. **Nonrepeatable read:** A transaction T1 may read a given value from a table. If another transaction T2 later updates that value and T1 reads that value again, T1 will see a different value.

3. Phantoms: A transaction T1 may read a set of rows from a table, perhaps based on some condition specified in the SQL WHERE-clause. Now suppose that a transaction T2 inserts a new row that also satisfies the WHERE-clause condition used in T1, into the table used by T1. If T1 is repeated, then T1 will see a phantom, a row that previously did not exist.

A sample SQL transaction

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
        READ WRITE
        DIAGNOSTIC SIZE 5
        ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno, Salary)
        VALUES ('Robert', 'Smith', '991004321', 2, 35000);
EXEC SQL UPDATE EMPLOYEE
        SET Salary = Salary * 1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ... ;
```

Table 21.1 Possible Violations Based on Isolation Levels as Defined in SQL

	Type of Violation		
Isolation Level	**Dirty Read**	**Nonrepeatable Read**	**Phantom**
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

Summary

- **READ UNCOMMITTED** is the **most forgiving**, and **SERIALIZABLE** is the most restrictive in that it **avoids all three of the problems** mentioned above.