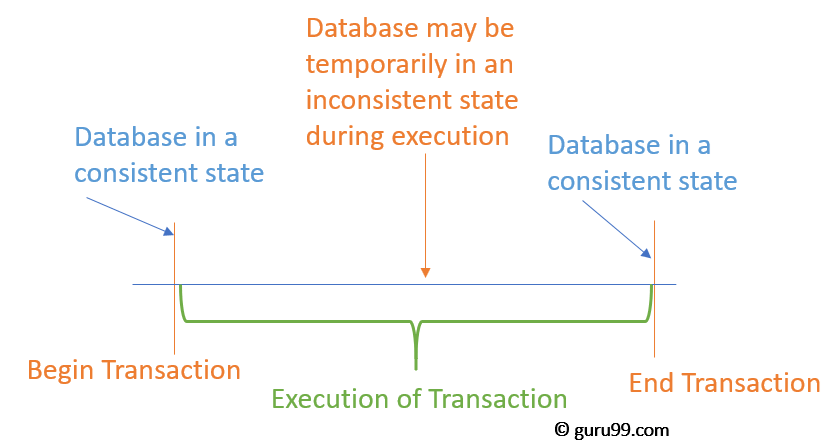
**What is a Database Transaction?**

A transaction is a logical unit of processing in a DBMS which entails one or more database access operation.

All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction.

During the transaction the database is inconsistent. 

## Facts about Database Transactions

* A transaction is a program unit whose execution may or may not change the contents of a database.
* The transaction is executed as a single unit
* If the database operations do not update the database but only retrieve data, this type of transaction is called a read-only transaction.
* A successful transaction can change the database from one CONSISTENT STATE to another
* DBMS transactions must be atomic, consistent, isolated and durable
* If the database were in an inconsistent state before a transaction, it would remain in the inconsistent state after the transaction.

## Concurrency in Transactions

A database is a shared resource accessed.

It is used by many users and processes concurrently. For example, the banking system, railway, and air reservations systems, stock market monitoring, supermarket inventory, etc.

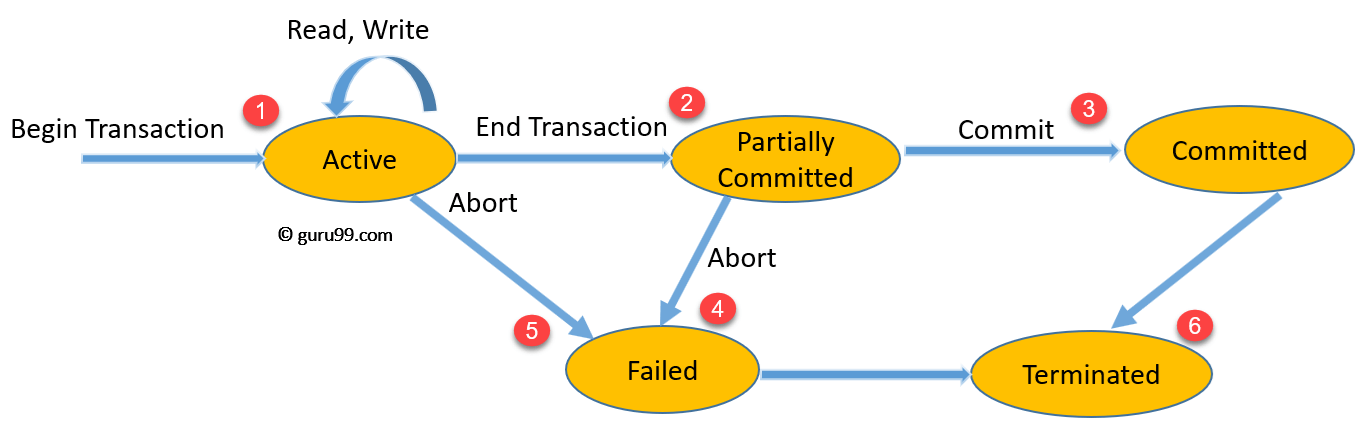
Not managing concurrent access may create issues like:

* Hardware failure and system crashes
* Concurrent execution of the same transaction, deadlock, or slow performance

**States of Transactions**

The various states of a Database Transaction are listed below

|  |  |
| --- | --- |
| **State** | **Transaction types** |
| Active State | A transaction enters into an active state when the execution process begins. During this state read or write operations can be performed. |
| Partially Committed | A transaction goes into the partially committed state after the end of a transaction. |
| Committed State | When the transaction is committed to state, it has already completed its execution successfully. Moreover, all of its changes are recorded to the database permanently. |
| Failed State | A transaction considers failed when any one of the checks fails or if the transaction is aborted while it is in the active state. |
| Terminated State | State of transaction reaches terminated state when certain transactions which are leaving the system can't be restarted. |

[](https://www.guru99.com/images/1/100518_0500_DBMSTransac6.png)State Transition Diagram for a Database Transaction

1. Once a transaction states execution, it becomes active. It can issue READ or WRITE operation.
2. Once the READ and WRITE operations complete, the transactions becomes partially committed state.
3. Next, some recovery protocols need to ensure that a system failure will not result in an inability to record changes in the transaction permanently. If this check is a success, the transaction commits and enters into the committed state.
4. If the check is a fail, the transaction goes to the Failed state.
5. If the transaction is aborted while it's in the active state, it goes to the failed state. The transaction should be rolled back to undo the effect of its write operations on the database.
6. The terminated state refers to the transaction leaving the system.

## ACID Properties

For maintaining the integrity of data, the DBMS system you have to ensure ACID properties. ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

* **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
* **Consistency:** Once the transaction is executed, it should move from one consistent state to another.
* **Isolation:**Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)
* **Durability:** **·**After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

**Serializability**

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

* **Schedule** − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
* **Serial Schedule** − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

## Types of Serializability

There are two types of Serializability.

### **Conflict Equivalence**

Two schedules would be conflicting if they have the following properties −

* Both belong to separate transactions.
* Both accesses the same data item.
* At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if −

* Both the schedules contain the same set of Transactions.
* The order of conflicting pairs of operation is maintained in both the schedules.

. **View equivalence** – If transactions in both schedules perform same actions in a similar manner then it is considered as view equivalence.

* Note − View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.
* Linearizability and serializability are both important properties about interleavings of operations in databases and distributed systems,
* Linearizability: single-operation, single-object, real-time order
* [Linearizability](http://cs.brown.edu/~mph/HerlihyW90/p463-herlihy.pdf)*is a guarantee about single operations on single objects.* It provides a real-time (i.e., wall-clock) guarantee on the behavior of a set of single operations (often reads and writes) on a single object. Linearizability can be viewed as a special case of strict serializability where transactions are restricted to consist of a single operation applied to a single object.
* Serializability: multi-operation, multi-object, arbitrary total order
* *Serializability is a guarantee about transactions, or groups of one or more operations over one or more objects.* It guarantees that the execution of a set of transactions (usually containing read and write operations) over multiple items is equivalent to *some* serial execution (total ordering) of the transactions.