Database Management System 19

Transactions

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

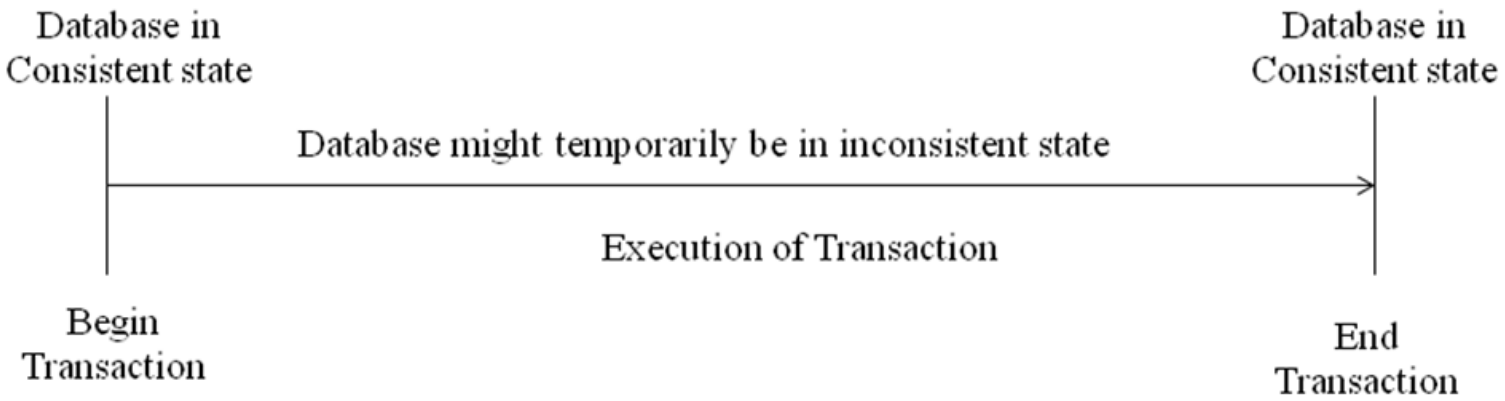
Transaction Concept

Transaction Concept

A transaction is a unit of program execution that accesses and possibly updates various data items

A transaction is a logical unit of work that contains one or more SQL statements. The effects of all the SQL statements in a transaction can be either all committed or all rolled back

A transaction that changes the contents of the database must alter the database from one consistent database state to another



ACID Properties

ACID Properties

- Atomicity: Either all operations of the transaction are reflected properly in the database, or none are. Atomicity requires that all operations of a transaction be completed; if not, the transaction is aborted by rolling back all the updates done during the transaction

- Consistency: Consistency means execution of a transaction should preserve the consistency of the database, i.e. a transaction must transform the database from one consistent state to another consistent state

- Isolation: Though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished execution before T_i started or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system

ACID Properties...

ACID Properties...

- Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures. Durability ensures that once transaction changes are done or committed, they can't be undone or lost, even in the event of a system failure

- The transactions access data item X using the following two operations:
  - ***Read(X)***: It transfers the data item X from the database to a local buffer belonging to the transaction that executed the read operation
  - ***Write(X)***: It transfers the data item X from the local buffer of the transaction that executed the write back to the database

ACID Properties...

Let T_1 be a transaction that transfers \$100 from account A to account B

T_1
Read(A);
A:=A-100;
Write(A);
Read(B);
B:=B+100;
Write(B);

1. Atomicity: The database system keeps track of the old values of any data on which a transaction performs a write, and if the transaction does not complete its execution, the database system restores the old values to make it appear as though the transaction never executed

- Ensuring atomicity is the responsibility of the database system itself. It is handled by the transaction management component

ACID Properties...

2. Consistency: Sum of A and B be unchanged by the execution of the transaction

- Ensuring the consistency for an individual transaction is the responsibility of the application programmers who codes the transaction

3. Isolation: The database is temporarily inconsistent while the transaction to transfer funds from account A to B is executing. The solutions are:

- Execute transactions serially
- However, concurrent execution of transactions provides significant performance benefits such as increased throughputs
- Ensuring the isolation property is the responsibility of concurrency control component of the database system

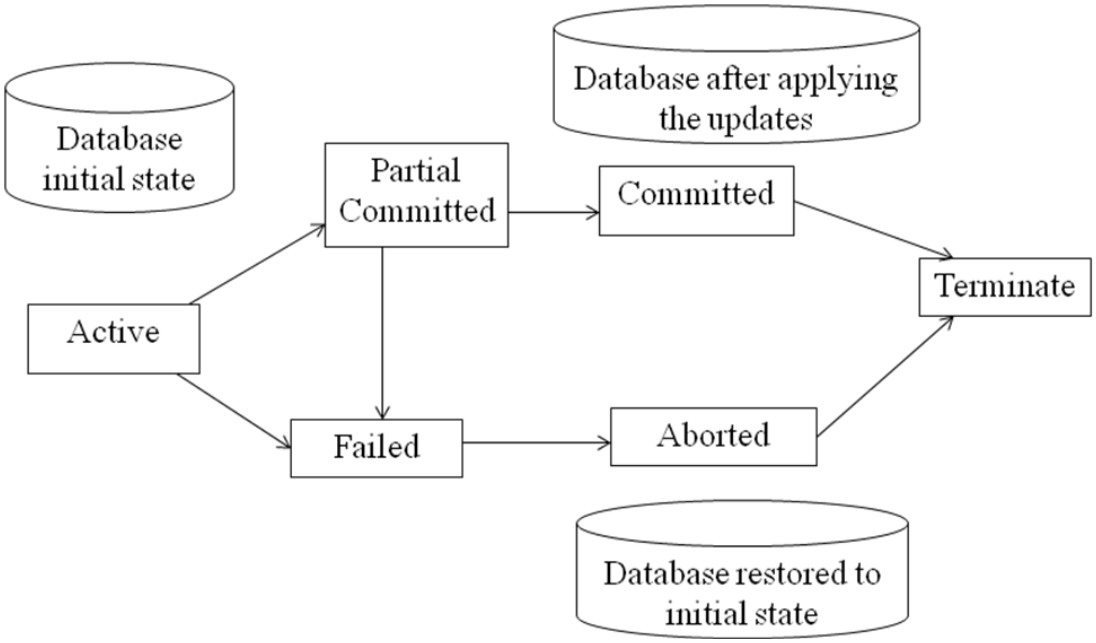
4. Durability: Once a transaction completes successfully, all the updates that is carried out on the database persist, even if there is a system failure after the transaction completes execution

- Ensuring durability is the responsibility of recovery management component of the database system

Transaction States

- **Active state**: This state is the initial state of a transaction. The transaction stays in this state while it is executing
- **Partial Committed state**: A transaction is partial committed after its final statement has been executed. A transaction enters this state immediately before the commit statement
- **Failed state**: A transaction enters the failed state after the discovery that normal execution can no longer proceed
- **Aborted state**: A transaction is aborted after it has been rolled back and the database is restored to its prior state before the transaction
- **Committed state**: Committed state occurs after successful completion of the transaction
- **Terminate**: Transaction is either committed or aborted

Transaction States...



When a transaction enters the aborted state, the system has two options:

- **Restart the transaction**: If the transaction was aborted as a result of a hardware failure or some software error (other than logical error), it can be restarted
- **Kill the transaction**: If the application program that initiated the transaction has some logical error