Transactions

Database Transaction

Database Transaction

A database transaction is a logical unit that generally includes several low-level steps. If one step of the transaction fails, the whole transaction fails. A database transaction is used to create, update, or retrieve data.

The transaction system ensures that the data in the database always remains in a reliable and consistent state.

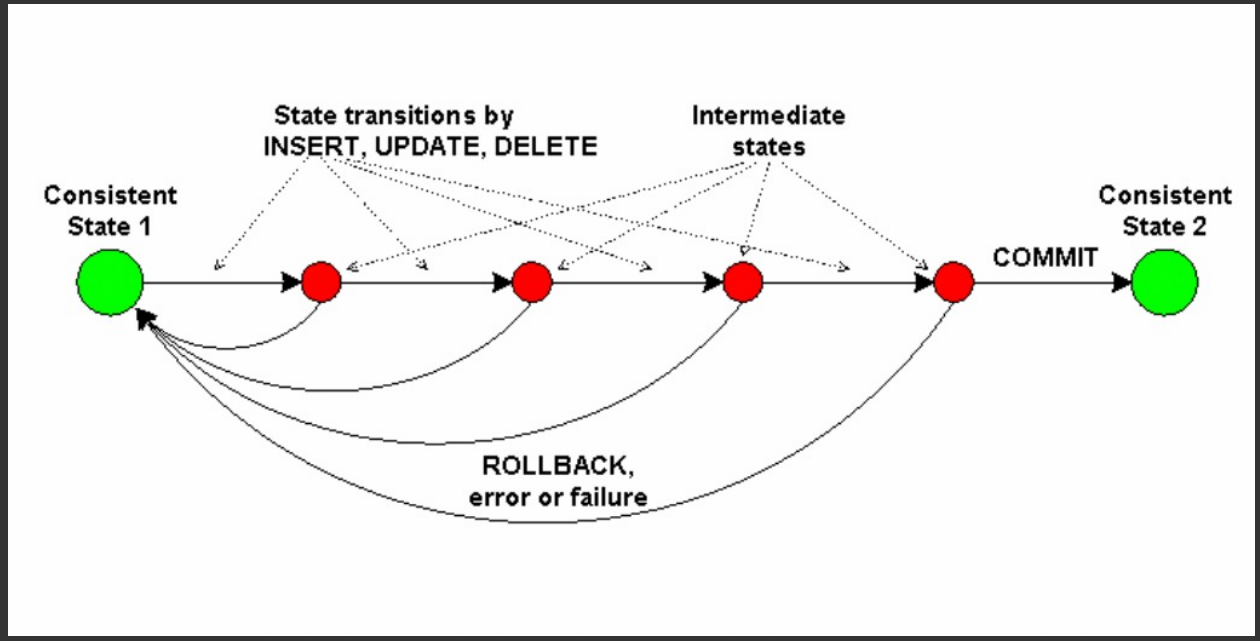
A's Account

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

Commit and Rollback

If a transaction is successful, the data in the database is updated as described in the instructions contained in the transaction. This is called a "commit."

If a transaction fails, all transaction steps performed prior to the step that led to the failure are reversed. The data in the database returns to its initial state as if the transaction had never been executed. This operation is called a "rollback."



ACID Properties

Atomicity: Ensures that a transaction is all-or-nothing; either all operations succeed, or none do.

For instance, in banking systems, transferring money between accounts should either debit and credit both accounts or not happen at all, ensuring no discrepancies.

Consistency: Guarantees that a transaction brings the database from one valid state to another, maintaining all rules and constraints.

In e-commerce platforms, applying business rules and constraints like stock availability and payment validity ensures users don’t over-purchase or face order errors.

ACID Properties

Isolation: Ensures that concurrent transactions do not interfere with each other, appearing as if they were executed sequentially.

In a reservation system (e.g., booking airline seats), isolation ensures that two people aren't able to book the same seat at the same time, maintaining order in concurrent operations.

Durability: Ensures that once a transaction is committed, its changes are permanent, even in the event of a system failure.

For instance, after a financial transaction is confirmed, it is stored permanently, ensuring it can be retrieved and verified later, avoiding financial discrepancies.

Advantages of ACID

- **Data Integrity**: ACID properties ensure data remains consistent and free from corruption.
- **Reliability**: ACID properties provide consistent and reliable execution of transactions.
- **Concurrency Control**: ACID properties enable simultaneous access to data without conflicts.
- **Fault Tolerance**: ACID properties ensure data durability, surviving system failures.
- **Transaction Management**: ACID properties offer structured transaction handling.

Needed in Banking systems, Reservation systems, e-commerce systems, etc.

Disadvantages of ACID

- **Performance Overhead**: ACID properties can impact system performance and throughput due to additional processing and resource utilization.
- **Complexity**: Implementing ACID properties adds complexity to database systems, increasing design and maintenance challenges.
- **Scalability Challenges**: ACID properties can pose difficulties in highly distributed or scalable systems, limiting scalability.
- **Potential for Deadlocks**: ACID transactions using locking mechanisms can lead to deadlocks and system halts.
- **Limited Concurrency**: ACID properties may restrict concurrency, impacting overall system throughput.

ACID properties can be ignored in Caching systems, analytics systems,

Large scale blogging systems, etc.