Database Transactions and ACID Properties in RDBMS

Introduction

A **database transaction** is a logical unit of work in a database that typically consists of multiple low-level operations, such as creating, updating, or retrieving data. It ensures that all the steps within the transaction are either fully completed or entirely undone (rolled back) if any step fails.

The transaction system guarantees that the database remains in a **reliable and consistent state**, even in cases of errors, system failures, or concurrent access. This reliability is enforced by adhering to **ACID properties**: Atomicity, Consistency, Isolation, and Durability.

To guarantee the integrity of transactions, RDBMS relies on the **ACID properties**:

- **Atomicity**: Ensures that a transaction is either fully completed or fully rolled back. No partial changes are allowed. 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**: Ensures that a transaction transforms the database from one valid state to another, adhering to all defined 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.
- **Isolation**: Ensures that transactions occur independently without interference, maintaining the effects of concurrent transactions as if 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**: Guarantees that once a transaction is committed, its changes are permanent, even in the event of a system crash. For instance, after a financial transaction is confirmed, it is stored permanently, ensuring it can be retrieved and verified later, avoiding financial discrepancies.

A **database transaction** ensures reliable and consistent operations in an RDBMS, like an **ATM withdrawal** process.
The **ACID properties** are:

1. **Atomicity**: Either the ATM gives you money and deducts it from your account, or neither happens (all or nothing).
2. **Consistency**: Your account balance is always correct before and after withdrawal (valid state to valid state).
3. **Isolation**: Your withdrawal doesn't interfere with someone transferring money to your account at the same time (no overlap).
4. **Durability**: Once the ATM gives you money, the transaction is permanently recorded, even during a power failure (results are saved).

Advantages of ACID

1. **Data Integrity**:
    - Ensures data consistency and prevents corruption during transactions.
2. **Reliability**:
    - Provides dependable and predictable execution of transactions, ensuring accurate outcomes.
3. **Concurrency Control**:

- Enables multiple users to access and modify data simultaneously without conflicts or interference.

4. **Fault Tolerance**:
    - Guarantees data durability, ensuring transactions are not lost during system crashes or failures.
5. **Transaction Management**:
    - Offers structured handling of transactions, making database operations efficient and error-free.

Use Cases:

- **Banking Systems**: For secure transfers and account updates.
- **Reservation Systems**: To prevent double-booking or overbooking.
- **E-commerce Systems**: For reliable order placement and payment processing.

Disadvantages of ACID

1. **Performance Overhead**:
    - Ensuring ACID compliance involves additional processing and resource usage, which can reduce system performance and throughput.
2. **Complexity**:
    - Implementing ACID properties increases the complexity of database systems, making their design, development, and maintenance more challenging.
3. **Scalability Challenges**:

- ACID properties can limit scalability, especially in highly distributed systems, as maintaining consistency across multiple nodes is resource-intensive.

4. **Potential for Deadlocks**:
    - The use of locking mechanisms in ACID transactions can result in deadlocks, halting operations, and requiring manual intervention.
5. **Limited Concurrency**:
    - ACID compliance may restrict simultaneous access to data, affecting system throughput and user experience.

When to Ignore ACID Properties:

ACID properties are not essential for systems prioritizing speed over strict consistency, such as:

- **Caching Systems**: Focused on fast data retrieval rather than durability.
- **Analytics Systems**: Where eventual consistency is acceptable.
- **Large-Scale Blogging Systems**: Prioritize high availability over strict transaction guarantees.