ACID Properties in DBMS: A Comprehensive Guide

By Simplilearn

Last updated on Jul 15, 2024 2390

Share This Article:

ACID features are used by Database Management Systems (DBMS) to maintain data integrity and consistency. Atomicity, Consistency, Isolation, and Durability (ACID) are basic concepts ensuring a database's transactional integrity. For developers working with relational database systems, understanding ACID characteristics is critical.

What are ACID Properties in DBMS?

ACID properties refer to four key principles - Atomicity, Consistency, Isolation, and Durability. They act as checks and balances for ACID properties database transactions to ensure accuracy and reliability.

- **Atomicity:** This principle states that database transactions must be all or nothing. If the transaction fails, the entire transaction is rolled back. Atomicity prevents partial and incomplete transactions.

- **Consistency:** According to this property, only valid data is written to the database. Consistency enforces integrity constraints to maintain the accuracy and correctness of data.
- **Isolation:** Running transactions independently is the core of isolation. Changes in one transaction will not impact others until committed. Isolation maintains data integrity across concurrent transactions.
- **Durability:** Durability guarantees that all committed transactions are permanently recorded in the database. They persist even after system failure. Durability provides recoverability.

Adhering to these four properties guarantees database transactions' reliability, accuracy, and consistency. The ACID principles are core mechanisms to manage the complexities of transactions, failures, and concurrency in database systems.

A – Atomicity

Atomicity requires transactions to be treated as a single "unit of work." The entire sequence of transaction operations succeeds or fails as one entity. There is no partial success or failure.

For example, transferring money from one bank account involves multiple steps:

- Debit amount X from Account A
- Credit amount X to Account B

As per atomicity, either all debit and credit operations succeed or they all fail. If the debit succeeds, but the credit fails for any reason, the entire transaction is rolled back. Atomicity ensures there are no partial or incomplete transactions.

Atomicity prevents unwanted data updates from unconcluded transactions. Without it, the debit may persist while the credit fails, causing data inconsistency. By reverting partial transactions, atomicity keeps the database consistent.

C – Consistency

The consistency property ensures that only valid data is written to the database. Before comr g a transaction, consistency checks are performed to maintain database constraints and business rules.

For example, a transaction crediting 5000 to a bank account with a current balance of 3000 is invalid if the account has an overdraft limit of 1000. The transaction violates consistency by exceeding the permissible account limit. Hence, it is blocked and aborted.

Consistency enforcement prevents data corruption and invalid entries. Only transactions abiding by consistency rules are committed to upholding data accuracy. Real-world business constraints are modeled into the database via consistency.

I – Isolation

Isolation maintains the independence of database transactions. Uncommitted transactions are isolated with locking mechanisms to prevent dirty reads or lost updates.

For instance, if Transaction T1 updates a row, Transaction T2 must wait until T1 commits or rolls back. Isolation prevents T2 from reading unreliable data updated by T1 but not committed yet.

Isolation avoids concurrency issues like:

- Dirty reads - Reading uncommitted data from other transactions
- Lost updates - Overwriting another transaction's uncommitted updates
- Non-repeatable reads - Same query yielding different results across transactions

By isolating transactions, consistency is maintained despite concurrent execution and updates. Changes remain isolated until permanent.

D – Durability

Durability provides persistence guarantees for committed transactions. The system upholds the changes once a transaction is committed even if it crashes later. Durability is achieved with database backups, transaction logs, and disk storage.

For example, if a transaction updates a customer's address, durability ensures the updated address is not lost due to a hard disk failure or power outage. The change will persist with the help of storage devices, backups, and logs.