ACID is a set of four properties that guarantee database transactions are processed reliably.[1] In the context of a database, a **transaction** is a single logical unit of work that may contain multiple steps (like a bank transfer).[2]

Without ACID compliance, a system failure or a simple power outage could leave your data in a "half-finished" or corrupted state.[3]

The Four Pillars of ACID

1. Atomicity ("All or Nothing")

Atomicity ensures that a transaction is treated as a single, indivisible unit.[4] If any part of the transaction fails, the **entire** transaction is aborted, and the database is rolled back to its state before the transaction started.[5]

- **Example:** In a money transfer, if the "Subtract $100" step succeeds but the "Add $100" step fails due to a crash, Atomicity ensures the first step is reversed so the money isn't lost in limbo.

2. Consistency[6]

Consistency ensures that a transaction only moves the database from one valid state to another, maintaining all predefined rules (constraints, cascades, triggers).[7]

- **Example:** If a database rule states that an account balance cannot be negative, a transaction that would result in a negative balance will be rejected to keep the data "consistent" with the rules.[8]

3. Isolation[9]

Isolation ensures that concurrent transactions (multiple people using the database at once) do not interfere with each other.[10] The result of running multiple transactions simultaneously should be the same as if they were run one after another.[11]

- **Key Levels:** Databases offer different "Isolation Levels" (like *Read Committed* or *Repeatable Read*) to balance strictness with performance.[12]

4. Durability

Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (like a power outage or a crash).[13]

- **Mechanism:** Most databases achieve this by using a **Write-Ahead Log (WAL)**, where changes are recorded in a permanent log file before they are applied to the actual data files.[14]

ACID in MySQL (InnoDB)

In MySQL, the **InnoDB** storage engine is the primary driver of ACID compliance.[15] It uses several internal mechanisms to handle these requirements:

- **Redo Logs:** Ensure **Durability** by recording every change so it can be replayed after a crash.
- **Undo Logs:** Support **Atomicity** by allowing the database to "undo" changes if a transaction is rolled back.[16]

- **Doublewrite Buffer:** Prevents data corruption during partial page writes (improving Durability).[17]

- **Snapshot Reads:** Provides **Isolation** by allowing users to see a "version" of the data at a specific point in time without locking the whole table.

ACID vs. BASE

While SQL databases usually prioritize ACID, many NoSQL databases use **BASE** (Basically Available, Soft state, Eventual consistency), which sacrifices immediate consistency for higher speed and availability in massive distributed systems.

Property	ACID (Traditional SQL)	BASE (Many NoSQL)
Consistency	Immediate & Strict	Eventual
Focus	Data Integrity	High Availability/Scale
Best For	Finance, ERP, Medical	Social Media, Real-time Analytics

ACID Properties in Databases Explained

This video provides a clear breakdown of each ACID property with real-world analogies to help visualize how they protect your data.

Would you like to see how to implement a safe ACID transaction in MySQL using START TRANSACTION, COMMIT, and ROLLBACK?