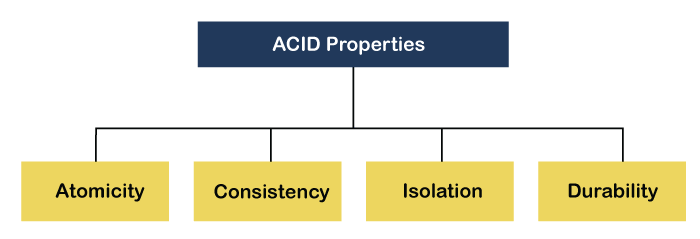
# **Acid Properties**



ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties collectively ensure the reliability and integrity of transactions, even in the face of system failures or errors.

The expansion of the term ACID defines for:



# **Atomicity**

**Atomicity** in MySQL, as part of the ACID properties (Atomicity, Consistency, Isolation, Durability), ensures that a series of database operations within a transaction either all succeed or all fail. This property guarantees that if any part of a transaction fails, the entire transaction is rolled back, leaving the database state unchanged as if the transaction was never executed.

# Example

Suppose we have two accounts in a banking system:

* Account A with a balance of 1000 Rupees.
* Account B with a balance of 500 Rupees.

We want to transfer 200 Rupees from Account A to Account B. The SQL operations to achieve this would include:

1. Subtracting 200 Rupees from Account A.
2. Adding 200 Rupees to Account B.

# Consistency

Consistency in the context of ACID means consistency in data, which is

defined by the creator of the database. The technical term for consistency in data is

called referential integrity. Referential integrity is a method of ensuring that

relationship between tables remain consistent.

# Example:

  Let us consider an example where one person is trying to book a ticket. They are able to

reserve their seat but their payment hasn’t gone through due to bank issues. In this case, their

transaction is rolled back. But just doing that isn’t sufficient. The number of available seats must also

be updated.

Otherwise, if it isn’t updated, there will be an inconsistency where the seat given up by the person is

not accounted for. Hence, the total sum of seats left in the train + the sum of seats booked by users

would not be equal to the total number of seats present in the train if not for consistency.

# **Isolation**

**Isolation** in MySQL, as part of the ACID properties (Atomicity, Consistency, Isolation, Durability), ensures that transactions operate independently and do not interfere with each other. Isolation levels control how and when the changes made by one transaction become visible to other concurrent transactions.

# Example:

  Let us consider an example where one person is trying to book a ticket. They are able to

reserve their seat but their payment hasn’t gone through due to bank issues. In this case, their

transaction is rolled back. But just doing that isn’t sufficient. The number of available seats must also

be updated.

Otherwise, if it isn’t updated, there will be an inconsistency where the seat given up by the person is

not accounted for. Hence, the total sum of seats left in the train + the sum of seats booked by users

would not be equal to the total number of seats present in the train if not for consistency.

# Durability

Durability simply means that once a transaction commits its changes,

those changes become part of the database’s permanent record, even in the event

of a power outage or other system failures. Database systems usually achieve

durability by moving in-memory data to non-volatile storage.

# Example :

Suppose that there is a system failure in the railway management system resulted in the loss

of all booked train details. Millions of users who had paid for their seats are now unable to board the

train, causing significant financial losses and eroding trust in the company. The situation is

particularly critical as these trains are needed for important reasons, causing widespread panic and

inconvenience.

# Use Cases

#### **1. E-Commerce Platform**

* **Atomicity:** Ensures that product inventory and order placement are updated together.
* **Consistency:** Maintains data integrity with constraints, such as not allowing negative stock levels.
* **Isolation:** Prevents issues such as overselling in high-concurrency environments.
* **Durability:** Guarantees that completed transactions (orders) are not lost, ensuring customer trust.

#### **2. Financial Systems**

* **Atomicity:** Transfers are fully completed or not executed at all.
* **Consistency:** Balances remain correct and consistent across transactions.
* **Isolation:** Avoids interference between concurrent transactions, preventing errors such as duplicate withdrawals.
* **Durability:** Transactions are permanently recorded, preserving account histories accurately.

#### **3. Reservation Systems**

* **Atomicity:** Booking transactions are either fully successful or rolled back.
* **Consistency:** Constraints (e.g., seat availability) are always respected.
* **Isolation:** Prevents concurrent bookings from resulting in overbooked seats.
* **Durability:** Ensures booking information is not lost and remains reliable.

### Conclusion

ACID properties are essential for maintaining the integrity and reliability of MySQL databases, especially in applications involving transactions like financial services, e-commerce, and reservation systems. By enforcing these properties, MySQL ensures data consistency, handles concurrency effectively, and provides durability against failures.