ACID Properties in SQL (MySql)

In the world of databases, ensuring data accuracy, consistency, and reliability is essential. MySQL, as a relational database management system, follows the ACID properties to maintain these requirements. ACID stands for **Atomicity, Consistency, Isolation, and Durability**. Let's break down each property in simple terms.

1. Atomicity

Definition: Atomicity ensures that a transaction is treated as a single, indivisible unit. Either all operations in the transaction succeed, or none of them do.

Why it matters: Imagine transferring money between two bank accounts. If only one part of the transaction completes (like deducting money from the sender but not adding it to the recipient), it would cause serious issues.

Example:

1) Transferring funds between bank accounts.

```
START TRANSACTION;
UPDATE bank_accounts SET balance = balance - 1000
WHERE account_number = '1234567890';

UPDATE bank_accounts SET balance = balance + 1000
WHERE account_number = '9876543210';
COMMIT;
```

Key Commands:

- **START TRANSACTION:** Begins a transaction.
- **COMMIT**: Saves the transaction changes.
- **ROLLBACK**: Cancels the transaction and restores the original state.

2. Consistency

Definition: Consistency ensures that the database remains in a valid state before and after a transaction.

Why it matters: Data integrity rules (like primary keys, foreign keys, and constraints) must be maintained during all operations.

Example:

1) Inserting a new order that requires a valid product ID and a non-negative quantity.

```
-- Adding an order with an invalid product_id breaks data integrity

INSERT INTO orders (order_id, product_id, quantity)

VALUES (101, 'INVALID_PRODUCT', -5);

-- This would trigger a constraint violation!
```

This example shows how constraints ensure valid relationships and prevent erroneous data entry.

3. Isolation

Definition: Isolation ensures that concurrent transactions do not interfere with each other. Each transaction is executed as if it were the only one running.

Why it matters: Imagine two users trying to book the last seat on a flight at the same time. Isolation ensures that only one transaction will succeed, and the other will see the updated seat availability.

Isolation Levels in MySQL:

1. **Read Uncommitted:** Transactions can read uncommitted changes (can lead to dirty reads).
2. **Read Committed:** Transactions only read committed changes.
3. **Repeatable Read (Default in MySQL):** Ensures consistent data for the duration of the transaction.
4. **Serializable:** The highest isolation level; transactions are fully isolated.

Example:

1) Two users simultaneously reserving the same hotel room.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION;
SELECT * FROM room_reservations
WHERE room_id = 202 AND status = 'available';
 -- User A confirms reservation while User B is blocked until User A's
transaction completes
UPDATE room_reservations SET status = 'booked'
WHERE room_id = 202;
COMMIT;
```

This ensures that no two users can book the same room at the same time.

4. Durability

Definition: Durability ensures that once a transaction is committed, the changes are permanently saved, even in the event of a system failure.

Why it matters: Imagine completing an e-commerce order. You expect the order details to be saved permanently, even if the system crashes afterward.

Example:

1) Recording an online purchase.

```
START TRANSACTION;

INSERT INTO orders (order_id, customer_id, order_date, total_amount)

VALUES (1001, 501, NOW(), 299.99);

COMMIT;
```

After the COMMIT command, MySQL ensures that the changes are stored safely on disk.

How MySQL Ensures Durability:

- Write-Ahead Logging (WAL)
- InnoDB storage engine with crash recovery mechanisms

Summary Table:

Property	Description	Example Scenario
Atomicity	All-or-nothing execution of a transaction	Bank money transfer
Consistency	Database remains in a valid state	Ensuring unique user IDs
Isolation	Concurrent transactions do not interfere	Booking the last flight seat concurrently
Durability	Changes are permanently saved after commit	Saving purchase details even after a system crash