01

Transaction Management & ACID Properties in MySQL

Ensure data consistency and reliability in your database operations

02

What Is a Transaction?

A transaction is a sequence of operations performed as a single logical unit. Transactions in databases ensure that all changes are executed successfully, or none of them are.

Key Points:

- Atomic: All or nothing. Every query within the transaction succeeds, or the transaction fails as a whole.
- Committed: Once a transaction is committed, all changes are permanent.

03

Transaction Example

```
START TRANSACTION;
UPDATE accounts SET balance = balance - 500
WHERE account_id = 1;
UPDATE accounts SET balance = balance + 500
WHERE account_id = 2;
COMMIT;
```

This example transfers $500 from account_id = 1 to account_id = 2. Both steps occur within the transaction, ensuring that either both succeed or both fail, maintaining data integrity.

04

What Are ACID Properties?

ACID properties are essential to maintain data consistency in databases. They ensure transactions are reliable.
ACID Stands For:

- Atomicity
- Consistency
- Isolation
- Durability

05

Atomicity

Atomicity ensures that all operations within a transaction succeed or none at all. If any part fails, the whole transaction is rolled back.

Example:
Transferring money between accounts: both debit and credit must be executed or neither.

06

Consistency

Consistency ensures that a transaction takes the database from one valid state to another, following all predefined rules (constraints, triggers).

Example:
If you update stock levels in a warehouse, the total stock should remain correct after the transaction.

07

Isolation

Isolation ensures that the outcome of a transaction is not affected by other concurrent transactions. One transaction cannot see intermediate changes made by others.

Example:
If two users are updating the same record at the same time, their changes won’t interfere if isolation is maintained.

08

Durability

Durability ensures that once a transaction is committed, it will remain so, even if the system crashes or the database shuts down.

Example:
If a power outage occurs after committing a transaction, the changes are still saved in the database.

09

Managing Transactions

You can control transactions using START TRANSACTION, COMMIT, and ROLLBACK commands in MySQL.

```
START TRANSACTION;
-- Execute SQL queries
COMMIT; -- Save changes permanently
ROLLBACK; -- Cancel changes
```

10

ROLLBACK Example

If a transaction encounters an error or needs to be canceled, the ROLLBACK command undoes all changes made within the transaction.

```
START TRANSACTION;
UPDATE products SET stock = stock - 1 WHERE
product_id = 123;
-- Something goes wrong
ROLLBACK;
```

In this case, the stock update is undone, ensuring no incorrect data is saved.

When to Use Transactions?

Transactions are useful in:

- Multi-step operations: e.g., transferring money, updating stock, and placing an order.
- Data consistency: Ensure changes are complete and correct.
- Handling errors: Prevent incomplete data changes in case of an error.

“Now that you've learned about transaction management and ACID properties, start applying them in your database projects to ensure consistency and integrity!”