G H raisoni COLLEGE
Engineering and Management
Nagpur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

Subject :- Database Management System

Unit IV

Transaction Management

Subject Teacher

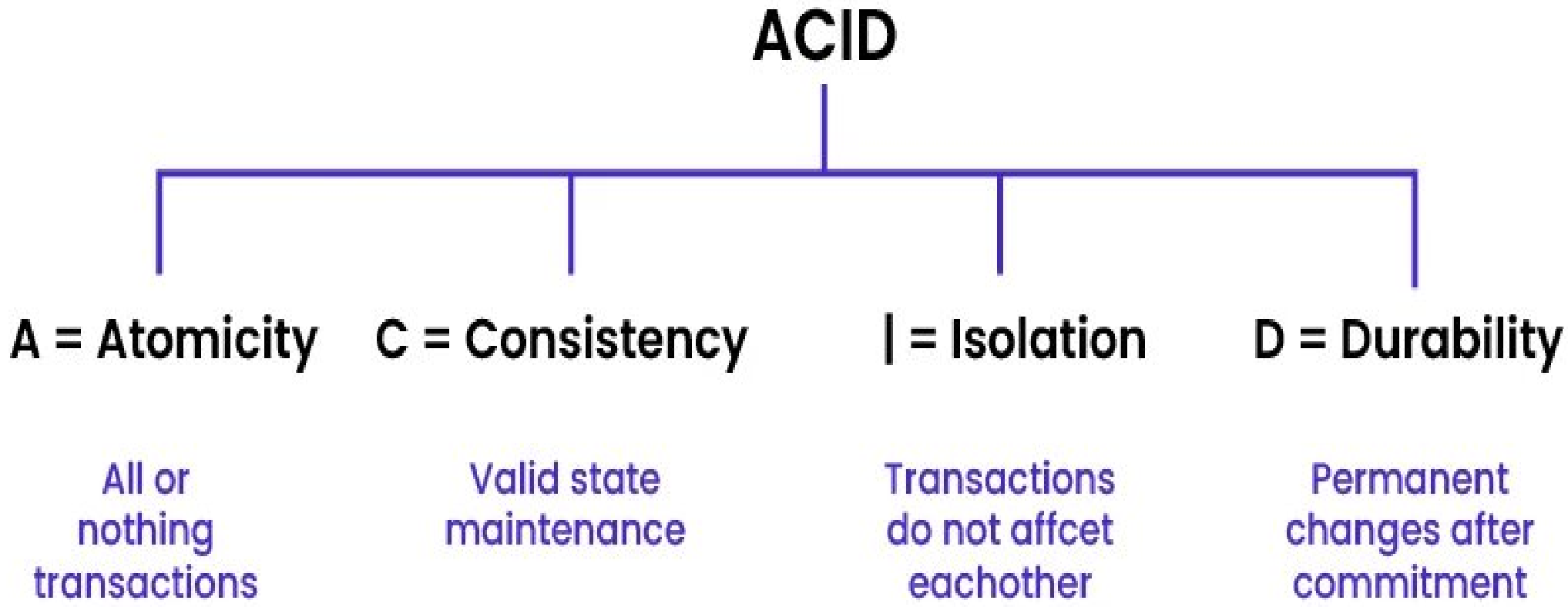
Prof.Vaishali Jangade

INTRODUCTION

- When using a Database Management System (DBMS), managing data accurately and securely is very important.
- Even small mistakes can cause serious problems such as incorrect balances, missing data, or loss of information.
- To handle this, the concept of a transaction is used.
- A transaction is a group of one or more operations, like reading, updating, or deleting data, that are treated as a single logical unit.
- The main goal of a transaction is to keep the database consistent and accurate even when unexpected issues like power failures or system crashes happen.
- If all the operations in a transaction complete successfully, the transaction is committed, which means the changes are saved permanently.
- If any operation fails, the transaction is rolled back to undo all changes and protect the database from errors.

For example:
If you transfer money from your bank account to your friend's bank account, the actions involved would include:
•Deducting money from your bank account.
•Crediting money to your friend's bank account.

ACID properties in DBMS



1. Atomicity

Atomicity refers to the property of a transaction where it behaves like a single atomic unit of any kind. Either all the operations occur or none of them occur.
Key point: No partial updates allowed.

Example: Let's say you are transferring money from your bank account to your friend's bank account.

- The bank would deduct Rs 500 from your bank account.
- The bank would add Rs 500 to your friend's bank account.

If the Rs 500 was deducted from your account, but the second step were to fail, the entire transaction would need to be undone. That is atomicity, either both steps succeed or neither does.

2. Consistency

Consistency ensures that all predefined rules, constraints, and data integrity conditions are maintained before and after a transaction in a database.
It guarantees that the database moves from one valid state to another.

Key point: A transaction must take the database from one consistent state to another.

Example: Consider a rule where the price of a car in the inventory cannot have a price less than Rs 1 Lakh.
If a transaction changes a car price to Rs 80,000, then the rule is broken and the database is now inconsistent.

3. Isolation

Isolation ensures that concurrent transactions do not interfere with one another.
Each transaction behaves as if it is executing alone. This prevents issues such as dirty reads and non-repeatable reads.

Key point: One transaction should not impact another transaction that is running at the same time.

Example: Two people are trying to buy the last car in stock at the same time.
•Transaction A checks the stock (1 car in stock).
•Transaction B also checks the stock (1 car in stock).

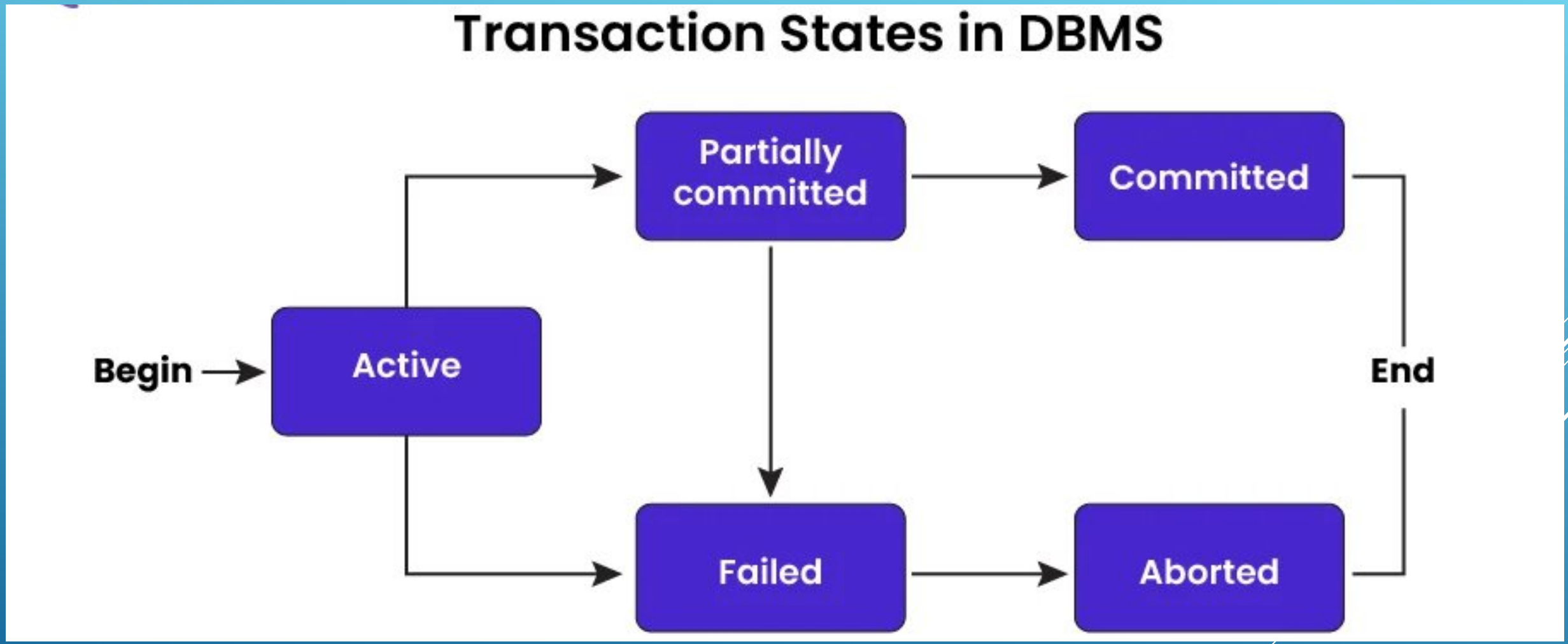
If both transactions are allowed to be processed, they will go beyond the stock. Isolation ensures that only one transaction occurs, while the other either waits or fails, thus avoiding concurrency conflicts.

4. Durability

Durability means once a transaction is successful and committed, its result is permanently stored in the system (even if the system crashes).

Key point: Once success occurs, changes are not lost.

Example: After a successful car sale, the database updates the stock and commits.
The car selling application stops unexpectedly.
When the system is restarted, after shutting down, stock updates should still be there.


Transaction States in DBMS
Partially committed
Committed
Begin
Active
End
Failed
Aborted

Practical Applications of Transactions in DBMS

1. Online Banking

An example of a transaction in an online banking service is when a user sends money from their checking account to the receiver's account.

Steps involved:

- Read the balance of the sender's account.
- Subtract the amount the user wishes to send from the sender's account.
- Add the amount the user has sent to the receiver's account.
- All changes are saved (Commit).

2. E-commerce Order Processing

When a customer places an order on an e-commerce site, multiple operations such as checking inventory, processing payment, and updating the order status must be executed as a single transaction to ensure data consistency.

Steps involved:

- Check if the product is in stock.
- Reduce the number of items in the inventory.
- Persist the order details.
- Process payment.
- Acknowledge the order with the customer.

Transaction Management in DBMS

Transaction management in DBMS involves a set of actions (a transaction) undertaken in order to keep the database accurate and secure. A transaction can be thought of as a single operation.

For example, if you transfer money from one bank account to another, there are two basic steps: you subtract money from one account, then you add it to another.

Benefits of Transaction Management in DBMS

Let's explore how transaction management helps keep database operations reliable, consistent, and secure.

1.Data Consistency: Database management systems (DBMS) guarantee that the database transitions between correct states, regardless of concurrent operations or concurrent users.

2.Error Recovery & Handling: A DBMS can recover the database state if a transaction fails, allowing the undo of any incomplete changes if the system crashes. This error handling allows the integrity of the database to remain intact.

3.Concurrency Control: A DBMS provides concurrency control, allowing multiple users to work simultaneously and maintain their transactions.

4.Simplified Application Logic: The DBMS allows the developer to "group" actions together for purposes of treating them as a unit of work, and for making the code easier to understand, maintain, and manage.

5.Secure and Reliable Operations: A DBMS protects the data from partial updates, unintended changes, or unauthorized users, so that you can ensure the information is correct and reliable.

Get 100% Hike!

SQL TCL statements

- Transaction Control Language (TCL) in SQL is a subset of SQL commands used to manage database transactions.
- Transactions are a fundamental concept in database systems, ensuring data integrity and consistency by grouping a set of SQL statements into a single logical unit of work.
- TCL commands allow developers to control when changes to the database are committed permanently or rolled back to a previous state.

- A transaction has two possible outcomes: success or failure.

- If all of the operations within a transaction are successful, the transaction is committed, and the changes are permanently stored in the database.

- However, if any of the operations within a transaction fail, the entire transaction is rolled back, and the database is restored to its state before the transaction began.
- TCL commands provide the mechanism for managing transactions, ensuring data integrity and consistency.

They enable users to control when changes are made persistent and provide a way to recover from errors or unexpected events.

1. COMMIT

The COMMIT statement is used to permanently save all changes made during the current transaction to the database.
Once a transaction is committed, the changes become permanent and cannot be rolled back.
Committing a transaction indicates that you are satisfied with the changes and want to make them a permanent part of the database.

```
COMMIT;
```

2. ROLLBACK

The ROLLBACK statement is used to undo all changes made during the current transaction,
effectively canceling the transaction.
It is often used when an error occurs during a transaction or when you decide not to apply the changes made.

```
ROLLBACK;
```

3. SAVEPOINT

The SAVEPOINT statement allows you to create a point within a transaction to which you can later roll back.
This is useful when you want to partially undo changes made within a transaction without canceling the entire transaction.

```
SAVEPOINT NAME;
```

THANK YOU