Database Management Systems (DBMS)

DBMS organizes, stores, and retrieves data efficiently.
RELATIONAL DATABASE CONCEPTS:
1. Tables (Relations)
   - Rows (Tuples): Individual records
   - Columns (Attributes): Data fields
   - Primary Key: Unique identifier
   - Foreign Key: Links tables
2. Normalization
   Purpose: Reduce redundancy, improve integrity
   - 1NF: Atomic values, no repeating groups
   - 2NF: 1NF + No partial dependencies
   - 3NF: 2NF + No transitive dependencies
   - BCNF: Stronger form of 3NF

SQL BASICS:
1. DDL (Data Definition Language)
```
CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
ALTER TABLE students ADD email VARCHAR(100);
DROP TABLE students;
```
2. DML (Data Manipulation Language)
```
INSERT INTO students VALUES (1, 'John', 20);
UPDATE students SET age = 21 WHERE id = 1;
DELETE FROM students WHERE id = 1;
SELECT * FROM students WHERE age > 18;
```
3. Joins
   - INNER JOIN: Matching records from both tables
   - LEFT JOIN: All from left + matching from right
   - RIGHT JOIN: All from right + matching from left
   - FULL OUTER JOIN: All records from both tables

ACID PROPERTIES:
1. Atomicity: All or nothing execution
2. Consistency: Database remains in valid state
3. Isolation: Concurrent transactions don't interfere
4. Durability: Committed changes persist

TRANSACTION MANAGEMENT:
Transaction States:
- Active: Initial state, executing
- Partially Committed: After final statement

- Committed: Successfully completed
- Failed: Cannot proceed further
- Aborted: Rolled back to previous state

INDEXING:

1. Primary Index: On primary key
2. Secondary Index: On non-key attributes
3. Clustered Index: Data stored in order
4. B-Tree Index: Balanced tree structure
5. Hash Index: Hash function based

Benefits: Faster query execution

Drawback: Slower inserts/updates

DATABASE DESIGN:

ER Model Components:

- Entity: Real-world object (Student, Course)
- Attribute: Properties (Name, Age)
- Relationship: Association between entities
- Cardinality: One-to-One, One-to-Many, Many-to-Many

NOSQL DATABASES:

Types:

1. Document (MongoDB): JSON-like documents
2. Key-Value (Redis): Simple key-value pairs
3. Column-Family (Cassandra): Wide column stores
4. Graph (Neo4j): Nodes and relationships

When to use: High scalability, flexible schema, big data