## Section 1: Basic Database Concepts

1. **What is a database?**
   A database is an organized collection of structured data that allows efficient storage, retrieval, and management of information. It is designed to store large amounts of data systematically, making it easier to access and manipulate using queries. Example: A customer database storing names, phone numbers, and purchase histories.

2. **What are the key features of a database?**
   - Data organization and storage
   - Data retrieval and querying
   - Data security and access control
   - Data consistency and integrity
   - Concurrency control and multi-user support
   - Backup and recovery mechanisms

3. **What are the different types of databases?**
   - Relational Databases (RDBMS): MySQL, PostgreSQL
   - NoSQL Databases: MongoDB, Redis
   - Object-Oriented Databases: db4o, ObjectDB
   - Hierarchical Databases: IBM Information Management System (IMS)
   - Graph Databases: Neo4j
   - Time-Series Databases: InfluxDB

4. **What is the difference between a file system and a database system?**
   - A file system stores data in separate files without relationships, while a database system organizes data in structured formats with relationships and indexing.
   - Databases provide ACID compliance, while file systems do not.
   - Databases support complex queries and data integrity features that file systems lack.
   - Why do we use databases instead of spreadsheets?
   - Databases handle large datasets efficiently, while spreadsheets are limited in scalability.
   - Databases provide better data integrity, security, and multi-user access.
   - Databases support complex queries and relationships, whereas spreadsheets lack relational capabilities.

5. **Why do we use databases instead of spreadsheets?**
   While spreadsheets (like Excel or Google Sheets) are useful for small-scale data management, they have limitations when dealing with large datasets, complex relationships, or multi-user environments. Databases, especially Relational Database Management Systems (RDBMS), offer several advantages over spreadsheets.

   **Limitations of Spreadsheets**

   1. **Scalability Issues** – Spreadsheets slow down with large amounts of data.

2. **Data Integrity Issues** – No strict rules to maintain data accuracy.
3. **Multi-User Collaboration Challenges** – Spreadsheets can cause conflicts when multiple users edit simultaneously.

**6. What is a DBMS (Database Management System)?**
A Database Management System (DBMS) is software that allows users to store, manage, and retrieve data efficiently. It provides an interface between users and databases, ensuring data organization, security, and consistency.

- **Key Functions of a DBMS**

**Data Storage & Organization** – Stores data in a structured manner (tables, key-value, documents, etc.).

**Data Retrieval & Querying** – Uses SQL (Structured Query Language) or other query languages.

**Data Security** – Controls user access, authentication, and encryption.

**Data Consistency & Integrity** – Ensures data accuracy using constraints.

Backup & Recovery – Protects data from failures or corruption.

**7. What are the key components of a DBMS?**

**Database Engine:** Core service for accessing and managing data

**Query Processor:** Interprets and executes queries

**Transaction Management:** Ensures data integrity and ACID properties

**Security Management:** Controls user access

**Backup and Recovery Management:** Protects data against loss

**8. What are the advantages of using a database?**
- Data consistency and integrity
- Efficient data retrieval and manipulation
- Scalability for large data volumes
- Security features for data protection
- Multi-user support with concurrency control

**9. What are Primary Key and Foreign Key in a database?**
- Primary Key: A unique identifier for each record in a table. Example: StudentID in a Students table.
- Foreign Key: A field that establishes a relationship between two tables. Example: CourseID in an Enrollment table linking to the Courses table.

**10. What do you mean by data integrity in databases?**
Data integrity ensures accuracy, consistency, and reliability of data throughout its lifecycle.
Example: Constraints like UNIQUE, NOT NULL, and FOREIGN KEY enforce integrity.

## Section 2: Relational Databases (RDBMS)

**11. What is an RDBMS (Relational Database Management System)?**
A Relational Database Management System (RDBMS) is a type of database management system that stores data in a structured format using tables (rows and columns) and establishes relationships between them. It is based on the relational model, which was introduced by E.F. Codd in 1970.

RDBMS enables users to store, retrieve, manipulate, and manage data efficiently while maintaining data integrity and consistency. It uses Structured Query Language (SQL) to perform database operations.

**12. How is data stored in an RDBMS?**
In a Relational Database Management System (RDBMS), data is stored in a structured and organized manner using tables. These tables consist of rows (records) and columns (fields), and they are connected through relationships. RDBMS follows the ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure reliability and data integrity.

1. Tables (Relation)
2. Rows (Records)
3. Columns (Fields/Attributes)
4. Primary Key (PK) and Foreign Key (FK)

**13. What are tables, rows, and columns in an RDBMS?**
- Table: A structured collection of data (Example: Employees table).
- Row: A single record (Example: Employee ID 101, Name: John Doe).
- Column: A data field (Example: Employee Name, Employee ID).

**14. What is ACID compliance in databases?**
- Atomicity: Ensures transactions are fully completed or not executed at all.
- Consistency: Maintains valid data states before and after transactions.
- Isolation: Ensures concurrent transactions do not affect each other.
- Durability: Ensures committed transactions remain even after failures.

**15. What is Normalization, and why is it important?**
Normalization is a process of organizing data in a relational database to reduce

redundancy and improve data integrity. It involves breaking down large tables into smaller related tables while ensuring that data remains consistent and efficient to query.

**Importance:**

- Eliminates Data Redundancy
- Improves Data Integrity & Consistency
- Enhances Query Performance
- Prevents Anomalies (Insertion, Update, Deletion Issues)
- What are the advantages of using an RDBMS?
- Data consistency and integrity

**16. What are the advantages of using an RDBMS?**

While spreadsheets (like Excel or Google Sheets) are useful for small-scale data management, they have limitations when dealing with large datasets, complex relationships, or multi-user environments. Databases, especially Relational Database Management Systems (RDBMS), offer several advantages over spreadsheets.

- **Limitations of Spreadsheets**

1. **Scalability Issues** – Spreadsheets slow down with large amounts of data.
2. **Data Integrity Issues** – No strict rules to maintain data accuracy.
3. **Multi-User Collaboration Challenges** – Spreadsheets can cause conflicts when multiple users edit simultaneously.

**17. What are some common RDBMS examples?**

Several RDBMS solutions are widely used in the industry for managing structured data efficiently.

- **MySQL :** Open-source, widely used for web applications (e.g., WordPress, Facebook).
- **PostgreSQL:** Advanced open-source RDBMS known for its robustness and support for complex queries.
- **Oracle Database:** Enterprise-grade database used in large-scale applications like banking and ERP systems.
- **Microsoft SQL Server:** RDBMS by Microsoft, used in corporate environments for secure data management.
- **SQLite:** Lightweight, file-based database commonly used in mobile applications and embedded systems.

**18. What are constraints in an RDBMS?**

Constraints in a Relational Database Management System (RDBMS) are rules applied to table columns to maintain data integrity, accuracy, and reliability. They ensure that only valid and consistent data is entered into a database.

**Constraints in RDBMS**

- **Primary Key Constraint:** Ensures that a column (or a combination of columns) uniquely identifies each record in a table.
- **Foreign Key Constraint:** Establishes a relationship between two tables by linking a column to the primary key of another table.
- **Unique Constraint:** Ensures that all values in a column are distinct (but allows NULL values).
- **Not Null Constraint:** Prevents a column from having NULL values.
- **Check Constraint:** Ensures that values in a column meet specific conditions.
- **Default Constraint:** Assigns a default value to a column when no value is provided.

## Section 3: NoSQL Databases

### 19. What is NoSQL, and why was it developed?

NoSQL (Not Only SQL) is a type of database that provides a flexible and scalable way to store and retrieve data, unlike traditional relational databases (RDBMS). It is designed to handle large amounts of unstructured, semi-structured, or structured data efficiently.

**Why Was NoSQL Developed?**

- Handling Big Data: Traditional RDBMS struggled to manage massive amounts of data generated by modern applications (social media, IoT, e-commerce).
- Scalability Needs: RDBMS scales vertically (upgrading hardware), but NoSQL scales horizontally (adding servers) for better performance.

### 20. How does NoSQL differ from RDBMS?

- **Scalability:** RDBMS scales vertically (more CPU/RAM), while NoSQL scales horizontally (adds more servers).
- **Flexibility:** RDBMS requires a fixed schema, while NoSQL can handle dynamic and evolving data models.
- **Performance:** NoSQL is faster for large-scale applications like social media, IoT, and big data, while RDBMS is ideal for transactions and structured data.

### 21. What are the different types of NoSQL databases?

- Document-based: MongoDB
- Key-Value: Redis
- Column-family: Cassandra
- Graph-based: Neo4j

### 22. Explain document-based databases with an example.

Document-based databases store data in a flexible, JSON-like format called documents, which can contain structured or semi-structured data. These databases do not require a predefined schema, making them ideal for handling dynamic and evolving data.

- **Flexible Schema:** We can add new fields (e.g., "discount" or "stock") without modifying the entire database structure.
- **Fast Queries:** Retrieves entire product information with a single query instead of multiple table joins.

**Example:** MongoDB

Consider an e-commerce application storing product details in a document-based database like MongoDB.

**23. What are Key-Value NoSQL databases, and where are they used?**

Key-Value databases store data as a collection of key-value pairs, similar to a dictionary or hash table. They are optimized for fast read/write operations and are highly scalable.

**Example: Redis**

A shopping cart system in an online store can use Redis for quick storage and retrieval of user session data:

"user123": { "cart": ["item101", "item202", "item303"], "total_price": 59.99 }

**Where are Key-Value Databases Used?**

- Caching (e.g., Redis for storing frequently accessed data)
- Session Management (e.g., tracking user logins)

**24. What is eventual consistency in NoSQL databases?**

Eventual Consistency is a property of distributed NoSQL databases ensuring that all replicas of data will become consistent over time, but not immediately. Instead of enforcing strict consistency like SQL databases, NoSQL prioritizes availability and performance.

**Example:**

- In a global social media platform, a user updates their profile picture.
- At first, some users might see the old profile picture while others see the new one.

After a short time, all database copies are updated, ensuring consistency across the system.

**25. What is horizontal scaling, and why is it useful in NoSQL?**

Horizontal scaling (scale-out) refers to adding more machines (servers or nodes) to a database system to handle increased load, rather than upgrading a single machine's hardware (which is vertical scaling). This approach is commonly used in NoSQL databases to distribute data across multiple nodes efficiently.

- **Example of Horizontal Scaling:**

  Imagine an e-commerce platform where millions of users browse and purchase products simultaneously. Instead of upgrading a single powerful server (which has limits), the system distributes the workload across multiple servers each server handles a portion of the data, reducing bottlenecks.

- **Benefits of Horizontal Scaling in NoSQL:**
  **1.** Improved Performance – More nodes mean requests are handled faster, reducing latency.
  **2.** Better Fault Tolerance – If one server fails, others continue functioning, ensuring availability.
  **3.** Infinite Scalability – Easily add more servers to handle increasing data and user load.
  **4.** Cost-Effective – Adding multiple smaller servers is often cheaper than upgrading a single powerful machine.

**26. What are the advantages and disadvantages of NoSQL?**

- **Advantages of NoSQL**:
  1. Scalability – Easily scales horizontally across multiple servers.
  2. Flexibility – Schema-less design allows changes without affecting existing data.
  3. High Availability – Distributed architecture ensures uptime even if some nodes fail.
  4. Fast Performance – Optimized for read/write-heavy workloads.
  5. Big Data & Real-time Processing – Handles unstructured data efficiently.
- **Disadvantages of NoSQL:**
  1. Lack of Standardization – No universal query language like SQL (except for some like MongoDB's aggregation framework).
  2. Eventual Consistency – Not always immediately consistent across replicas.
  3. Complex Queries – Not as powerful as SQL for complex joins and aggregations.
  4. Memory Consumption – Some NoSQL databases (e.g., Redis) store data in memory, requiring more RAM.

## Section 4: Comparing RDBMS & NoSQL + DBMS Advantages

**27. What is the main difference between SQL and NoSQL databases?**

The main difference between SQL (Structured Query Language) databases and NoSQL (Not Only SQL) databases lies in their data structure, scalability, and usage scenarios:

**Data Structure:**

- SQL Databases: Store data in structured tables with rows and columns. They follow a fixed schema with predefined relationships.

- NoSQL Databases: Store data in various flexible formats like key-value pairs, documents, wide-columns, or graphs. They do not require a fixed schema.

**Scalability:**

- SQL Databases: Scale vertically by upgrading hardware (adding more RAM, CPU, etc.).
- NoSQL Databases: Scale horizontally by distributing data across multiple servers (sharding).

**Use Cases:**

- SQL Databases: Best suited for applications needing complex queries, structured data, and ACID transactions (e.g., banking, enterprise applications).
- NoSQL Databases: Ideal for handling large volumes of unstructured or semi-structured data, real-time applications, and big data (e.g., social media, IoT, content management).

**Example**:

- A bank managing customer transactions with strict consistency uses an SQL database like MySQL.
- A social media platform storing user posts, likes, and comments uses a NoSQL database like MongoDB.

**28. When should you choose RDBMS over NoSQL?**

- Structured Data: When data follows a fixed schema (e.g., customer records, financial data).
- ACID Compliance Required: When transactions must be reliable (e.g., banking, e-commerce).
- Strong Data Relationships: When data has complex relationships (e.g., employee management systems).
- Data Consistency is Crucial: When strict consistency is needed over high availability (e.g., inventory systems).
- Standardized Querying: When SQL-based querying and analytics are essential.
- Regulatory Compliance: When meeting security and compliance requirements is mandatory

**29. What are the advantages of database security in DBMS?**

- Data Protection: Prevents unauthorized access and data breaches.
- Access Control: Ensures only authorized users can access or modify data.
- Data Integrity: Prevents accidental or malicious data corruption.
- Compliance: Helps meet legal and industry regulations (e.g., GDPR, HIPAA).
- Prevention of SQL Injection: Protects against cyber-attacks like SQL injection.
- Backup & Recovery Support: Ensures data is recoverable in case of failure.
- Auditing & Monitoring: Tracks user activity to detect suspicious behavior.

**30. Why is database backup and recovery important in DBMS?**

Database backup and recovery are critical components of a Database Management System (DBMS) that ensure data safety, integrity, and availability. Since databases store crucial information such as financial records, customer details, and business transactions, any loss or corruption can lead to severe consequences. Backup and recovery mechanisms help mitigate risks associated with hardware failures, software crashes, cyber-attacks, accidental deletions, or natural disasters.

**1. Importance of Database Backup**

A database backup is a copy of the database that can be restored in case of failure. It serves as a safety net to prevent data loss and ensure business continuity.

**2. Types of Database Backups**

There are different types of database backups, depending on how data is stored and restored:

- Full Backup: A complete copy of the entire database, taken periodically.
- Incremental Backup: Backs up only the data that has changed since the last backup, reducing storage usage.
- Differential Backup: Backs up changes made since the last full backup.
- Physical Backup: Copies the database files directly from disk storage.

**3. Key Benefits of Database Recovery:**

- Restoring Lost Data: Helps recover accidentally deleted or corrupted data.
- Maintaining Data Consistency: Ensures the database remains in a valid and usable state.

**4. Database Recovery Techniques**

- Rollback: Reverts the database to its previous state before an erroneous transaction was committed.
- Redo Logs: Reapplies committed transactions after a crash to ensure consistency.
- Undo Logs: Rolls back uncommitted transactions after a failure.
- Mirroring: Maintains real-time copies of a database on multiple servers for instant recovery.

**Example Scenario**

Imagine an e-commerce company that processes thousands of transactions daily. If a sudden server crash occurs without a backup, all transaction records may be lost. However, with proper backups and recovery mechanisms, the company can restore its data and continue operations with minimal disruption.