**1. Single Data (Attribute)**

* **What**: A single piece of information in a database column, e.g., "Employee Name" or "Employee ID".
* **Example**: In a table, Employee Name = John Doe is an attribute.

**2. All Rows (Entity)**

* **What**: Rows in a table represent entities (real-world objects or items).
* **Example**: Each row in an Employee table represents a single employee.

**3. Primary Key**

* **What**: A unique identifier for a row in a table.
* **Why**: Ensures each row is unique.
* **Example**: Employee ID in an Employee table.

**4. Foreign Key**

* **What**: A column in one table referencing the primary key in another table.
* **Why**: Establishes relationships between tables.
* **Example**: Department\_ID in the Employee table referencing the Department table.

**5. Unique Key**

* **What**: Ensures all values in a column are unique but allows one NULL value.
* **Example**: Email column in a Users table.

**6. Foreign Key Reference**

* **What**: Links data between two tables.
* **Example**: Order\_ID in an Order\_Items table refers to ID in the Orders table.

**7. Benefits of RDBMS**

* **What**: Advantages of Relational Database Management Systems.
  + Data integrity
  + Reduced redundancy
  + Scalability
  + Structured Query Language (SQL)
* **Example**: MySQL or PostgreSQL for managing customer data.

**8. CQRS (Command Query Responsibility Segregation)**

* **What**: Separates "read" operations (queries) from "write" operations (commands).
* **Why**: Optimizes performance and scalability.
* **Example**: An e-commerce site: querying product details (read) vs updating stock (write).

**9. Phantom Read**

* **What**: Occurs when a transaction reads rows that another transaction inserts or deletes later.
* **Example**: A query for orders might initially return 10 rows, but after a new insert, it shows 11 rows.

**10. Serializable & Repeatable Read**

* **What**: Transaction isolation levels.
  + **Serializable**: Prevents all anomalies but is slower.
    - **Example**: Ensures no new rows are inserted during a transaction.
  + **Repeatable Read**: Prevents non-repeatable reads (data changes), but phantom reads can occur.
    - **Example**: Querying orders twice gives the same results.

**11. Eventual Consistency**

* **What**: In distributed systems, data is not instantly synchronized across nodes but will eventually become consistent.
* **Why**: Improves availability and performance.
* **Example**: In social media, a post may not appear immediately for all users.

**12. Sharding**

* **What**: Dividing a database into smaller pieces (shards) for better performance.
* **Why**: Scales horizontally by distributing data.
* **Example**: Splitting customer data by region.

**13. CAP Theorem**

* **What**: In distributed systems, you can achieve at most two out of three:
  + **Consistency**: All nodes have the same data.
  + **Availability**: System responds to all requests.
  + **Partition Tolerance**: System works despite network splits.
* **Example**: NoSQL databases like MongoDB favor availability and partition tolerance over consistency.

**14. NoSQL (Availability & Partition Tolerance)**

* **What**: Non-relational databases prioritize speed and fault tolerance.
* **Example**: Amazon DynamoDB for highly scalable, distributed data storage.