1. ACID Properties in Database Transactions

Answer:

Atomicity: This property ensures that all parts of a transaction are treated as a single, indivisible unit. Either all operations of the transaction are completed successfully, or none are applied at all. For example, in a bank transaction that transfers money from account A to account B, atomicity ensures that if any part of the transaction fails, the entire transaction fails, and the database state remains unchanged.

Consistency: Ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants. That is, any data written to the database must be valid according to all defined rules, including constraints, cascades, and triggers.

Isolation: This property ensures that concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. This means the intermediate state of one transaction should not be visible to other transactions.

Durability: Once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. This means the changes made by the transaction are permanently stored in the database and cannot be undone without another transaction.

These properties ensure reliable, safe, and predictable management of data in a database system and are crucial for maintaining the integrity and reliability of data.

2. Differences Between SQL and NoSQL Databases

Answer:

SQL Databases are relational, table-based databases, whereas NoSQL Databases are either document-based, key-value pairs, graph databases, or wide-column stores. SQL databases are designed to maintain strict consistency (ACID compliance). They use structured query language and are ideal for complex queries. SQL databases are generally not suited for hierarchical data storage.

NoSQL Databases offer more flexibility with data models and are designed for distributed data stores with horizontal scalability. They typically provide eventual consistency and are best suited for large data sets with simple lookup queries.

Scalability: SQL databases scale vertically, but NoSQL databases are designed to scale horizontally.

Use Cases:

SQL: Used in systems where complex transactions and relationships are a requirement. Ideal for banking systems or any system that requires high transactionality.

NoSQL: Used for large data sets such as big data applications and real-time web apps. Suitable for applications that require rapid, non-relational reads and writes.

3. Normalization and Normal Forms

Answer:

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves decomposing a table into smaller tables and defining relationships between them to minimize redundancy and dependency.

First Normal Form (1NF): Ensures that all table columns have atomic (indivisible) values and each record is unique. Example: Storing multiple phone numbers in one column violates 1NF.

Second Normal Form (2NF): Achieved when it is on 1NF and all non-key attributes are fully functional on the primary key. Example: If an attribute depends only on part of a multi-column primary key, break it into separate tables.

Third Normal Form (3NF): A table is in 3NF if it is in 2NF and all its columns are not transitively dependent on the primary key. Example: If a non-primary attribute depends on other non-primary attributes rather than on the primary key, move it to a separate table.

4. Entity-Relationship (ER) Model

Answer:

The ER Model is a high-level conceptual data model that defines data elements and their relationships. Key components include:

Entities: Objects or things within the domain that have a distinct existence. Example: Employee, Department.
Attributes: Properties that define an entity. Example: An Employee entity might have attributes like Employee_ID, Name, Age.
Relationships: Associations between entities. Example: EMPLOYEES work in DEPARTMENTS.
The ER model helps in designing a database at the conceptual level and can be directly transformed into a relational database schema.

5. Indexing in Databases

Answer:

Indexing is a data structure technique used to quickly locate and access the data in a database. Indexes are created using one or more columns of a database table, providing fast retrieval of data.

Types of Indexes:

Primary Index: Unique index on primary key fields.

Secondary Index: An index that is not a primary index, used to prevent extra lookups.
Clustered Index: Reorders the physical order of the table and search based on key values.
Indexes significantly improve query performance by minimizing the number of disk accesses required when a query is processed, but they can also lead to increased maintenance as the table data changes.

These answers provide a foundational understanding of key concepts in database management systems, crucial for any study or application in this field.