Database Management System

Question 1: What is Database?

Answer: A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

Question 2: What is Database Management System?

Answer: Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data.

Question 3: What is a Relational Database?

Answer:

- A relational database typically stores information in tables containing specific pieces and types of data. For example, a shop could store details of their customers' names and addresses in one table and details of their orders in another. This form of data storage is often called structured data.
- Relational databases use Structured Query Language (SQL). In relational database design, the database usually contains tables consisting of columns and rows. When new data is added, new records are inserted into existing tables or new tables are added. Relationships can then be made between two or more tables.
- Relational databases work best when the data they contain doesn't change very often, and when accuracy is crucial. Relational databases are, for instance, often found in financial applications.

Question 5: What is NoSQL/Non-Relational Databases?

Answer:

- Non-relational databases (often called NoSQL databases) are different from traditional relational databases in that they store their data in a non-tabular form. Instead, non-relational databases might be based on data structures like documents. A document can be highly detailed while containing a range of different types of information in different formats. This ability to digest and

organize various types of information side-by-side makes non-relational databases much more flexible than relational databases.
- Non-relational databases are often used when large quantities of complex and diverse data need to be organized.
- Non-relational databases often perform faster because a query doesn’t have to view several tables in order to deliver an answer, as relational datasets often do. Non-relational databases are therefore ideal for storing data that may be changed frequently or for applications that handle many different kinds of data. They can support rapidly developing applications requiring a dynamic database able to change quickly and to accommodate large amounts of complex, unstructured data.

Question 4: Why is MongoDB not called a relational database?

Answer: Relations between data will always be there. NoSQL came later and has flexibility unlike old databases known as relational database. Now this does not mean those are bad, this just means new one provided more flexibility and didn't require rigid structures. So that's why it is called non-relational.

Question 5: SQL vs NoSQL? When to use which one?

Answer:

Keypoint	SQL	NoSQL
Model	SQL databases are vertically scalable while NoSQL databases are horizontally scalable.	Document: JSON documents, Key-value: key-value pairs, Wide-column: tables with rows and dynamic columns, Graph: nodes and edges
Purpose	General purpose	Document: general purpose, Key-value: large amounts of data with simple lookup queries, Wide-column: large amounts of data with predictable query patterns, Graph: analyzing and traversing relationships between connected data
Schemas	Rigid	Flexible
Multi-Record ACID Transactions	Supported	Not All, some like MongoDB do

Keypoint	SQL	NoSQL
Data to Object Mapping	Requires ORM (object-relational mapping)	Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages.
Sharding	Most including newer ones do not support automatic sharding	MongoDB has auto sharding
Scalability	Vertically Scalable	Horizontaly Scalable
Nesting	To find	To find
Data Limit	In Postgres, Row: 1.6 TB Field: 1 GB, No. of rows: Unlimited	Document can be at max of 16 MB

Question 7: Define Consistency, Redudancy and Integrity in DBMS.

Answer:

- **Consistency:** Consistency in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
- **Redudancy:** Data redundancy occurs when the same piece of data is stored in two or more separate places
- **Integrity:** The term data integrity refers to the accuracy and consistency of data. When creating databases, attention needs to be given to data integrity and how to maintain it. Maintaining data integrity means making sure the data remains intact and unchanged throughout its entire life cycle

Question 8: What is Entity Relationship model?

Answer: ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

Question 9: What is total participation in Relationships?

Answer: It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set. That is why, it is also called as mandatory participation.

Question 10: What are weak entity sets?

Answer: An entity type should have a key attribute which uniquely identifies each entity in the entity set, but there exists some entity type for which key attribute can't be defined. These are called Weak Entity type.

The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets and the entity sets which have a primary key are known as strong entity sets.

Question 11: Define Candidate Key, Primary Key, Super Key, Foreign Key, Unique Key, Alternate Key.

Answer:

- **Candidate Key-** Candidate key is the minimal set of attribute (**columns**) which can derive each and every column of the remaining table.
- **Primary Key-** One of the candidate key is choosen as primary key.
- **Alternate Key-** All candidate keys other than primary keys are known as alternate keys.
- **Super Key-** Super key is the set of attribute (**columns**) which can derive each and every column of the remaining table. Not that difference between Candidate Key and Super Key is that super key need not be minimal.
- **Unique Key-** Any attribute which has all unique value but it can be NULL as well.
- **Foriegn Key-** Primary key to refer to some other row in different table (Primary key of that row of different table).

Question 12: What is functional dependancy and state Armstrong Axioms.

Answer: Function dependency is a relation between two entities that if we know data of one entity then we can derive data of functionally dependent entity as well.

It is represented as $A \rightarrow B$, i.e. B is functionally dependent of A

1. **Reflexivity-** $A \rightarrow A$
2. **Tranitivity-** $A \rightarrow B$ and $B \rightarrow C \implies A \rightarrow C$
3. **Augmentation-** $X \rightarrow Y \implies XZ \rightarrow YZ$

Question 13: What are anomalies and explain update, insert and delete anomalies.

Answer: Anomalies are defined as something that deviates from what is standard, normal, or expected.

- **Update Anomalies-** Let say we have 10 columns in a table out of which 2 are called employee Name and employee address. Now if one employee changes it's location then we would have to update the table. But the problem is, if the table is not normalized one employee can have multiple entries and while updating all of those entries one of them might get missed.

- **Insert Anomalies-** Let’s say we have a table that has 4 columns. Student ID, Student Name, Student Address and Student Grades. Now when a new student enroll in school, even though first three attributes can be filled but 4th attribute will have NULL value because he doesn't have any marks yet.
- **Deletion Anomalies-** This anomaly indicates unnecessary deletion of important information from the table. Let’s say we have student’s information and courses they have taken as follows (student ID,Student Name, Course, address). If any student leaves the school then the entry related to that student will be deleted. However, that deletion will also delete the course information even though course depends upon the school and not the student.

Question 14: Normalisation and its forms

Answer: Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

1. 1^{st} Normal Form
    - Every Attribute must be atomic
2. 2^{nd} Normal Form
    - No partial dependency (Student,Subject ID,Time) $\rightarrow$ (Class Room)
      (Subject ID, Time) $\rightarrow$ (Class Room)
3. 3^{rd} Normal Form
    - Non Prime attribute should not be functionally dependent on other non prime attributes
4. Boyce–Codd Normal Form
    - Prime attribute should not be functionally dependent on anything

Question 15: What is a transaction and what are ACID properties of transaction in DBMS?

Answer: A database transaction symbolizes a unit of work performed within a database management system against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in a database.

- **Atomic-** Atomicity guarantees that each transaction is treated as a single "unit", which either succeeds completely, or fails completely: if any of the statements constituting a transaction fails to complete, the entire transaction fails and the database is left unchanged
- **Consistency-** Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining database invariants: any data written

to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

- **Isolation-** Transactions are often executed concurrently (e.g., multiple transactions reading and writing to a table at the same time). Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially.
- **Durability-** Durability guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure (e.g., power outage or crash). This usually means that completed transactions (or their effects) are recorded in non-volatile memory.

Question 16: What is Database Sharding?

Answer: A database shard is a horizontal partition of data in a database or search engine. Each individual partition is referred to as a shard or database shard. Each shard is held on a separate database server instance, to spread load. Tip: If we are starting with scaling, more common methods should be tried first like indexing as sharding is difficult to implement.

Question 17: What is SQL and types of commands?

Answer: SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

- **Data Definition Language-** Data Definition Language, DDL, is the part of SQL that allows a database user to create and restructure database objects, such as the creation or the deletion of a table.
    - For example- CREATE, ALTER, DROP
- **Data Manipulation Language-** Data Manipulation Language, DML, is the part of SQL used to manipulate data within objects of a relational database.
    - For example- INSERT, UPDATE, DELETE
- **Data Query Language-** Data Query Language (DQL) is the most concentrated focus of SQL for modern relational database users.
    - For example- SELECT
- **Data Control Language-** Data control commands in SQL allow you to control access to data within the database.
    - For example- ALTER PASSWORD, GRANT, REVOKE

Question 18: What is database indexing?

Answer: Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

- **Clustering Index-** Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.
- **Prime Index-** Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.
- **Secondary Index-** Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

Question 19: What is dense indexing and sparse indexing?

Answer:

- **Dense Indexing-** In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.
- **Sparse Indexing-** In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

Question 20: What are different states of transaction?

Answer:

- **Active State-** A transaction enters into an active state when the execution process begins. During this state read or write operations can be performed.
- **Partially Committed State-** A transaction goes into the partially committed state after the end of a transaction.
- **Committed State-** When the transaction is committed to state, it has already completed its execution successfully. Moreover, all of its changes are recorded to the database permanently.
- **Failed State-** A transaction considers failed when any one of the checks fails or if the transaction is aborted while it is in the active state.
- **Terminated State-** State of transaction reaches terminated state when certain transactions which are leaving the system can't be restarted.

Question 21: What is dirty read problem?

Answer: Consider two transactions where initial $A = 10$

Step	T_1	T_2

Step	T_1	T_2
1	$Read(A)$	$-$
2	$A = A + 1$	$-$
3	$-$	$Read(A)$
4	$-$	$Commit$
5	$Fail$	$-$
6	$Rollback$	$-$

Now A after step 2 is 11 and T_2 reads A from shared buffer, thus reading 11 and committing that but T_1 fails and makes T_2 commit inconsistent.

Question 22: What is unrepeatable read problem?

Answer: Consider two transactions where initial $A = 10$

Step	T_1	T_2
1	$Read(A)$	$-$
2	$-$	$Read(A)$
3	$A = A + 1$	$-$
4	$-$	$Read(A)$
5	$-$	$Failure$
6	$Commit$	$-$

Notice that T_2 while solving a problems reads 2 inconsistent values of A which causes a problem, this is called unrepeatable read problem

Phantom Read- If in step 3 instead of update, we delete the variable then this causes phantom read problem.

Question 23: What is a schedule?

Answer: A chronological execution sequence of a transaction is called a schedule, for example table in above two answers.

Question 24: What is Conflict Serializibility?

Answer:

- **Conflict Serializable:** A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
- **Conflicting operations:** Two operations are said to be conflicting if all conditions satisfy:
  - They belong to different transactions
  - They operate on the same data item
  - At Least one of them is a write operation