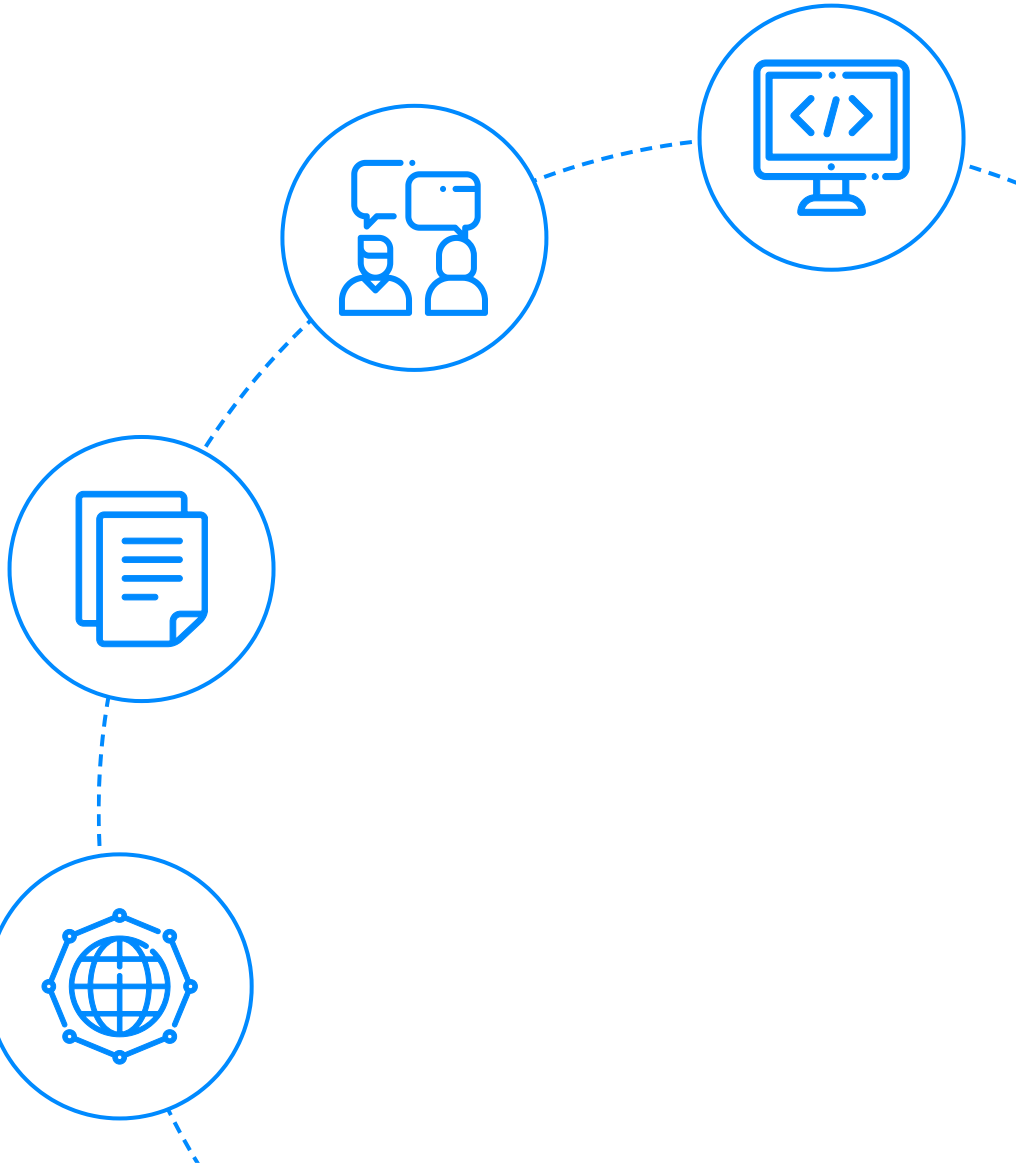
InterviewBit

DBMS Interview Questions



To view the live version of the page, click here.

Contents

Advanced DBMS Interview Questions

(.....Continued)

18. Explain the difference between a 2-tier and 3-tier architecture in a DBMS.

Let's get Started

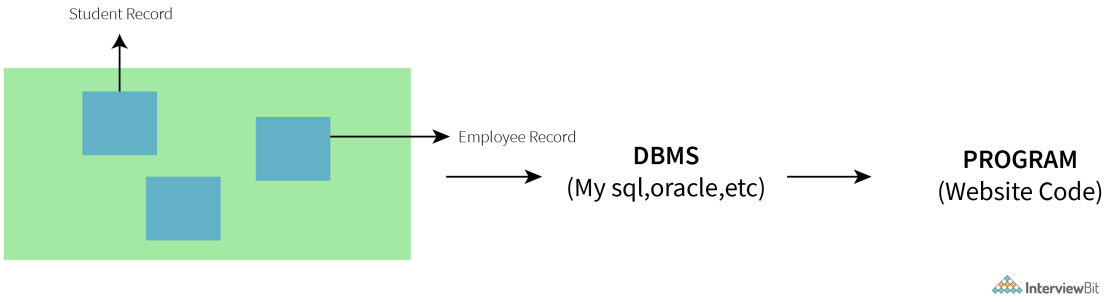
To consolidate your knowledge and concepts in DBMS, here we've listed the most commonly asked DBMS interview questions to help you ace your interview!

Basic DBMS Interview Questions

1. What is meant by DBMS and what is its utility? Explain RDBMS with examples.

As the name suggests DBMS or Database Management System is a set of applications or programs that enable users to create and maintain a database. DBMS provides a tool or an interface for performing various operations such as inserting, deleting, updating, etc. into a database. It is software that enables the storage of data more compactly and securely as compared to a file-based system. A DBMS system helps a user to overcome problems like data inconsistency, data redundancy, etc. in a database and makes it more convenient and organized to use it.

Examples of popular DBMS systems are file systems, XML, Windows Registry, etc.



RDBMS stands for Relational Database Management System and was introduced in the 1970s to access and store data more efficiently than DBMS. RDBMS stores data in the form of tables as compared to DBMS which stores data as files. Storing data as rows and columns makes it easier to locate specific values in the database and makes it more efficient as compared to DBMS.

Examples of popular RDBMS systems are MySQL, Oracle DB, etc.

2. What is meant by a database?

A Database is an organized, consistent, and logical collection of data that can easily be updated, accessed, and managed. Database mostly contains sets of tables or objects (anything created using create command is a database object) which consist of records and fields. A tuple or a row represents a single entry in a table. An attribute or a column represents the basic units of data storage, which contain information about a particular aspect of the table. DBMS extracts data from a database in the form of queries given by the user.

3. Mention the issues with traditional file-based systems that make DBMS a better choice?

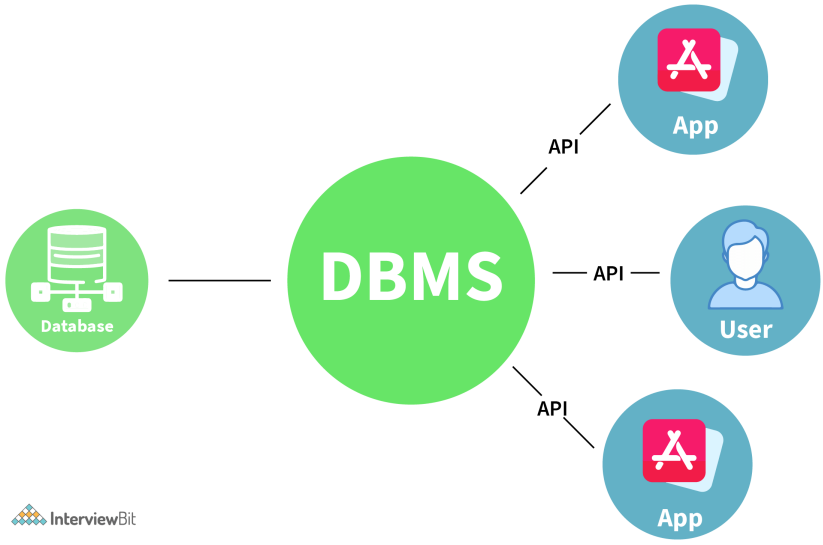
The absence of indexing in a traditional file-based system leaves us with the only option of scanning the full page and hence making the access of content tedious and super slow. The other issue is redundancy and inconsistency as files have many duplicate and redundant data and changing one of them makes all of them inconsistent. Accessing data is harder in traditional file-based systems because data is unorganized in them.

Another issue is the lack of concurrency control, which leads to one operation locking the entire page, as compared to DBMS where multiple operations can work on a single file simultaneously.

Integrity check, data isolation, atomicity, security, etc. are some other issues with traditional file-based systems for which DBMSs have provided some good solutions.

4. Explain a few advantages of a DBMS.

Following are the few advantages of using a DBMS.



- **Data Sharing:** Data from a single database can be simultaneously shared by multiple users. Such sharing also enables end-users to react to changes quickly in the database environment.
- **Integrity constraints:** The existence of such constraints allows storing of data in an organized and refined manner.
- **Controlling redundancy in a database:** Eliminates redundancy in a database by providing a mechanism that integrates all the data in a single database.
- **Data Independence:** This allows changing the data structure without altering the composition of any of the executing application programs.
- **Provides backup and recovery facility:** It can be configured to automatically create the backup of the data and restore the data in the database whenever required.
- **Data Security:** DBMS provides the necessary tools to make the storage and transfer of data more reliable and secure. Authentication (the process of giving restricted access to a user) and encryption (encrypting sensitive data such as OTP, credit card information, etc.) are some popular tools used to secure data in a DBMS.

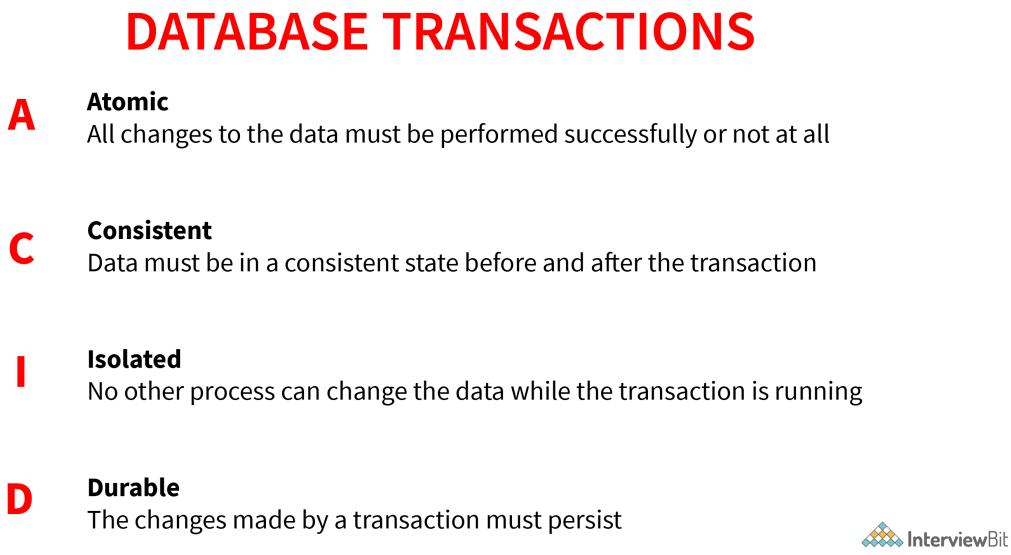
5. Explain different languages present in DBMS.

Following are various languages present in DBMS:

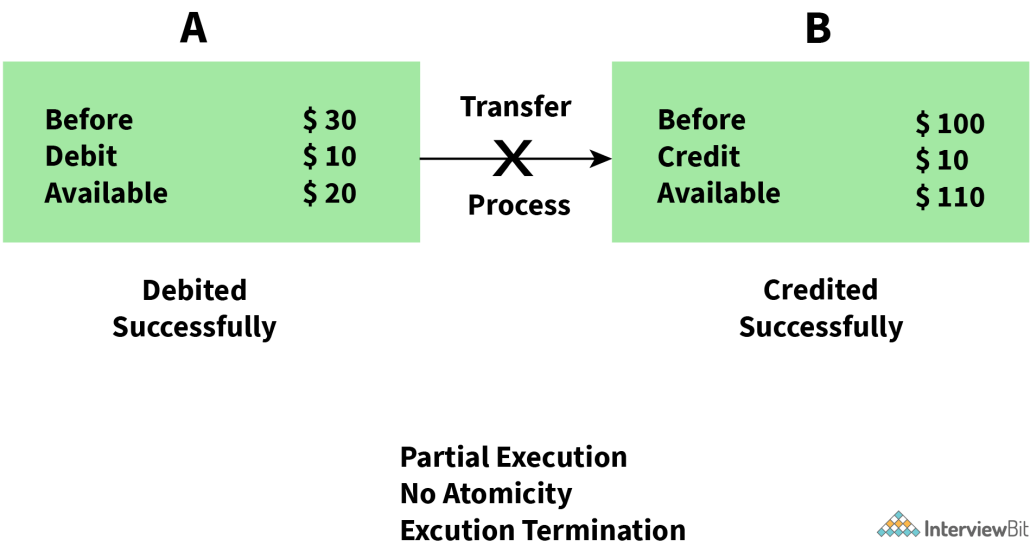
- **DDL(Data Definition Language):** It contains commands which are required to define the database.
  E.g., CREATE, ALTER, DROP, TRUNCATE, RENAME, etc.
- **DML(Data Manipulation Language):** It contains commands which are required to manipulate the data present in the database.
  E.g., SELECT, UPDATE, INSERT, DELETE, etc.
- **DCL(Data Control Language):** It contains commands which are required to deal with the user permissions and controls of the database system.
  E.g., GRANT and REVOKE.
- **TCL(Transaction Control Language):** It contains commands which are required to deal with the transaction of the database.
  E.g., COMMIT, ROLLBACK, and SAVEPOINT.

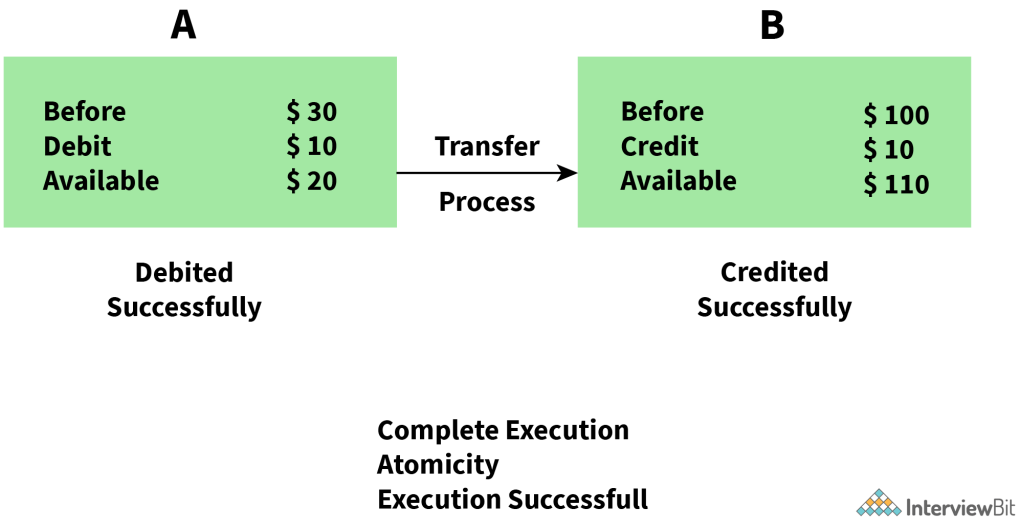
6. What is meant by ACID properties in DBMS?

ACID stands for Atomicity, Consistency, Isolation, and Durability in a DBMS these are those properties that ensure a safe and secure way of sharing data among multiple users.

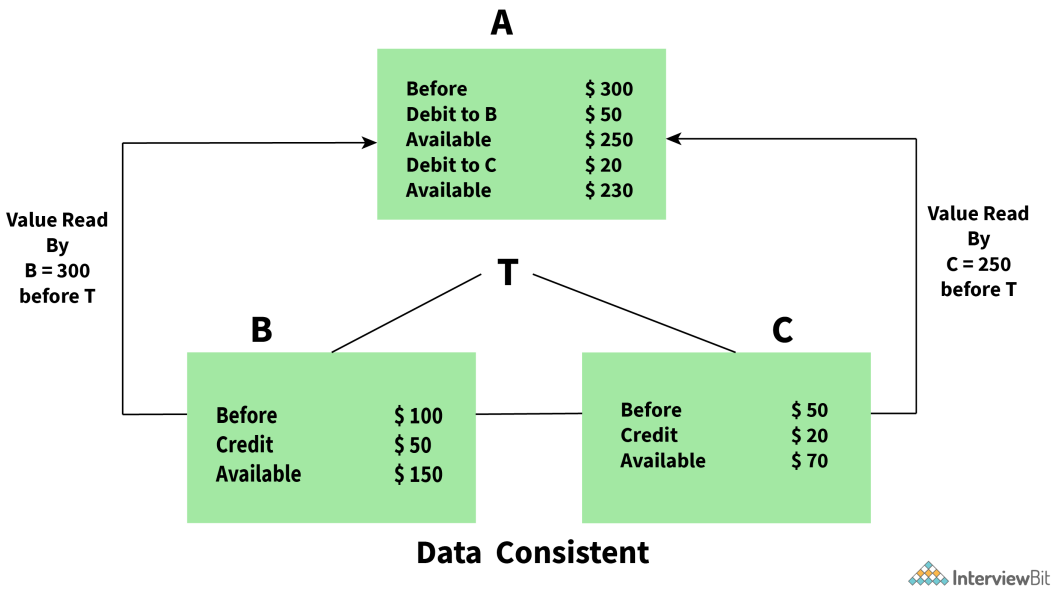


- **Atomicity**: This property reflects the concept of either executing the whole query or executing nothing at all, which implies that if an update occurs in a database then that update should either be reflected in the whole database or should not be reflected at all.

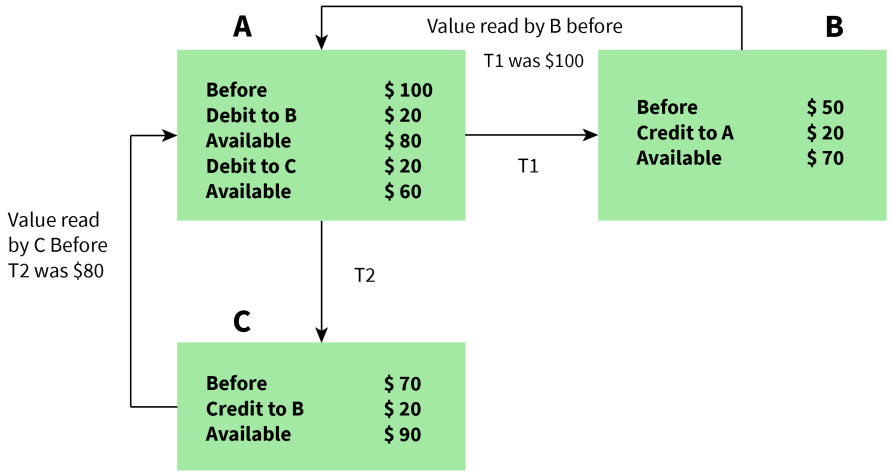




- **Consistency:** This property ensures that the data remains consistent before and after a transaction in a database.



- **Isolation:** This property ensures that each transaction is occurring independently of the others. This implies that the state of an ongoing transaction doesn't affect the state of another ongoing transaction.



Isolation - Independent execution T1 and T2 by A

InterviewBit

- **Durability:** This property ensures that the data is not lost in cases of a system failure or restart and is present in the same state as it was before the system failure or restart.

7. Are NULL values in a database the same as that of blank space or zero?

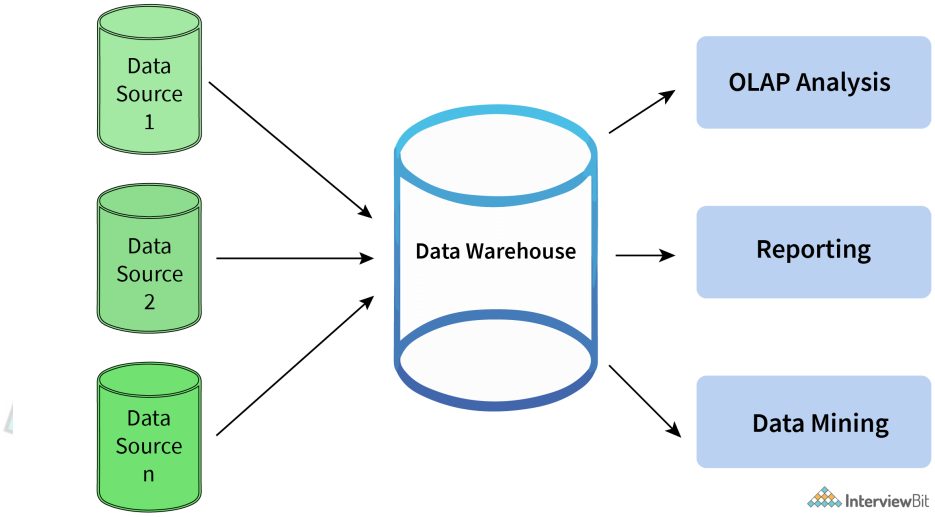
No, a NULL value is very different from that of zero and blank space as it represents a value that is assigned, unknown, unavailable, or not applicable as compared to blank space which represents a character and zero represents a number.

Example: NULL value in “number_of_courses” taken by a student represents that its value is unknown whereas 0 in it means that the student hasn’t taken any courses.

Intermediate DBMS Interview Questions

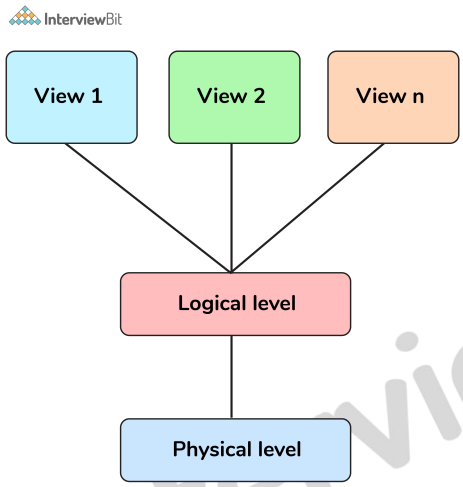
8. What is meant by Data Warehousing?

The process of collecting, extracting, transforming, and loading data from multiple sources and storing them into one database is known as data warehousing. A data warehouse can be considered as a central repository where data flows from transactional systems and other relational databases and is used for data analytics. A data warehouse comprises a wide variety of organization's historical data that supports the decision-making process in an organization.



9. Explain different levels of data abstraction in a DBMS.

The process of hiding irrelevant details from users is known as data abstraction. Data abstraction can be divided into 3 levels:

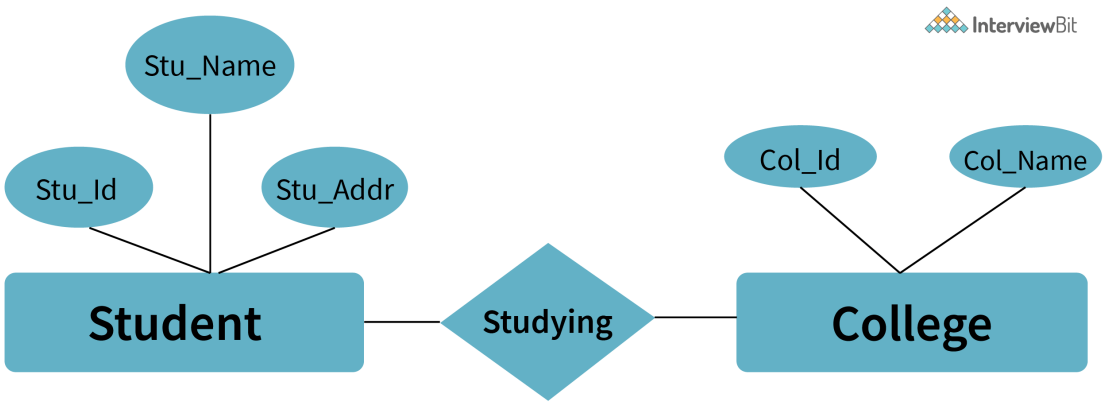


Three levels of data abstraction

- **Physical Level:** it is the lowest level and is managed by DBMS. This level consists of data storage descriptions and the details of this level are typically hidden from system admins, developers, and users.
- **Conceptual or Logical level:** it is the level on which developers and system admins work and it determines what data is stored in the database and what is the relationship between the data points.
- **External or View level:** it is the level that describes only part of the database and hides the details of the table schema and its physical storage from the users. The result of a query is an example of View level data abstraction. A view is a virtual table created by selecting fields from one or more tables present in the database.

10. What is meant by an entity-relationship (E-R) model? Explain the terms Entity, Entity Type, and Entity Set in DBMS.

An entity-relationship model is a diagrammatic approach to a database design where real-world objects are represented as entities and relationships between them are mentioned.



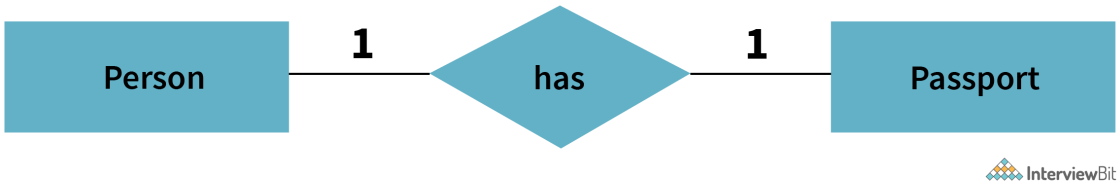
Sample E-R Diagram

- **Entity:** An entity is defined as a real-world object having attributes that represent characteristics of that particular object. For example, a student, an employee, or a teacher represents an entity.
- **Entity Type:** An entity type is defined as a collection of entities that have the same attributes. One or more related tables in a database represent an entity type. Entity type or attributes can be understood as a characteristic which uniquely identifies the entity. For example, a student represents an entity that has attributes such as student_id, student_name, etc.
- **Entity Set:** An entity set can be defined as a set of all the entities present in a specific entity type in a database. For example, a set of all the students, employees, teachers, etc. represent an entity set.

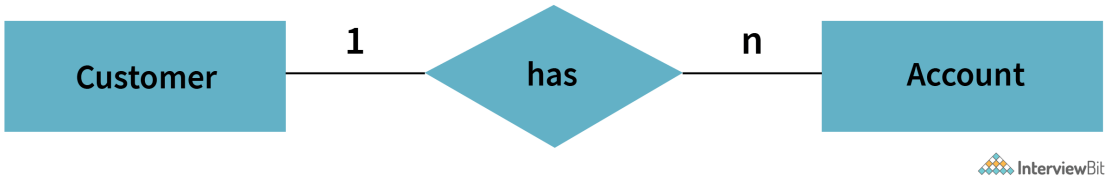
11. Explain different types of relationships amongst tables in a DBMS.

Following are different types of relationship amongst tables in a DBMS system:

- **One to One Relationship:** This type of relationship is applied when a particular row in table X is linked to a singular row in table Y.

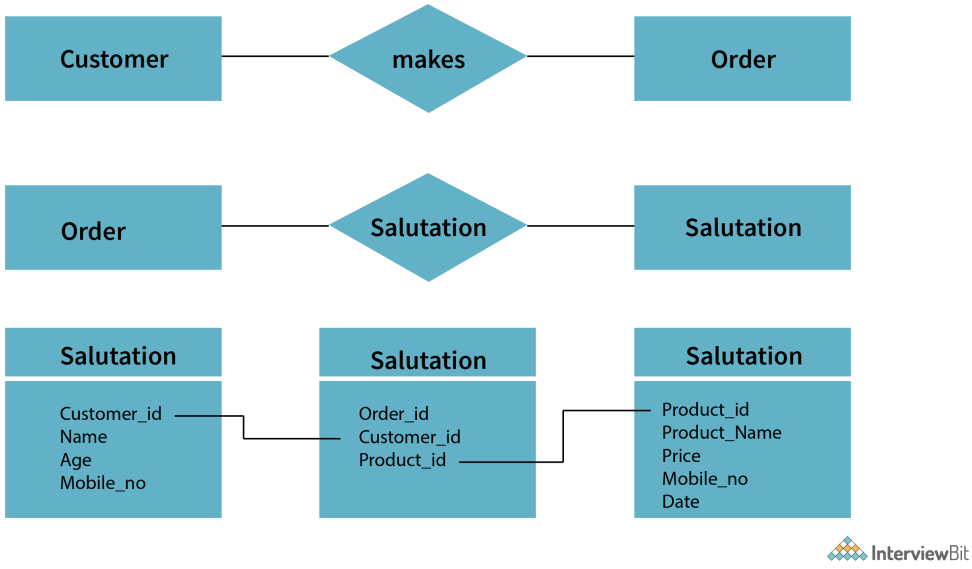


- **One to Many Relationship:** This type of relationship is applied when a single row in table X is related to many rows in table Y.



- **Many to Many Relationship:** This type of relationship is applied when multiple rows in table X can be linked to multiple rows in table Y.

- **Self Referencing Relationship:** This type of relationship is applied when a particular row in table X is associated with the same table.



12. Explain the difference between intension and extension in a database.

Following is the major difference between intension and extension in a database:

- **Intension:** Intension or popularly known as database schema is used to define the description of the database and is specified during the design of the database and mostly remains unchanged.
- **Extension:** Extension on the other hand is the measure of the number of tuples present in the database at any given point in time. The extension of a database is also referred to as the snapshot of the database and its value keeps changing as and when the tuples are created, updated, or destroyed in a database.

13. Explain the difference between the DELETE and TRUNCATE command in a DBMS.

DELETE command: this command is needed to delete rows from a table based on the condition provided by the WHERE clause.

- It deletes only the rows which are specified by the WHERE clause.
- It can be rolled back if required.
- It maintains a log to lock the row of the table before deleting it and hence it's slow.

TRUNCATE command: this command is needed to remove complete data from a table in a database. It is like a DELETE command which has no WHERE clause.

- It removes complete data from a table in a database.
- It can be rolled back even if required.
- It doesn't maintain a log and deletes the whole table at once and hence it's fast.

14. What is a lock. Explain the major difference between a shared lock and an exclusive lock during a transaction in a database.

A database lock is a mechanism to protect a shared piece of data from getting updated by two or more database users at the same time. When a single database user or session has acquired a lock then no other database user or session can modify that data until the lock is released.

- **Shared Lock:** A shared lock is required for reading a data item and many transactions may hold a lock on the same data item in a shared lock. Multiple transactions are allowed to read the data items in a shared lock.
- **Exclusive lock:** An exclusive lock is a lock on any transaction that is about to perform a write operation. This type of lock doesn't allow more than one transaction and hence prevents any inconsistency in the database.

15. What is meant by normalization and denormalization?

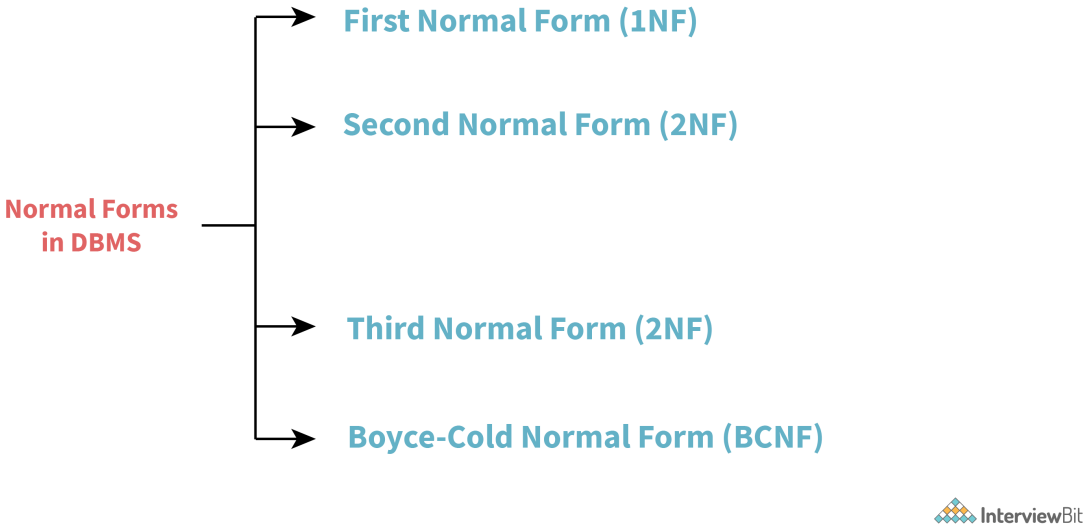
Normalization is a process of reducing redundancy by organizing the data into multiple tables. Normalization leads to better usage of disk spaces and makes it easier to maintain the integrity of the database.

Denormalization is the reverse process of normalization as it combines the tables which have been normalized into a single table so that data retrieval becomes faster. JOIN operation allows us to create a denormalized form of the data by reversing the normalization.

Advanced DBMS Interview Questions

16. Explain different types of Normalization forms in a DBMS.

Following are the major normalization forms in a DBMS:



Full Names	Physical Address	Movies Rented	Salutation
Janet Jones	First street Plot No 4	Pirates. the Caribbean Clash of the Titans	Ms.
Robert Phil	3rd street 34	Forgetting Sarah Marshal Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

InterviewBit

Considering the above Table-1 as the reference example for understanding different normalization forms.

- 1NF: It is known as the first normal form and is the simplest type of normalization that you can implement in a database. A table to be in its first normal form should satisfy the following conditions:
  - Every column must have a single value and should be atomic.
  - Duplicate columns from the same table should be removed.
  - Separate tables should be created for each group of related data and each row should be identified with a unique column.

Full Names	Physical Address	Movies Rented	Salutation
Janet Jones	First street Plot No 4	Pirates. the Caribbean	Ms.
Janet Jones	First street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

InterviewBit

Table-1 converted to 1NF form

- **2NF:** It is known as the second normal form. A table to be in its second normal form should satisfy the following conditions:
  - The table should be in its 1NF i.e. satisfy all the conditions of 1NF.
  - Every non-prime attribute of the table should be fully functionally dependent on the primary key i.e. every non-key attribute should be dependent on the primary key in such a way that if any key element is deleted then even the non_key element will be saved in the database.

Membership ID	Full Names	Physical Address	Salutation
1	Janet Jones	First street Plot No 4	Ms.
2	Robert Phil	3rd street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

InterviewBit

Membership ID	Movies Rented
1	Pirates. the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

InterviewBit

Breaking Table-1 into 2 different tables to move it to 2NF.

- 3NF: It is known as the third normal form. A table to be in its second normal form should satisfy the following conditions:
  - The table should be in its 2NF i.e. satisfy all the conditions of 2NF.
  - There is no transitive functional dependency of one attribute on any attribute in the same table.

Membership ID	Full Names	Physical Address	SalutationID
1	Janet Jones	First street Plot No 4	2
2	Robert Phil	3rd street 34	1
3	Robert Phil	5th Avenue	1

InterviewBit

Membership ID	Movies Rented
1	Pirates. the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Salutation ID	Salutation
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

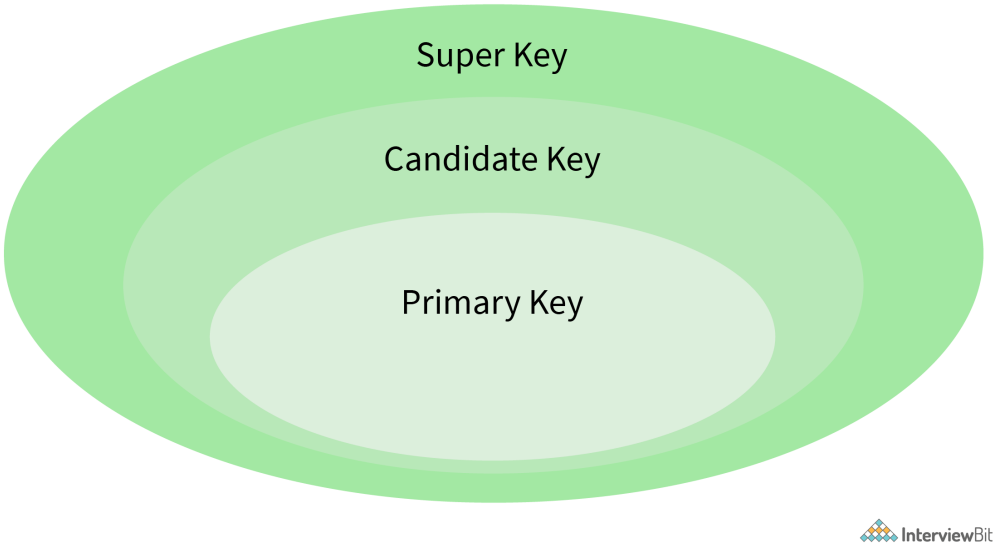
Breaking Table-1 into 3 different tables to move it to 3NF.

- **BCNF:** BCNF stands for Boyce-Codd Normal Form and is an advanced form of 3NF. It is also referred to as 3.5NF for the same reason. A table to be in its BCNF normal form should satisfy the following conditions:
  - The table should be in its 3NF i.e. satisfy all the conditions of 3NF.
  - For every functional dependency of any attribute A on B (A->B), A should be the super key of the table. It simply implies that A can't be a non-prime attribute if B is a prime attribute.

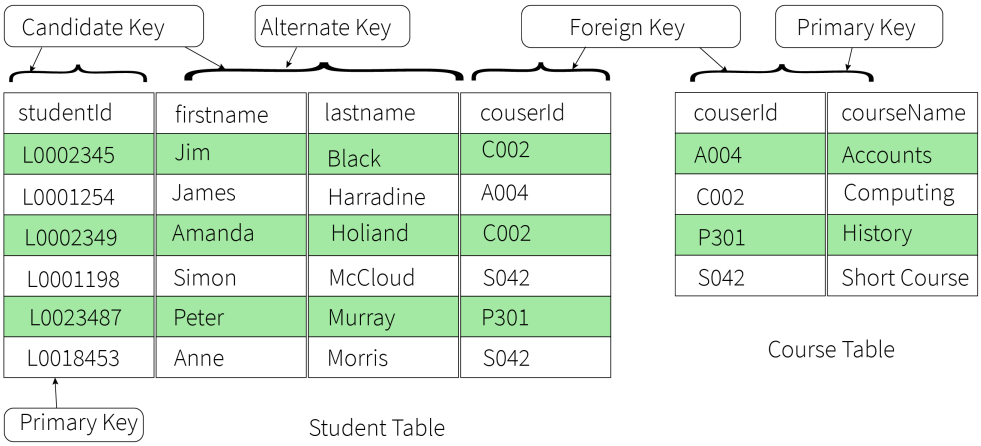
17. Explain different types of keys in a database.

There are mainly 7 types of keys in a database:

- **Candidate Key:** The candidate key represents a set of properties that can uniquely identify a table. Each table may have multiple candidate keys. One key amongst all candidate keys can be chosen as a primary key. In the below example since studentId and firstName can be considered as a Candidate Key since they can uniquely identify every tuple.
- **Super Key:** The super key defines a set of attributes that can uniquely identify a tuple. Candidate key and primary key are subsets of the super key, in other words, the super key is their superset.



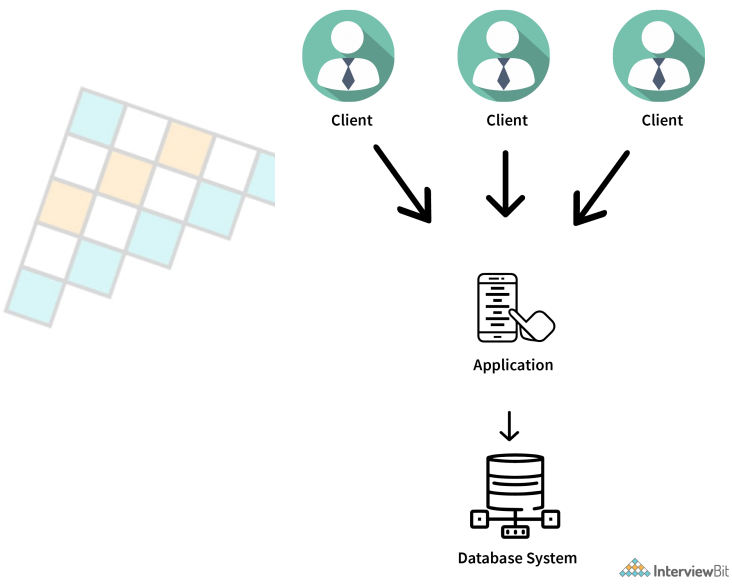
- **Primary Key:** The primary key defines a set of attributes that are used to uniquely identify every tuple. In the below example studentId and firstName are candidate keys and any one of them can be chosen as a Primary Key. In the given example studentId is chosen as the primary key for the student table.
- **Unique Key:** The unique key is very similar to the primary key except that primary keys don't allow NULL values in the column but unique keys allow them. So essentially unique keys are primary keys with NULL values.
- **Alternate Key:** All the candidate keys which are not chosen as primary keys are considered as alternate Keys. In the below example, firstname and lastname are alternate keys in the database.
- **Foreign Key:** The foreign key defines an attribute that can only take the values present in one table common to the attribute present in another table. In the below example courseId from the Student table is a foreign key to the Course table, as both, the tables contain courseId as one of their attributes.
- **Composite Key:** A composite key refers to a combination of two or more columns that can uniquely identify each tuple in a table. In the below example the studentId and firstname can be grouped to uniquely identify every tuple in the table.



Relationship Between Keys

18. Explain the difference between a 2-tier and 3-tier architecture in a DBMS.

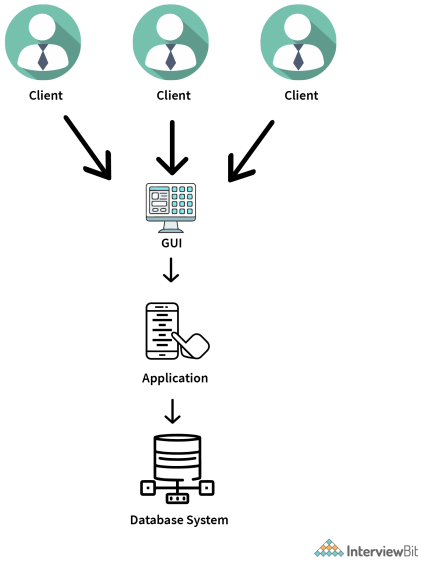
The **2-tier architecture** refers to the client-server architecture in which applications at the client end directly communicate with the database at the server end without any middleware involved.
Example – Contact Management System created using MS-Access or Railway Reservation System, etc.



The above picture represents a 2-tier architecture in a DBMS.

The **3-tier architecture** contains another layer between the client and the server to provide GUI to the users and make the system much more secure and accessible. In this type of architecture, the application present on the client end interacts with an application on the server end which further communicates with the database system.

Example – Designing registration form which contains a text box, label, button or a large website on the Internet, etc.



The above picture represents a 3-tier architecture in a DBMS.

Recommended Tutorials:

SQL Interview Questions
SQL Server Interview Questions
MySQL Interview Questions
MongoDB Interview Questions
PL SQL Interview Questions

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions

IOTA ACADEMY · Jun 5 · 9 min read

50 Top DBMS Interview Questions and Answers

Database Management Systems (DBMS) are a crucial part of managing and organizing data efficiently. Whether you're preparing for an interview or looking to strengthen your database knowledge, these **50 top DBMS interview questions and answers** will help you understand key concepts.

1. **What is DBMS?**

A Database Management System (DBMS) is software that allows users to store, retrieve, and manage data efficiently. It ensures **data security, consistency, and integrity** while allowing multiple users to access data simultaneously. Examples include **MySQL, PostgreSQL, MongoDB, and Oracle**.

2. **What are the types of DBMS?**

DBMS is categorized into four types:

- Hierarchical DBMS – Organizes data in a tree-like structure (e.g., IBM Information Management System).
- Network DBMS – Uses a graph structure to model relationships (e.g., IDMS).
- Relational DBMS (RDBMS) – Uses tables (relations) to store data (e.g., MySQL, PostgreSQL).
- Object-oriented DBMS (OODBMS) – Stores data as objects (e.g., db4o, ObjectDB).

3. **What is a relational database?**

A **relational database** stores data in structured tables with **rows and columns**, where relationships between tables are established using **primary keys and foreign keys**.

4. **What is SQL?**

SQL (Structured Query Language) is used to **create, retrieve, update, and delete (CRUD)** data in relational databases. It includes commands like **SELECT, INSERT, UPDATE, DELETE, JOIN, and GROUP BY**.

5. What is the difference between DBMS and RDBMS?

Feature	DBMS	RDBMS
Data Storage	Stores data as files	Stores data in tables
Data Relationships	No relationships	Uses primary and foreign keys
Normalization	Not supported	Supports normalization
Example	XML, JSON databases	MySQL, SQL Server

6. What is a primary key?

A **primary key** is a unique identifier for each record in a table. It **must be unique and cannot contain NULL values**.

7. What is a foreign key?

A **foreign key** is a column in one table that references the **primary key** of another table, establishing a relationship between them.

8. What is normalization in DBMS?

Normalization is the process of organizing data to eliminate redundancy and improve integrity. The different normal forms include:

- **1NF (First Normal Form)** – Removes duplicate columns.
- **2NF (Second Normal Form)** – Ensures all columns are dependent on the primary key.
- **3NF (Third Normal Form)** – Removes transitive dependencies.

9. What are the ACID properties in DBMS?

ACID stands for:

- **Atomicity** – Transactions are **all-or-nothing**.
- **Consistency** – Ensures valid database state before and after transactions.
- **Isolation** – Transactions do not interfere with each other.
- **Durability** – Changes remain permanent even after a system failure.

10. What is indexing in DBMS?

Indexing improves query performance by creating data structures that enable faster searches. Types include:

- **Clustered Index** – Rearranges table records for faster access.
- **Non-clustered Index** – Stores index separately without affecting table order.

11. What is the difference between DELETE and TRUNCATE ?

In SQL, DELETE and TRUNCATE differ primarily in how they eliminate data from a table. In addition to allowing the use of a WHERE clause to restrict which rows to delete, DELETE is used to remove individual entries. Because it is a logged operation, every deleted row is noted in the transaction log, enabling a rollback if necessary. In comparison, TRUNCATE is a minimally logged action that is much faster but irreversible because it eliminates every entry from a database without the need for a WHERE clause. Therefore, TRUNCATE is the greatest option for rapidly removing all data from a table without the need for a rollback, whereas DELETE is appropriate for selective and recoverable deletions.

Feature	DELETE	TRUNCATE
Removes	Specific records	All records
WHERE clause	Allowed	Not allowed
Rollback	Possible	Not possible

12. What are the advantages of using a DBMS?

A DBMS (Database Management System) helps manage data efficiently by ensuring **data integrity, security, consistency, and easy access**. It eliminates redundancy through normalization, supports multiple users, and provides backup and recovery features.

13. What is data abstraction in DBMS?

Data abstraction refers to the process of hiding the **complexity of database structures** from users while exposing only necessary details. It has three levels: **Physical level (storage details), Logical level (structure of data), and View level (user-specific representation).**

14. What is a database schema?

A database schema is the **blueprint of a database** that defines its structure, including tables, relationships, constraints, and indexes. It is a logical representation that does not contain actual data but describes how the data is organized.

15. What is a database instance?

A database instance is the **actual set of data** stored in the database at a particular moment. While the schema remains constant, the database instance keeps changing as data is added, updated, or deleted.

16. What are tuples and attributes in DBMS?

A **tuple** is a row (record) in a table, representing a single entity. An **attribute** is a column in a table, representing a property of an entity. For example, in an "Employees" table, "Employee_ID" is an attribute, and each employee's details form a tuple.

17. What are the different types of SQL commands?

SQL commands are classified into:

- **DDL (Data Definition Language)** – CREATE, ALTER, DROP (modifies schema).
- **DML (Data Manipulation Language)** – INSERT, UPDATE, DELETE (modifies data).
- **DCL (Data Control Language)** – GRANT, REVOKE (controls access).
- **TCL (Transaction Control Language)** – COMMIT, ROLLBACK (manages transactions).

18. What is the difference between HAVING and WHERE clauses?

- **WHERE** filters rows **before** aggregation and works with individual records.
- **HAVING** filters rows **after** aggregation and works with **GROUP BY** results.

19. What is the difference between COUNT, SUM, AVG, MIN, and MAX?

These are **aggregate functions** in SQL:

- **COUNT()** – Returns the number of rows.
- **SUM()** – Adds up numerical values.
- **AVG()** – Calculates the average value.
- **MIN()** – Finds the smallest value.
- **MAX()** – Finds the largest value.

20. What is a self-join in SQL?

A self-join is a join where a table is joined with itself using **aliases**. It is used to compare rows within the same table.

21. What are the advantages of normalization?

Normalization eliminates **data redundancy**, ensures **data consistency**, and improves **database efficiency**. It divides large tables into smaller, related tables and establishes relationships, reducing anomalies like insertion, update, and deletion anomalies.

22. Explain different normal forms up to BCNF.

- **1NF (First Normal Form)** – Removes duplicate columns, ensures atomicity.
- **2NF (Second Normal Form)** – Removes partial dependencies, ensuring all non-key attributes depend on the full primary key.
- **3NF (Third Normal Form)** – Removes transitive dependencies, ensuring no non-key attribute depends on another non-key attribute.
- **BCNF (Boyce-Codd Normal Form)** – Ensures that every determinant is a candidate key, eliminating anomalies further.

23. What is a transaction in DBMS?

A transaction is a **logical unit of work** in a database that consists of multiple operations (INSERT, UPDATE, DELETE). It follows **ACID properties (Atomicity, Consistency, Isolation, Durability)** to ensure reliability and consistency.

24. What is a dirty read in DBMS?

A dirty read occurs when one transaction reads uncommitted data from another transaction, which may later be rolled back. This can cause inconsistency.

25. What is two-phase locking (2PL)?

Two-phase locking is a concurrency control mechanism with two phases:

- **Growing phase** – Transactions acquire locks but do not release them.
- **Shrinking phase** – Transactions release locks but do not acquire new ones.

This prevents conflicts and ensures serializability.

26. What is indexing, and why is it important?

Indexing improves **query performance** by creating a data structure (index) that allows faster retrieval of records. It reduces the number of disk accesses required for a query.

27. What is the difference between a clustered and non-clustered index?

- **Clustered Index** – Sorts the actual data in the table (one per table).
- **Non-clustered Index** – Creates a separate structure pointing to data (multiple allowed).

28. What is the difference between SQL and NoSQL databases?

SQL databases are **relational**, structured, and use a predefined schema, while NoSQL databases are **non-relational**, flexible, and store unstructured or semi-structured data.

29. What is CAP theorem?

CAP theorem states that a distributed database can achieve only **two out of three** properties:

- **Consistency** – Every node has the same data.
- **Availability** – System remains operational.
- **Partition Tolerance** – The system works despite network failures.
-

30. What is OLAP, and how is it different from OLTP?

- **OLAP (Online Analytical Processing)** is designed to analyze vast volumes of historical data and run sophisticated queries. It facilitates processes like trend analysis, forecasting, and reporting and is primarily utilized in data warehouses. Due to their read-heavy operating optimization, OLAP systems enable users, including business analysts and decision-makers, to analyze data in a variety of ways (e.g., sales by region, product, and time).
- **OLTP (Online Transaction Processing),** is utilized in daily company operations and is centered on managing transactional data in real-time. Numerous brief, straightforward transactions can be processed quickly and accurately thanks to its optimization for speedy insert, update, and delete operations. Applications where data consistency and speed are crucial, such as banking, retail, and airline reservation systems, frequently use OLTP systems.

31. What is SQL injection, and how can it be prevented?

SQL injection is a security attack where malicious SQL code is injected into queries, allowing unauthorized access. It can be prevented using:

- **Prepared statements** instead of dynamic queries.
- **Validating and sanitizing user input.**
- **Using least-privilege database access.**

32. What are the different types of databases?

Databases can be classified as **relational (SQL), non-relational (NoSQL), hierarchical, network, object-oriented, and cloud databases.** Relational databases use structured data, while NoSQL databases handle unstructured and semi-structured data.

33. What is a foreign key?

A **foreign key** is an attribute in one table that refers to the **primary key** of another table. It enforces referential integrity by ensuring valid relationships between records.

34. What is referential integrity?

Referential integrity ensures that a foreign key value in a child table always refers to an existing record in the parent table. It prevents orphan records and maintains consistency.

35. What is the difference between a file system and a DBMS?

A file system stores data in individual files with no structured relationships, leading to redundancy. A DBMS organizes data efficiently, ensuring consistency, security, and easy retrieval through queries.

36. What are candidate keys?

Candidate keys are attributes that can uniquely identify a record. One candidate key is chosen as the primary key, while others can be alternate keys.

37. What is a super key?

A super key is any combination of attributes that uniquely identify a record. A primary key is a minimal super key with no unnecessary attributes.

38. What is a composite key?

A composite key consists of two or more attributes that together uniquely identify a record, especially when a single attribute cannot be a unique identifier.

39. What is functional dependency?

A functional dependency occurs when the value of one attribute uniquely determines another. For example, in a table of employees, Employee_ID → Employee_Name means each Employee_ID determines a unique Employee_Name.

40. What is denormalization?

Denormalization is the process of combining tables to improve query performance by reducing joins. It increases redundancy but enhances read performance in large-scale databases.

41. What are anomalies in DBMS?

Anomalies occur due to redundancy and poor database design. They include:

- Insertion anomaly – Inability to insert data due to missing values.
- Update anomaly – Redundant data requires multiple updates.
- Deletion anomaly – Deleting one record removes necessary data.

42. What is the difference between 3NF and BCNF?

3NF removes transitive dependencies, ensuring that non-key attributes depend only on the primary key. BCNF (Boyce-Codd Normal Form) is stricter, ensuring that every determinant is a candidate key, eliminating further redundancy.

43. What is an entity in DBMS?

An entity is an object with attributes that represent real-world things (e.g., a student, an employee). Entities are stored as records in tables.

44. What is a deadlock in DBMS?

A deadlock occurs when two transactions wait for each other to release resources, causing a standstill. Deadlock prevention techniques include timeouts and resource ordering.

45. What is a rollback in DBMS?

A rollback undoes all changes made by a transaction that encounters an error or failure, ensuring database consistency.

46. What is a savepoint in DBMS?

A savepoint marks a specific point within a transaction, allowing partial rollbacks instead of restarting the entire transaction.

47. What is serializability?

Serializability ensures that transactions execute in a way that produces the same result as if they were executed sequentially, maintaining consistency.

48. What is the difference between an index and a view?

An index speeds up searches by creating a data structure, while a view is a virtual table based on query results without storing data separately.

49. What are B-Trees and B+ Trees?

B-Trees and B+ Trees are used in **indexing** to organize and retrieve data efficiently. **B+ Trees store all data in leaf nodes, while B-Trees store keys at multiple levels.**

50. What is the difference between a materialized view and a regular view?

A **materialized view** stores query results physically and updates periodically, while a **regular view** is just a saved SQL query that retrieves data dynamically.

Conclusion

Having a strong grasp of these **DBMS interview questions** will help you confidently tackle any interview, whether for a **data analyst, database administrator, or software developer role**.

Looking to enhance your database skills? **Join Iota's Courses and master SQL, NoSQL, and more!** 🚀

[dp1]This answer only contain a table explain it in para also
Web Development
[dp2]This is a imp question you can explain it more