RDBMS

Interview Questions

(Practice Project)

PW SKILLS

Interview Questions

Easy

Q1.What is RDBMS?

Answer -
RDBMS stands for Relational database management system this data management tool organizes data into a table with rows and columns, where each row and column represents data for one or more options. RDBMS uses a query model (frame) to manage and update statistical data and allows users to perform operations including editing, manipulating, and querying statistical data. RDBMS's ability to use keys to create organization in data while allowing for good data storage and integrity is one of its distinguishing features. Major RDBMSs include PostgreSQL, Oracle Database, MySQL, and Microsoft sq. RDBMS is widely used for secure and powerful data storage and management in various applications and industries.

Q2.What is the difference between a primary key and a foreign key?

Answer -

Basis	Primary Key	Foreign Key
Definition	A primary key is a unique identifier for each record in a table.	A foreign key establishes a relationship between tables by referencing the primary key of another table.
Basic	Ensures uniqueness and data integrity within a single table.	Establishes relationships and maintains referential integrity between tables.
NULL	Does not allow NULL values in the primary key field.	Allows NULL values in the foreign key field, indicating a missing or unspecified relationship.
Count	Only one primary key can exist per table.	Multiple foreign keys can exist within a table, depending on the relationships with other tables.

Duplication	No duplicate primary key values are allowed within the table.	Can contain duplicate foreign key values, reflecting multiple records associated with the same reference.
Indexing	Primary keys are automatically indexed to enhance data retrieval speed.	Foreign keys can be indexed but are not automatically indexed.
Deletion	Deleting a record with a primary key constraint can affect referential integrity in related tables.	Deleting a record with a foreign key constraint can be restricted or cascaded to maintain integrity.
Insertion	Each new record must have a unique primary key value assigned.	The foreign key can reference an existing primary key value or be NULL if the relationship is optional.
Temporary table	Primary keys can be applied to temporary tables.	Foreign keys can also be applied to temporary tables to establish relationships.
Relationship	Primary keys define the basis for establishing relationships with other tables.	Foreign keys establish relationships and connect data between related tables.

Q3. What is the importance of the indexes?

Answer -

Indexes increase the effectiveness of data retrieval processes, indexes are essential components of database management systems. They act as streamlined data structures that let databases find and retrieve particular rows in a table with ease. There are multiple important reasons why indexes are important:

Query Performance: By lowering the number of rows scanned when retrieving data, indexes improve query performance.
Sorting and Filtering: By supplying pre-sorted values for indexed columns, they speed up the sorting and filtering process.
Data Integrity: By enforcing restrictions on data integrity, indexes help to guarantee accuracy and consistency throughout the database.
Optimization of Joins: By streamlining join procedures between tables, they enable quicker and more effective data retrieval.
Support for Constraints: Indexes uphold data integrity standards by supporting constraints such as primary keys, unique keys, and foreign keys.
Total Efficiency: Well-thought-out indexes enhance database usability and performance, which boosts total efficiency.

Q4. What is the difference between a clustered index and a non-clustered index?

Answer -

The main difference between a clustered index and a non-clustered index lies in how they organize and store data in a table:

Clustered Index:
- Physically orders the data rows in the table based on the indexed column(s).
- Defines the physical order of the table on disk.
- There can only be one clustered index per table.
- Usually created on the primary key column(s), but can be created on any other column(s).
- Since the data is physically stored based on the clustered index, there's no separate storage overhead for the index itself.

Non-Clustered Index:
- Stores a separate list of pointers to the data rows, rather than physically ordering the rows.
- Does not affect the physical order of the table on disk.
- Multiple non-clustered indexes can be created on a single table.
- Ideal for columns frequently used in queries for filtering, sorting, or joining operations.
- Requires additional storage space for the index itself, as it stores pointers to data rows rather than the data itself.

Q5. What are the Types of relationships in RDBMS?

Answer -

There are basically three different kinds of relationships between tables in relational database management systems, or RDBMS:

One-to-One relationship:
A one-to-one relationship, or 1:1 relationship, is one in which every entry in one table has an exact counterpart in another table, and vice versa. In database design, this kind of relationship is somewhat rare and is usually employed when there is a distinct, one-to-one correspondence between two items.

One-to-Many relationship:
A one-to-many relationship, or 1:N relationship, allows each record in one table to have multiple associations with records in another, while each record in the second table has only one association with each record in the first dataset. In relational databases, this is the most prevalent kind of relationship and is frequently used to simulate parent-child or hierarchical relationships.

Many-to-Many relationship:
This type of relationship allows for the association of one record in one table with several records in another table, and vice versa. For this kind of interaction, the many-to-many relationship must be represented by a junction table, sometimes referred to as an associative or connecting table. The primary keys of the two related tables are referenced in foreign key columns found in the junction table

Q6. Are NULL values in a database the same as those of blank space or zero?

Answer -
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.

Q7. What are the most popular Relational Databases?

Answer-

RDBMS is the optimal choice for data management in every business domain. Examples of popular relational database management systems include MySQL, Oracle, SQLite, SQL Server, PostgreSQL, and Microsoft SQL Server.

INTERMEDIATE

Q8.What are the different types of indexes, When should you use an index?

Answer -

There are several types of indexes in relational database management systems (RDBMS), each serving specific purposes:

1. **Clustered Index:** Determines the physical order of data rows in a table.
2. **Non-Clustered Index:** Stores a separate list of pointers to data rows, allowing for efficient data retrieval based on indexed columns.
3. **Unique Index:** Ensures uniqueness of values in indexed columns across the table.
4. **Composite Index:** Combines multiple columns into a single index, useful for queries involving multiple criteria.
5. **Covering Index:** Includes all columns required for a query, minimizing disk I/O.
6. **Partial Index:** Indexes a subset of rows based on a specified condition.

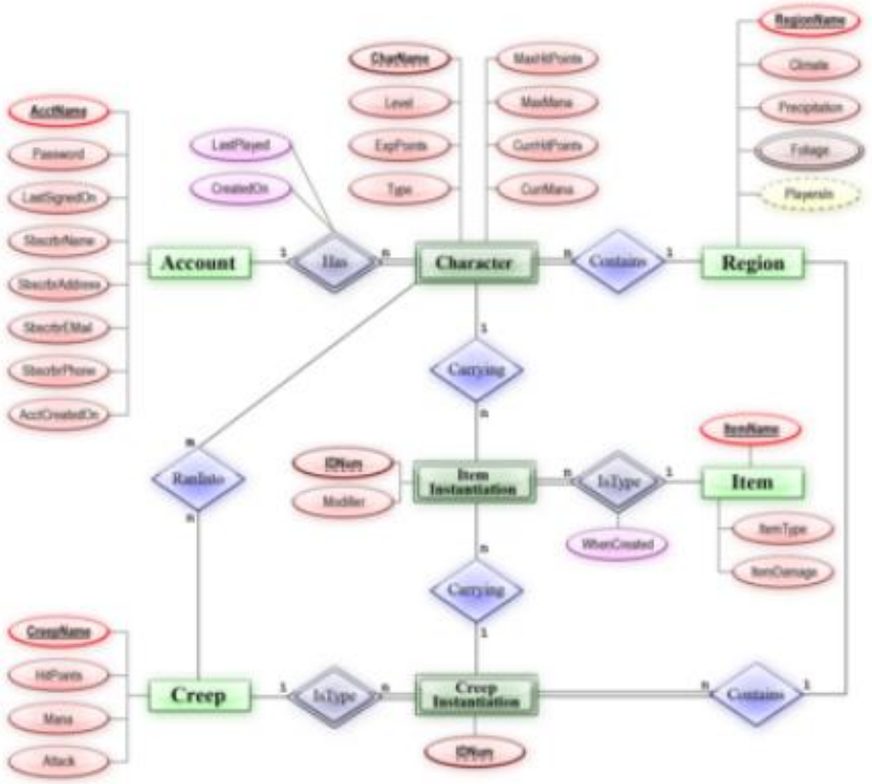
Indexes should be used:

For frequently queried columns.
- In large datasets to improve query performance.
- To enforce unique constraints.
- For sorting and ordering operations.
- On primary and foreign key columns.

Q9.What is the ER Model in RDBMS?

Answer -

An ER model is usually the result of systematic analysis to define and describe what data is created and needed by processes in an area of a business. Typically, it represents records of entities and events monitored and directed by business processes, rather than the processes themselves. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. It can also be expressed in a verbal form, for example: one building may be divided into zero or more apartments, but one apartment can only be located in one building.



Q10.Q. Describe the four ACID properties in the context of database transactions.

Answer -
ACID properties are a set of fundamental principles that ensure the reliability and integrity of data transactions. The acronym ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties are crucial for maintaining data consistency and preventing data corruption in database systems.

Atomicity
Atomicity refers to the principle that a transaction must be treated as a single, indivisible unit of work. In other words, either all the operations within a transaction are executed successfully, or none of them are executed at all. This property ensures that partial updates or incomplete transactions do not occur.

Consistency
Consistency ensures that the database remains in a valid state before and after any transaction. It enforces the rules and constraints defined in the database schema, such as primary keys, foreign keys, and other integrity constraints.

Isolation
Isolation deals with how multiple transactions interact with each other and access shared resources concurrently. It ensures that the execution of one transaction does not interfere with or affect the execution of another concurrent transaction.

Q11. What do you understand about Database Triggers?

Answer -
Database triggers are types of stored procedures that run in response to an event occurring in a database. They are typically associated with changes made to a table's data, like insertions, updates, or deletions. Triggers are useful for maintaining the data integrity in the database. They can adjust the entire database systematically whenever there is a change. Here are some common use cases for triggers:

Triggers are used to maintain an audit file to record changes made in a transaction history table. This makes it easy to track changes and add this information to user reports.
They are a helpful tool for calculating values within columns. For instance, triggers can maintain a TotalSales column on a customer record which would need to be updated every time a sale is made.
Triggers help implement business rules. They inspect all data before running a data manipulation statement, apply any business rules, and ultimately complete the transaction. This helps maintain customer status based on purchase history.

Q12. Q. What is the key difference between a candidate key and a superkey in the context of a relational database?

Answer -

Super Key	Candidate Key
A super key is an attribute, or collection of attributes, that is used to identify each attribute in a relation in a unique way.	Candidate Key is a subset of a super key.
All super keys can’t be candidate keys.	But all candidate keys are super keys.
The criteria for choosing the candidate keys is made up of several super keys together.	The criteria for choosing the primary keys are made up of a variety of candidate keys.
In a relation, number of super keys	While in a relation, number of
is more than number of candidate keys.	candidate keys are less than number of super keys.

Q13. Explain Data Integrity in RDBMS.

Answer-

Data integrity refers to the complete consistency, completeness, and accuracy of data across its entire lifecycle. When data has integrity, mechanisms have been established to ensure that an unauthorized person or software cannot alter data-in-use, data-in-transit, and data-at-rest.

The major types of Data Integrity in an RDBMS are as follows

Integrity Type	Description
Entity Integrity	This one states that a table cannot include any duplicate tuples.
Domain Integrity	This one directs us to save data in any specified column in the proper format, type, and range.
Referential Integrity	This one states that we cannot delete necessary records for accessing other records.
User-Defined Integrity	This one states that the rules given by the user when creating the database are consistently applied and verified before entering data into the database.

Q14.What is the difference between DELETE and TRUNCATE?

Answer-

The below table illustrates the differences between DELETE and TRUNCATE commands

DELETE	TRUNCATE
Specific rows get deleted using the DELETE command (one or more).	While using this command, all the rows in a table get deleted.
It is a DML command (Data Manipulation Language).	It's a DDL (Data Definition Language) command.

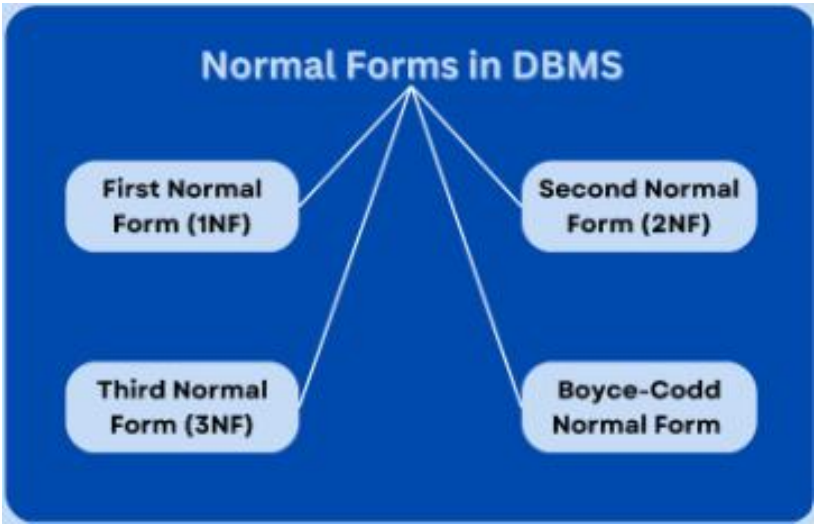
The DELETE command could include a "WHERE clause" for records filtering.	While the TRUNCATE command does not include a "WHERE clause."
A tuple gets locked before being deleted when using the DELETE command.	The data page is locked while this command runs before the table data gets deleted.
The DELETE command eliminates rows one at a time while adding a transaction log entry for each deleted row.	By reallocating the data pages used to store the table data, TRUNCATE Database removes the data from the table. It simply logs the page deallocations in the transaction log.
Compared to TRUNCATE, DELETE is the slower command.	The TRUNCATE command, however, is quicker than the DELETE command.
You require the table's DELETE permission to utilize Delete.	We require at least the ALTER permission on the table before using Truncate.
Using the DELETE Statement on the table leaves the identification of the fewer columns intact.	Identity If an identity column is present in the table, you can reset it to its seed value.
This command also has an active trigger option.	This command does not have an active trigger option.
Compared to Truncate, the DELETE statement uses more transaction spaces.	Truncate statements take up fewer transaction spaces than DELETE statements.

ADVANCE

Q15. What is Normalization? Explain the different types of Normalization forms in RDBMS.

Answer -

Normalization is a process used to organize a database into tables and columns in such a way that redundancy and dependency are minimized. It involves applying a series of rules, called normal forms, to ensure that data is organized efficiently and relationships are well-defined.



First Normal Form (1NF): Ensures that each table has atomic attributes without repeating groups.
Second Normal Form (2NF): Eliminates partial dependencies by ensuring all non-key attributes are fully functionally dependent on the primary key.
Third Normal Form (3NF): Removes transitive dependencies by ensuring that non-key attributes are not dependent on other non-key attributes.
Boyce-Codd Normal Form (BCNF): A stricter form of 3NF where every determinant is a candidate key.

Q16. Describe the technique of denormalization.

Answer -

By purposefully adding redundancy, a technique known as "denormalization" can enhance query performance by lowering the complexity of data retrieval and the necessity for joins. Optimizing read-heavy procedures involves duplicating data or adding redundant columns to tables. In order to improve query performance, denormalization attempts to give up some of the advantages of normalization, such as data integrity and storage efficiency. Denormalization lowers the need for complex joins and calculations during query execution, resulting in faster data retrieval and enhanced system responsiveness. It does this by precomputing aggregated or frequently requested data and storing it redundantly.

Q17.Q. How is data abstraction achieved in RDBMS?

Answer -
Data abstraction is the procedure of concealing irrelevant or unwanted data from the end user.

Data abstraction in RDBMS is achieved through several techniques:

1. **Entity-Relationship (ER) Modeling:** Abstracts real-world entities and their relationships into logical structures.
2. **Table Structures:** Organize data into tables, with rows representing instances and columns defining attributes.
3. **Normalization:** Reduces redundancy and dependency by decomposing tables into smaller, well-defined entities.
4. **SQL (Structured Query Language):** Provides a standardized interface for data manipulation, abstracting database complexity.
5. **Views:** Present subsets of data in a simplified format, enhancing abstraction and usability.
6. **Stored Procedures:** Encapsulate complex logic for abstraction and reusability.

Q18. Explain buffer manager in RDBMS and its importance.

Answer-

In a Relational Database Management System (RDBMS), the Buffer Manager is a critical component responsible for managing the database's memory resources and facilitating efficient data retrieval and storage operations. Its primary role is to bridge the gap between the high-speed main memory (RAM) and the slower disk storage. Here's a detailed explanation of the Buffer Manager, its functions, and its importance:

Functions of the Buffer Manager:

1. **Data Caching:**

The Buffer Manager maintains a cache of frequently accessed data pages in a memory area called the buffer pool. When a query needs data, the Buffer Manager first checks if the data is in the buffer pool (a hit). If the data is not found (a miss), it is loaded from the disk into the buffer pool.

2. **Page Replacement Policies:**

When the buffer pool is full and new data needs to be loaded, the Buffer Manager must decide which existing pages to evict. Common page replacement algorithms include:

Least Recently Used (LRU): Evicts the pages that have not been used for the longest time.
First-In-First-Out (FIFO): Evicts the oldest pages first.

Clock Algorithm: A more efficient version of LRU that uses a circular list and a "clock hand" to track page usage.

3. **Disk I/O Management:**

The Buffer Manager handles all disk I/O operations, reading data from the disk into the buffer pool and writing data from the buffer pool back to the disk. Efficient disk I/O management is crucial because disk operations are much slower than memory operations.

4. **Concurrency Control:**

The Buffer Manager works with the transaction manager to ensure that multiple transactions can read and write data concurrently without conflicts. This involves implementing locking mechanisms to maintain data consistency and isolation between transactions.

5. **Dirty Page Management:**

A dirty page is a page in the buffer pool that has been modified but not yet written to disk. The Buffer Manager tracks dirty pages and ensures they are written to the disk either periodically or when the page is evicted from the buffer pool. This is essential for maintaining data integrity and durability.

6. **Checkpoints:**

To facilitate recovery after a system crash, the Buffer Manager periodically performs checkpoints. During a checkpoint, all dirty pages are written to disk, and a consistent state of the database is saved. This reduces the amount of work needed during recovery by ensuring that the most recent changes are already persisted.

Importance of the Buffer Manager

1. **Performance Improvement:**

The Buffer Manager significantly enhances database performance by minimizing disk I/O operations. Since memory access is much faster than disk access, keeping frequently used data in the buffer pool reduces query response times and overall system latency.

2. **Efficient Resource Utilization:**

By managing the buffer pool effectively, the Buffer Manager ensures optimal use of available memory resources. It balances the need to keep frequently accessed data in memory with the need to free up space for new data.

3. **Data Consistency and Integrity:**

The Buffer Manager plays a key role in maintaining data consistency and integrity. Through locking mechanisms and careful management of dirty pages, it ensures that transactions are processed reliably and that the database can recover to a consistent state after a failure.

4. **Scalability:**

As the database grows, the Buffer Manager helps maintain performance by efficiently managing larger volumes of data. This scalability is crucial for handling increasing workloads and ensuring the database can support a growing number of concurrent users and transactions.