**DataBase Management System Interview Book**

1. **What is a database?**

A database is an organized collection of data, so that it can be easily accessed and managed.

1. **What is RDBMS?**

RDBMS stands for Relational DataBase Management Systems. It is basically a program that allows us to create, delete, and update a relational database. Relational Database is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns.

1. **Difference between RDBMS and DBMS?**

|  |  |
| --- | --- |
| DBMS stores data as a file. | RDBMS stores data in tabular form. |
| Data elements need to be accessed individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |

1. **What is normalization? Explain advantages and disadvantages.**

The process which allows you to winnow out redundant data within your database. This involves restructuring the tables to successively meeting higher forms of Normalization.

**Advantages:**

1. Narrow tables: Having all the more adjusted tables permits your tables to have fewer sections

2. Data consistency: Data consistency means that the data is always real and it is not ambiguous.

3. Data becomes non-redundant: non-redundant means that only original copy of data is available for each user and for every time. There are no multiple copies of the same data

**Disadvantages:**

1. Requires more joins to get the coveted effect. Complex Queries are required.
2. Maintenance overhead. The higher the level of normalization, the more stupendous the number of tables in the database.
3. More tables to join: By spreading out your information into more tables, you expand the need to join tables
4. **What is Functional Dependency?**

If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes

**Advantages of Functional Dependency**

* Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
* It helps you to maintain the quality of data in the database
* It helps you to defined meanings and constraints of databases
* It helps you to identify bad designs
* It helps you to find the facts regarding the database design

1. **Explain the first normal form**

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

1. **Explain the second normal form**

For a table to be in 2NF, there are two requirements

The database is in first normal form

All non key attributes in the table must be functionally dependent on the entire primary key

1. **Explain the third normal form**

There can be no interdependencies among non-key attributes. For a table to be in 3NF, there are two requirements:

The table should be in the second normal form.

No attribute is transitively dependent on the primary key.

1. **Explain the fourth normal form**

Fourth normal form eliminates independent many-to-one relationships between columns.

To be in Fourth Normal Form, a relation must first be in Boyce-Codd Normal Form.

A given relation may not contain more than one multi-valued attribute

1. **Explain the fifth normal form**

Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

1. **Explain CODD’s rules**

**Rule 0: The Foundation Rule**

The database must be in relational form. So that the system can handle the database through its relational capabilities.

**Rule 1: Information Rule**

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

**Rule 2: Guaranteed Access Rule**

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

**Rule 3: Systematic Treatment of Null Values**

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.

**Rule 4: Active/Dynamic Online Catalog based on the relational model**

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

**Rule 5: Powerful language**

• One well defined language to provide all manners of access to data

• Example: SQL

• If file supporting table can be accessed by any manner except a SQL Interface, then a violation

**Rule 6: View Updating Rule**

All views tables can be theoretically updated and must be practically updated by the database systems.

**Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule**

A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

**Rule 8: Physical Data Independence Rule**

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

**Rule 9: Logical Data Independence Rule**

It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

**Rule 10: Integrity Independence Rule**

A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application.

**Rule 11: Distribution Independence Rule**

The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

**Rule 12: Non-Subversion Rule**

The non-submersion rule defines RDBMS as a SQL language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.

1. **What is denormalization? Explain advantages and disadvantages of it?**

Denormalization- Process of combining multiple small tables into few big tables to increase data redundancy in order to increase the performance of select queries. Generally, denormalization is used for OLAP systems. Disadvantage is redundancy of data

**Pros of Denormalization: -**

* Retrieving data is faster since we do fewer joins
* Queries to retrieve can be simpler (and therefore less likely to have bugs), since we need to look at fewer tables.

**Cons of Denormalization: -**

* Updates and inserts are more expensive.
* Denormalization can make update and insert code harder to write.
* Data may be inconsistent.
* Data redundancy necessitates more storage.

1. **What is BCNF?**BCNF does not allow dependencies between attributes that belong to candidate keys. BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form. Third normal form and BCNF are not same if the following conditions are true:

* **The table has two or more candidate keys**
* **At least two of the candidate keys are composed of more than one attribute.**
* **The keys are not disjoint i.e. The composite candidate keys share some attributes**

Example 1 - Address (Not in BCNF)

Scheme {City, Street, ZipCode }

Key1 {City, Street }

Key2 {ZipCode, Street}

No non-key attribute hence 3NF

{City, Street} {ZipCode}

{ZipCode} {City}

Dependency between attributes belonging to a key

**Decomposition Loss of Information**

1.If decomposition does not cause any loss of information, it is called a **lossless** decomposition.

2.If a decomposition does not cause any dependencies to be lost it is called a **dependency-preserving** decomposition.

3.Any table scheme can be decomposed in a lossless way into a collection of smaller schemas that are in BCNF form. However, the dependency preservation is not guaranteed.

4.Any table can be decomposed in a lossless way into 3rd normal form that also preserves the dependencies.

3NF may be better than BCNF in some cases

1. **What is Integrity Constraint?**

**Unique Key**- means only values are allowed (no duplicates) but can have null values.

One table can have more than 1 unique key

**Primary Key**- allows only unique values. Null value is not allowed Unique + Not Null.

One table can have only 1 Primary key

**Not Null** – will not allow null values

**Check Constraint**-define what range of values will be allowed/not allowed for a column

**Foreign Key**- Parent Child Relationship – **Referential Integrity Constraints**

A value is allowed in the child table only if the value is present in the parent table

Column of the parent table which is being referred should be defined as Primary or Unique Key

Child table can have a null value in the foreign key column irrespective of whether parent table column is defined as Primary or Unique Key

**Default-** defines what value column value will take if explicit value for the column is not provided while inserting the data

1. **What is Referential Integrity?**

**Foreign Key**- Parent Child Relationship – Referential Integrity Constraints

* A value is allowed in the child table only if the value is present in the parent table
* Column of the parent table which is being referred should be defined as Primary or Unique Key
* Child table can have a null value in the foreign key column irrespective of whether parent table column is defined as Primary or Unique Key

1. **What is an ER Diagram?**

**ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

1. **What could be the three types of relationships between 2 possible entities?**

**Relationship -** describes an association among entities

**One-to-many** (1:M) relationship

**Many-to-many** (M:N or M:M) relationship

**One-to-one** (1:1) relationship

1. **What is Structured Query Language?**

**Structured Query Language** is a standard Database language which is used to create, maintain and retrieve the relational database. Following are some interesting facts about SQL.

SQL is case insensitive. But it is a recommended practice to use keywords (like **SELECT, UPDATE, CREATE,** etc) in capital letters and use user defined things (like table name, column name, etc) in small letters.

We can write comments in SQL using “–” (double hyphen) at the beginning of any line.

SQL is the programming language for relational databases (explained below) like **MySQL, Oracle, Sybase, SQL Server, Postgres, etc**. Other non-relational databases (also called NoSQL) databases like MongoDB, DynamoDB, etc. do not use SQL

Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax. So, we may encounter queries that work in SQL Server but do not work in MySQL.

1. **What are the categories or different types of SQL commands?**

**DDL- Data Definition Language**

**DML- Data Manipulation Language**

**DCL- Data Control Language:** Grant, Revoke

**TCL- Transaction Control Language:** Commit, Rollback

**DRL- Data Read Language:** Select- use to read data

1. **Name the commands in the DDL category?**

Create table

Alter Table

Drop Table

1. **Name the commands in the DML category?**

Insert

Delete

Update

Merge combination of update/Delete/Insert

1. **Name the commands in the DQL category?**

**DQL statements a**re used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

Example of DQL:

SELECT – is used to retrieve data from the database.

1. **Name the commands in the DCL category?**

DCL(Data Control Language): DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions and other controls of the database system.

Examples of DCL commands:

GRANT-gives users access privileges to the database.

REVOKE-withdraw user’s access privileges given by using the GRANT command

1. **Name the commands in the TCL category?**

TCL(transaction Control Language): TCL commands deal with the transaction within the database.

Examples of TCL commands:

**COMMIT**– commits a Transaction.

**ROLLBACK**– rollbacks a transaction in case of any error occurs.

**SAVEPOINT**–sets a savepoint within a transaction.

**SET TRANSACTION**–specific characteristics for the transaction.

1. **Explain different storage engines in MySQL?**

**InnoDB** – Default Engine from Version 5.1

**MyISAM**- Default Engine prior to version 5.1

**CSV**

**Archive**

**Memory**

**InnoDB**

Supports Transaction

Fully ACID Compliant

Foreign Key

Log file and Data File

Clustered Indexes

Can perform Insert,update, delete etc.

Default Engine

**MyISAM- create table t\_myisam (id int) engine=myisam;**

Doesn’t support Transaction

Doesn’t support Foreign Key

No Log file

Has Data File

Can perform Insert,update, delete etc.

No Clustered Indexes

Not Default Engine

**CSV- create table t\_csv (id int not null,name varchar(100) not null) engine=csv;**

Doesn’t support Transaction

Doesn’t support Foreign Key

No Log file

Has Data File with CSV format

All columns should be defined as not null columns for CSV engine

Can perform Insert,update, delete etc.

No Clustered Indexes

Not Default Engine

**archive- create table t\_archive(id int,name varchar(100)) engine=archive;**

Doesn’t support Transaction

Doesn’t support Foreign Key

No Log file

Has Data File

Can only perform Insert and select

Delete,update, truncate not allowed

No Clustered Indexes

Not Default Engine

**Memory- create table t\_memory(id int,name varchar(100)) engine=memory;**

Doesn’t support Transaction

Doesn’t support Foreign Key

No Log file

No Data File which means data is removed when ever the serve restarts

No Clustered Indexes

Can perform Insert,update, delete etc.

Not Default Engine

1. **Difference between primary and secondary key?**

|  |  |  |
| --- | --- | --- |
| **BASIS OF COMPARISON** | **PRIMARY KEY** | **SECONDARY KEY** |
| **Description** | The attribute that uniquely identifies a row or record in a relation is known as Primary key. | A field or combination of fields that is the basis for retrieval is known as a secondary key (mainly used for finding details from large data). |
| **Use** | It uniquely identifies a record in the relational database table. | It is used for identification of rows but not usually unique. |
| **NULL Values** | It does not allow NULL values to be deleted from the parent table. | Allows NULL values. |
| **Number Of Keys** | Only one primary key is allowed in a table. | We can have multiple secondary key per table. |
| **Examples** | Examples of primary keys include: Unique last name, social security number, Online username | Examples of secondary keys include: Street address number, Phone number, Middle name etc |
| **Deletion** | Cannot be deleted from the parent table. | Can be deleted from the parent table. |

1. **What are different types of keys?**

There are broadly seven types of keys in DBMS:

* Primary Key
* Candidate Key
* Super Key
* Foreign Key
* Composite Key
* Alternate Key
* Unique Key

**1. Primary Key**

A primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely. There can be only one primary Key in a table. Also, the primary Key cannot have the same values repeating for any row. Every value of the primary key has to be different with no repetitions.

The PRIMARY KEY (PK) constraint put on a column or set of columns will not allow them to have any null values or any duplicates. One table can have only one primary key constraint. Any value in the primary key cannot be changed by any foreign keys (explained below) which refer to it.

**2. Super Key**

Super Key is the set of all the keys which help to identify rows in a table uniquely. This means that all those columns of a table than capable of identifying the other columns of that table uniquely will all be considered super keys.

Super Key is the superset of a candidate key (explained below). The Primary Key of a table is picked from the super key set to be made the table’s identity attribute.

**3. Candidate Key**

Candidate keys are those attributes that uniquely identify rows of a table. The Primary Key of a table is selected from one of the candidate keys. So, candidate keys have the same properties as the primary keys explained above. There can be more than one candidate keys in a table.

**4. Alternate Key**

As stated above, a table can have multiple choices for a primary key; however, it can choose only one. So, all the keys which did not become the primary Key are called alternate keys.

**5. Foreign Key**

Foreign Key is used to establish relationships between two tables. A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table. Foreign keys help to maintain data and referential integrity.

**6. Composite Key**

Composite Key is a set of two or more attributes that help identify each tuple in a table uniquely. The attributes in the set may not be unique when considered separately. However, when taken all together, they will ensure uniqueness.

**7. Unique Key**

Unique Key is a column or set of columns that uniquely identify each record in a table. All values will have to be unique in this Key. A unique Key differs from a primary key because it can have only one null value, whereas a primary Key cannot have any null values.

1. **When does implicit commit happen?**

SET autocommit = 1 causes an implicit commit

Some commands like CREATE TABLE ... SELECT, cause a commit immediately after execution. This means that, even if the statement fails with an error, the transaction is committed.. Such statements couldn't be rollbacked in any case.

1. **When does implicit rollback happen?**

It is a system crash.

1. **COMMIT, ROLLBACK, SAVEPOINT**

**COMMIT command**

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

**Following is commit command's syntax,**

COMMIT;

**ROLLBACK command**

This command restores the database to the last committed state. It is also used with the SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command.

**Following is rollback command's syntax,**

**ROLLBACK TO savepoint\_name;**

**SAVEPOINT command**

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

**Following is savepoint command's syntax,**

**SAVEPOINT savepoint\_name;**

1. **Explain the ORDER BY clause.**

* Order by Clause sorts the output in either descending or ascending order.
* Order by is always the last clause in the query. Only limit clause can come after order by clause
* The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
* SELECT column1 FROM table\_name ORDER BY column1 ASC;

1. **What is the difference between group or aggregate functions and single row functions**

**Single Row functions -** Single row functions are the one who work on a single row and return one output per row. For example, length and case conversion functions are single row functions.

Eg. SELECT UPPER (first\_name), INITCAP (last\_name), LOWER (job\_id) FROM employees

**Multiple Row functions -** Multiple row functions work upon a group of rows and return one result for the complete set of rows. They are also known as Group Functions. eg max(), min() etc

1. **Explain the truncate command**

TRUNCATE TABLE removes all rows from a table, but the table structure and its columns, constraints, indexes, and so on remain.

Syntax: TRUNCATE TABLE <table\_name>;

1. **What is the difference between delete and truncate commands**

|  |  |
| --- | --- |
| **TRUNCATE** | **DELETE** |
| Truncate also delete data from the table but truncate doesn’t have any where clause which means truncate will remove all the records | Delete can delete specific records using where clause |
| Truncate cannot be rollback | Delete can be rollback |
| Truncate command doesn’t get logged.  Logging for truncate and other DDL commands happened only at the statement level | Delete is a logged command  Row level logging happens for delete |
| Truncate is faster in performance | Delete is slower in performance than truncate |
| Truncate resets the auto\_increment value to initial value | Delete doesn’t reset the auto\_increment value |
| Truncate cannot have a trigger | Delete can have a trigger |

1. **Explain what happens when group functions are used without the group by clause**

When we use GROUP BY clause in the SELECT statement without using aggregate functions then it would behave like a DISTINCT clause.

1. **What is the having clause**

Having Clause is used to apply filters on aggregate columns. Having clause can only be used if you have group by clause

1. **What is the difference between where and having clauses?**

* Where clause is used to apply filters on non-aggregate columns (table columns) and having is used to apply filters on aggregate columns (sum, max, min, avg etc.)
* Where clause can be used without group by clause but having clause can be used only with group by clause
* On Clause is used to specify the joining condition when you use inner or outer join in the ansi syntax of joins

1. **What is a column level constraint?**

Column-level constraints refer to a single column in the table and do not specify a column name (except check constraints). They refer to the column that they follow.

Column constraints include:

**NOT NULL**

**PRIMARY KEY**

**UNIQUE**

**FOREIGN KEY**

**CHECK:** Specifies rules for values in the column

1. **How many primary key constraints can we have on a table**

We can have only one primary key constraint.

1. **What is the difference between unique key and primary key constraints?**

**Unique key**: - can have only unique values (no duplicates) but can have NULL values. One table can have more than one unique key.

**Primary key**: - Values must be unique and NOT NULL. Only one primary key is allowed per table

1. **Differentiate between CHAR and VARCHAR2 datatypes**

**CHAR-**  is used to store character strings of fixed length specified. If the length of the string is less than set or fixed length then it is padded with extra blank spaces so that its length becomes equal to the set length.

**VARCHAR**- is used to store character strings of variable length but maximum of set length specified. If the length of the string is less than set or fixed length then it will store as it is without padding with extra blank spaces.

1. **Why do we need joins?**

Joins are needed for us to use the contents of other tables by using the relation between the tables. Join clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

1. **What are the different types of joins? Explain each**

**NON EQUI JOIN**

**CROSS JOIN**

**(INNER) JOIN**: Returns records that have matching values in both tables

**LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table

**RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table

**FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

1. **What are Subqueries**

A subquery in MySQL is a query, which is nested into another SQL query and embedded with **SELECT, INSERT, UPDATE or DELETE** statement along with the various operators. We can also nest the subquery with another subquery. A subquery is known as the inner query, and the query that contains the subquery is known as the outer query.

1. **What is an inline view?**

When you write a query instead of a table name in the from clause

An inline view is **a SELECT statement in the FROM-clause of another SELECT statement to create a temporary table that could be referenced by the SELECT statement**. Inline views are utilized for writing complex SQL queries without join and subqueries operations.

1. **What are Multi row comparison operators name them**

Multiple row subquery returns one or more rows to the outer SQL statement. You may use the IN, ANY, or ALL operator in outer query to handle a subquery that returns multiple rows**.**

1. **What are views why do we use views**

Views- a logical object, a saved query, a virtual table

It doesn’t store any data in it

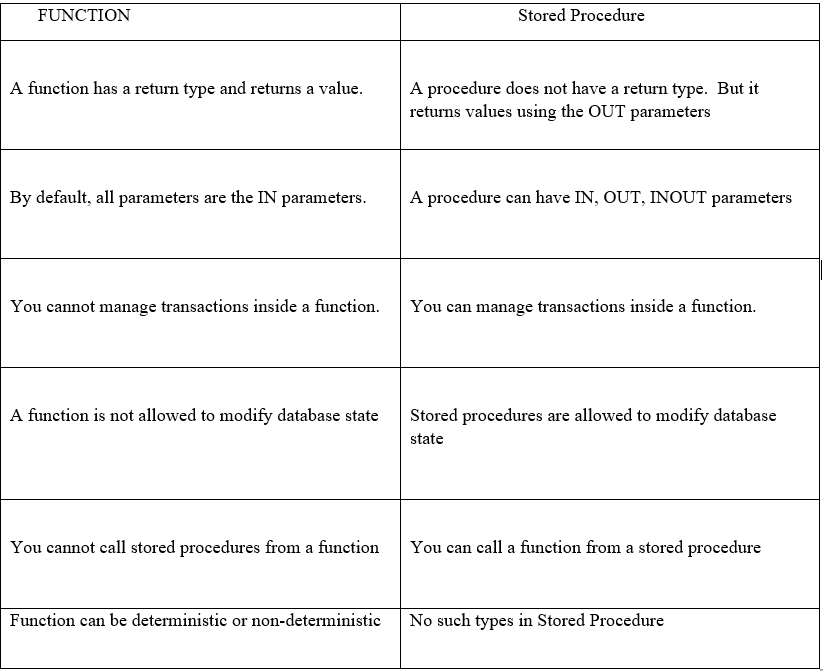
Views doesn’t occupy space for data

**Advantages:**

Security- Hide specific columns or rows

Reusability- Once a view is created you can reuse it

1. **Function versus Stored Procedure**



1. **Difference between simple and complex views**

|  |  |
| --- | --- |
| Simple View | Complex View |
| Contains only one single base table or is created from only one table. | Contains more than one base tables or is created from more than one tables. |
| We cannot use group functions like MAX(), COUNT(), etc. | We can use group functions. |
| Does not contain groups of data. | It can contain groups of data. |
| DML operations could be performed through a simple view. | DML operations could not always be performed through a complex view. |
| INSERT, DELETE and UPDATE are directly possible on a simple view. | We cannot apply INSERT, DELETE and UPDATE on complex view directly. |

1. **What happens to a view when one or more table on which the view is based are dropped?**

We can see the view but we cannot perform DML commands on it.

1. **When indexes are implicitly created they are of what type unique or non unique**

Unique

1. **What is a default value for a column of a table**

Default constraint is used to set a default value for a column.

1. **When do default values set for a column of table come into the picture**

The default value will be added to all new records, if no other value is specified.

1. **What is the data type of the value obtained when you subtract a date from another date value**
2. **Does group by clause ignore null values**

Yes

1. **Do group or aggregate functions ignore null values**

Yes. aggregate functions and group functions ignore null values.

1. **Which is the special operator used for performing a comparison with null values**

SQL Null check is performed using either IS NULL or IS NOT NULL to check whether a value in a field is NULL or not. When a field value is NULL it means that the database assigned nothing in that field for that row. The NULL is not zero or blank. It represents an unknown or inapplicable value. It can’t be compared using AND / OR logical operators. The special operator ‘IS’ is used with the keyword ‘NULL’ to locate ‘NULL’ values. NULL can be assigned in both type of fields i.e. numeric or character type of field.

1. **What is the order of precedence between OR , not and and logical operators (All precedence)**

|  |  |
| --- | --- |
| Operator | Operation |
| \*\* | exponentiation |
| +, - | identity, negation |
| \*, / | multiplication, division |
| +, -, || | addition, subtraction, concatenation comparison |
| NOT | logical negation |
| AND | conjunction |
| OR | inclusion |

1. **What is a cursor**

**Cursor** is a pointer to a set of records and is used to fetch data row by row.

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML operations on Table by User. Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors

1. **What is IMPLICIT CURSOR**

**Implicit Cursors:**

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

1. **What is EXPLICIT CURSOR**

**Explicit Cursors:**

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for fetching data from Table in Row-By-Row Manner.

1. **What are the four cursor attributes**

Each cursor has a set of attributes that enables an application program to test the state of the cursor. These attributes are **%ISOPEN, %FOUND, %NOTFOUND, and %ROWCOUNT.** This attribute is used to determine whether a cursor is in the open state

1. **Name and explain a few single-row character functions**

**Character functions - Accepts character input and returns number or character value. Functions under the category are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE**.

1. **CONCAT** function concatenates two string values.
2. **LENGTH** function returns the length of the input string.
3. **SUBSTR** function returns a portion of a string from a given start point to an end point.
4. **The INSTR** function returns the numeric position of a character or a string in a given string.
5. **LPAD** and **RPAD** functions pad the given string up to a specific length with a given character.
6. **TRIM** function trims the string input from the start or end.
7. **REPLACE** function replaces characters from the input string with a given characters
8. **Name and explain a few single row numeric functions.**

**Number functions** - Accepts numeric input and returns numeric values.

Functions under the category are **ROUND, TRUNC, and MOD**. **ROUND and TRUNC** functions are used to round and truncate the number value. **MOD** is used to return the remainder of the division operation between two numbers.

1. **Name and explain a few single row data functions.**

**Date functions** - Date arithmetic operations return date or numeric values.

1. Functions under the category are **MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY, LAST\_DAY, ROUND and TRUNC.**
2. **MONTHS\_BETWEEN** function returns the count of months between the two dates.
3. **ADD\_MONTHS** function add 'n' number of months to an input date.
4. **NEXT\_DAY** function returns the next day of the date specified.
5. **LAST\_DAY** function returns last day of the month of the input date.
6. **ROUND** and **TRUNC** functions are used to round and truncates the date value.

**THE GREAT RAAAAAZ QUESTIONS lol**

**1.** **What is Data**

facts and statistics collected together for reference or analysis.

**2.** **Advantages/Features of RDBMS**

* Supports Relationship
* All the data is stored in the form of tables (set of rows and columns)
* Data can be access in RDBMS using one standard language called as SQL (Structured Query Language)
* Secured- only users with permissions can access your data
* Supports Transaction

**3.** **What is a Table/ entity?**

Tables/ entity are database objects that contain all the data in a database. In tables, data is logically organized in a row-and-column format similar to a spreadsheet.

**4.** **What are attributes**

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

• **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

• **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.

• **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved

directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.

• **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.

• **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a auto inc person can have more than one phone number, email\_address, etc.

These attribute types can come together in a way like −

* **Simple single-valued attributes**
* **Simple multi-valued attributes**
* **Composite single-valued attributes**
* **Composite multi-valued attributes**

**5.** **Different types of SET**

* **Union**
* **Union all**
* **Intersect** Not supported in MySQL
* **Minus/Except** Not Supported in MySQL

**6.** **Prerequisites of using SET operators**

Both the queries should have same number of columns

Corresponding data types of the columns in both the queries should be same/compatible

**7.** **What are auto increment columns and its prerequisites**

Auto increment columns take value automatically through a sequence

Autoincrement column as to be defined as a primary key

create table account (accountno int primary key auto\_increment , accname varchar(100));

alter table account auto\_increment=1001;

**8.** **Different types of Ranking Functions and its parameters**

Different types of Ranking Functions-

* **Row\_Number**
* **Rank**
* **Dense\_Rank**

They have two types of parameters that they can take

1. **Partition by Clause** – This is an optional parameter
2. **Order by Clause** – This is a mandatory parameter

**Ranking Functions are used to assign rank to the rows based on some condition.**

Ranking Functions are used to assign rank to the rows based on some condition.

They can be used only in 2 places-

1. Select column clause
2. Order by clause

**9.** **What is Transaction and its properties**

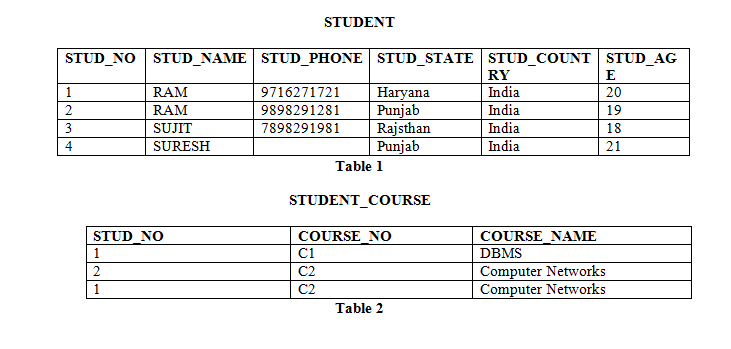
**Properties of Transaction**

* A- **Atomicity**- A transaction is either fully committed or fully rollback. Transaction should be treated as an atomic unit
* C- **Consistency**- Database should always remain in a consistent state after any transaction irrespective of whether transaction is committed or rollback or not completed
* RDBMS writes the changes first to the log file before change the changing the data in the buffer pool or data file and this process is called as Write Ahead Logging
* Consistency is achieved by instance recovery done during the startup of the instance
* RDBMS analyses the log file to identify all the transactions that were not committed but yet written to the disk and also the transactions that were committed but were not written to the disk
* Redo all the transactions that were committed but not written to the disk
* Undo or rollback all the transactions that were not committed but yet written to the disk
* I- **Isolation**- No two users can update the same data at the same time RDBMS use locks to implement isolation
* MVCC- Multi Version concurrency control- It means that users can read the data even if the same data is getting modified by some other session/user. In MVCC RDBMS takes a snapshot (row versioning) of the last committed data.
* D- **Durability**- Once the data is stored in the RDBMS it should remain forever even if the server is restarted unless the user deletes the data. Durability is implemented by storing the data on a non-volatile storage

**10.** **Explain DML anomalies**

Anomalies

There are different types of anomalies which can occur in referencing and referenced relation which can be discussed as:



**Insertion anomaly**: If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation. For Example, If we try to insert a record in STUDENT\_COURSE with STUD\_NO =7, it will not allow.

**Deletion and Updation anomaly**: If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation. For Example, If we try to delete a record from STUDENT with STUD\_NO =1, it will not allow. To avoid this, following can be used in query

**ON DELETE/UPDATE SET NULL**: If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and set the value of referencing attribute to NULL.

**ON DELETE/UPDATE CASCADE**: If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and referencing relation as well.

**11.** **What is Network Model**

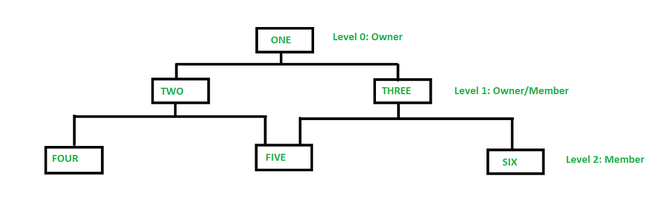
Network Model :

This model was formalized by the Database Task group in the 1960s. This model is the generalization of the hierarchical model. This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level. Mostly, there exists a many-to-many logical association between any of the two segments. We called graphs the logical associations between the segments. Therefore, this model replaces the hierarchical tree with a graph-like structure, and with that, there can more general connections among different nodes. It can have M: N relations i.e, many-to-many which allows a record to have more than one parent segment.

Here, a relationship is called a set, and each set is made up of at least 2 types of record which are given below:

An owner record that is the same as of parent in the hierarchical model.

A member record that is the same as a child in the hierarchical model.



In the above figure, member TWO has only one owner ‘ONE’ whereas member FIVE has two owners i.e, TWO and THREE. Here, each link between the two record types represents 1 : M relationship between them. This model consists of both lateral and top-down connections between the nodes. Therefore, it allows 1: 1, 1 : M, M : N relationships among the given entities which helps in avoiding data redundancy problems as it supports multiple paths to the same record. There are various examples such as TOTAL by Cincom Systems Inc., EDMS by Xerox Corp., etc.

**12.** **What is Hierarchical Model**

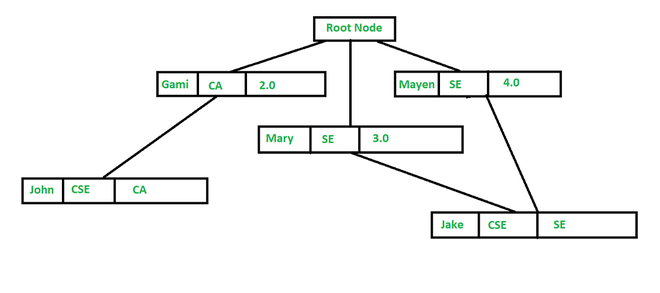
**Hierarchical Model :**

This is one of the oldest models in a data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. In this, the data is organized into a tree-like structure where each record consists of one parent record and many children. Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.

In the hierarchical model, segments pointed to by the logical association are called the child segment and the other segment is called the parent segment. If there is a segment without a parent then that will be called the root and the segment which has no children are called the leaves. The main disadvantage of the hierarchical model is that it can have one-to-one and one-to-many relationships between the nodes.

**Applications of hierarchical model :**

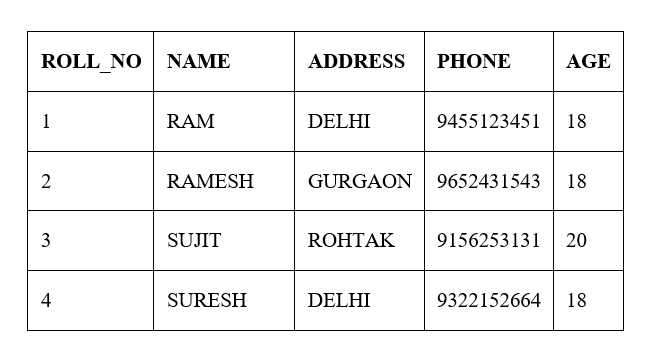
Hierarchical models are generally used as semantic models in practice as many real-world occurrences of events are hierarchical in nature like biological structures, political, or social structures.

Hierarchical models are also commonly used as physical models because of the inherent hierarchical structure of the disk storage system like tracks, cylinders, etc. There are various examples such as Information Management System (IMS) by IBM, NOMAD by NCSS, etc.

**13.** **What is relational model**

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL\_NO, NAME, ADDRESS, PHONE and AGE shown in Table 1.

**STUDENT**

****

**Attribute:** Attributes are the properties that define a relation. e.g.; ROLL\_NO, NAME

**Relation Schema:** A relation schema represents the name of the relation with its attributes. e.g.; STUDENT (ROLL\_NO, NAME, ADDRESS, PHONE and AGE) is relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.

**Tuple:** Each row in the relation is known as tuple.

**Relation Instance**: The set of tuples of a relation at a particular instance of time is called a relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is insertion, deletion or updation in the database.

**14.** **What is view**

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition

**15.** **Advantages of using views**

A good database should contain views due to the given reasons:

**Restricting data access –**

Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.

**Hiding data complexity –**

A view can hide the complexity that exists in a multiple table join.

**Simplify commands for the user –**

Views allows the user to select information from multiple tables without requiring the users to actually know how to perform a join.

**Store complex queries –**

Views can be used to store complex queries.

**Rename Columns –**

Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in select statement. Thus, renaming helps to to hide the names of the columns of the base tables.

**Multiple view facility –**

Different views can be created on the same table for different users.

**16.** **Can we perform DML on Views?**

DMLs on view are allowed with some restrictions

When DMLs are performed on views the changes happen on the base table

DMLs cannot modify multiple tables through view

**17.** **Restrictions of performing DML on views with example**

DML operations could be performed through a simple view. DML operations could not always be performed through a complex view. INSERT, DELETE and UPDATE are directly possible on a simple view. We cannot apply INSERT, DELETE and UPDATE on complex views directly.

**18.** **What is self-join?**

**SELF JOIN**: As the name signifies, in SELF JOIN a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.

**Syntax:**

SELECT a.column1 , b.column2

FROM table\_name a, table\_name b

WHERE some\_condition;

table\_name: Name of the table.

some\_condition: Condition for selecting the rows.

Example Queries(SELF JOIN):

SELECT a.ROLL\_NO , b.NAME

FROM Student a, Student b

WHERE a.ROLL\_NO < b.ROLL\_NO;

**19.** **What is a correlated subquery?**

**Subqueries in which we define a relation of a column from the outer query with the column of the inner query. A correlated subquery is evaluated once for each row processed by the parent statement.**

The parent statement can be a SELECT, UPDATE, or DELETE statement.

SELECT column1, column2, ....

FROM table1 outer

WHERE column1 operator

(SELECT column1, column2

FROM table2

WHERE expr1 =

outer.expr2);

**20.** **Disadvantage of using correlated subquery**

Correlated subqueries are generally very slow in performance because the correlated sub query gets executed as many times as you have the number of rows in the outer query.

**21.** **What are different isolation levels**

Read Uncommitted

Read Committed

Repeatable Reads

Serializable

**22.** **Which isolation level is most strict or most lenient?**

****

**23.** **What is procedure**

Set of code/program which can take input through some parameters and can also return values throughout parameters. You cannot call procedures inside queries

**24.** **What is function**

Set of code/program which can take input through some parameters but has to return a value through a return clause. You can call functions inside queries

**25.** **What is deterministic functions**

**Deterministic**- if the function is expected to return same output for the same input value each tim**e**

**26.** **Steps of using a cursor**

**Steps to use cursor**

* Declare cursor
* Declare not found handler for cursor
* Open Cursor
* Fetch data from cursor in a loop
* Close cursor

**27.** **Difference between procedure and functions**

|  |  |  |
| --- | --- | --- |
|  | Function | Procedure |
| 1. | A function deals with as an expression. | Whereas a procedure does not deal with as an expression. |
| 2. | Function is used to calculate something from a given input. Hence it got its name from Mathematics. | While procedure is the set of commands, which are executed in a order. |
| 3. | The function can be called by a procedure. | But a procedure can not be called by a function. |
| 4. | In SQL, inside the function we cannot use the DML (Data manipulation language) commands such as Insert, Delete, Update. | Here, in SQL, inside the procedure we can use DML commands. |
| 5. | Functions can be called through SQL queries. | However, the procedure can’t be called through a SQL query. |
| 6. | Each time functions are compiled when they are called. | Whereas, procedures are compiled only once and can be called again and again as needed without being compiled each time. |
| 7. | The return statement of a function returns the control and function’s result value to the calling program. | While the return statement of the procedure returns control to the calling program, it cannot return the result value. |
| 8. | Function doesn’t support try-catch blocks. | While it supports try-catch blocks. |
| 9. | Function can be operated in the SELECT statement. | While it can’t be operated in the SELECT statement. |
| 10. | Function does not support explicit transaction handles. | While procedure supports explicit transaction handles. |

**28.** **What are triggers**

Trigger is a set of code that gets executed whenever a specific event occurs. Trigger is like a hidden code

**29.** **What are different types of triggers?**

**Events**

**DML Commands**

Insert, update , delete

**Logon Triggers**

**System Commands**

Startup, Shutdown

**DDL Commands**

Create , Drop, Alter

**30.** **What is OLTP vs OLAP**

|  |  |  |
| --- | --- | --- |
| Parameters | OLTP | OLAP |
| Process | It is an online transactional system. It manages database modification. | OLAP is an online analysis and data retrieving process. |
| Characteristic | It is characterized by large numbers of short online transactions. | It is characterized by a large volume of data. |
| Functionality | OLTP is an online database modifying system. | OLAP is an online database query management system. |
| Method | OLTP uses traditional DBMS. | OLAP uses the data warehouse. |
| Query | Insert, Update, and Delete information from the database. | Mostly select operations |
| Table | Tables in OLTP databases are normalized. | Tables in the OLAP database are not normalized. |
| Source | OLTP and its transactions are the sources of data. | Different OLTP databases become the source of data for OLAP. |
| Data Integrity | OLTP databases must maintain data integrity constraints. | OLAP database does not get frequently modified. Hence, data integrity is not an issue. |
| Response time | It's response time is in millisecond. | Response time in seconds to minutes. |
| Data quality | The data in the OLTP database is always detailed and organized. | The data in the OLAP process might not be organized. |

**31.** **What is an Index**

Indexes- are objects which are used to optimize search

**32.** **Indexes**

| **CLUSTERED INDEX** | **NON-CLUSTERED INDEX** |
| --- | --- |
| Clustered index is faster. | Non-clustered index is slower. |
| Clustered index requires less memory for operations. | Non-Clustered index requires more memory for operations. |
| In clustered index, index is the main data. | In Non-Clustered index, index is the copy of data. |
| A table can have only one clustered index. | A table can have multiple non-clustered indexes. |
| Clustered index has inherent ability of storing data on the disk. | Non-Clustered index does not have inherent ability of storing data on the disk. |
| Clustered index store pointers to block not data. | Non-Clustered index store both value and a pointer to actual row that holds data. |
| In Clustered index leaf nodes are actual data itself. | In Non-Clustered index leaf nodes are not the actual data itself rather they only contain included columns. |
| In Clustered index, Clustered key defines order of data within table. | In Non-Clustered index, index key defines order of data within index. |
| A Clustered index is a type of index in which table records are physically reordered to match the index. | A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk. |

**Clustered Index-** Entire row of the table is stored in the leaf level of the Index. Since the entire row is stored in the leaf you cannot create more than 1 clustered index on a table. **In MySQL, for INNODB tables when you create a primary key a unique clustered index is created on the table automatically**. **When a clustered index is created then internal structure of the table is dropped**

**Non-Clustered Index-** only the ROWID is stored along with the index column in the leaf level. Tables can have multiple non-clustered indexes. Whenever you create a unique key in the table a unique non-clustered index is added automatically

**33.** **Star Schema vs Snowflake schema**

|  |  |
| --- | --- |
| Star Schema | Snowflake Schema |
| Hierarchies for the dimensions are stored in the dimensional table. | Hierarchies are divided into separate tables. |
| It contains a fact table surrounded by dimension tables. | One fact table surrounded by dimension table which are in turn surrounded by dimension table |
| In a star schema, only a single join creates the relationship between the fact table and any dimension tables. | A snowflake schema requires many joins to fetch the data. |
| Simple DB Design. | Very Complex DB Design. |
| Denormalized Data structure and query also run faster. | Normalized Data Structure. |
| High level of Data redundancy | Very low-level data redundancy |
| The Single Dimension table contains aggregated data. | Data Split into different Dimension Tables. |
| Cube processing is faster. | Cube processing might be slow because of the complex join. |
| Offers higher performing queries using Star Join Query Optimization. Tables may be connected with multiple dimensions. | The Snowflake schema is represented by a centralized fact table which is unlikely to be connected with multiple dimensions. |

**34. Nth Highest Salary**

+------+--------+

|  |
| --- |
| | id salary |
| | 1 | 10000 |  | 2 | 20000 |  | 3 | 20000 |  | 4 | 30000 |  | 5 | 40000 |  | 6 | 50000 |  +------+--------+ |

6 rows in set (0.00 sec)

**select id,salary from employee e1 where N-1=(select count(distinct salary) from employee e2 where e2.salary>e1.salary);**

**Queries**

1. **Write an SQL query to get the third maximum salary of an employee from a table named employee table.**

select id,salary from employee e1 where N-1=(select count(distinct salary) from employee e2 where e2.salary>e1.salary); //N=3

1. **Write an SQL query to find names of employees starting with "A’'?**

Select \* from Employee where ename like ‘A%’;

1. **How can you create an empty table from an existing table?**

CREATE TABLE new\_table AS (SELECT \* FROM old\_table WHERE 1=2);

1. **How to fetch common records from two tables?**

Let us consider we are working on two tables students and teacher and we want to retrieve common records between the two ( let Subject\_code ) then the query will be as follows -

SELECT Subject\_code

FROM students

INTERSECT

SELECT Subject\_code

FROM teacher ;

### INTERSECT Operator using IN and Subquery

The following syntax uses the IN and Subquery clause for returning the distinct rows from both tables:

1. mysql> **SELECT** **DISTINCT** column\_list **FROM** table\_name1
2. **WHERE** column\_name IN (**SELECT** column\_list **FROM** table\_name2);
3. **How to fetch alternate records from a table?**

To fetch even Numbered row:

SELECT \* FROM table\_name WHERE column\_name % 2 = 0

To fetch odd Numbered row:

SELECT \* FROM table\_name WHERE column\_name % 2 = 1

1. **What is the command used to fetch the first 5 characters of the string?**

MySQL SUBSTRING() returns a specified number of characters from a particular position of a given string.

**Syntax:**

SUBSTRING(str, pos, len)

OR

SUBSTRING(str FROM pos FOR len)

1. **Which operator is used in a query for pattern matching?**

**>>> %**