**MYSQL**

**Q1. What is Database?**

* A database is an application that stores the organized collection of records.
* It can be accessed and manage by the user very easily.
* It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly.
* Each database contains distinct [API](https://www.javatpoint.com/api-full-form) for performing database operations such as creating, managing, accessing, and searching the data it stores.
* Today, many databases available like MySQL, Sybase, [Oracle](https://www.javatpoint.com/what-is-oracle), [MongoDB](https://www.javatpoint.com/mongodb-tutorial), [PostgreSQL](https://www.javatpoint.com/postgresql-tutorial), [SQL Server](https://www.javatpoint.com/sql-server-tutorial), etc. In this section, we are going to focus on MySQL mainly.

**Q2. What is MYSQL?**

* MySQL is a relational database management system
* MySQL is open-source
* MySQL is free
* MySQL is ideal for both small and large applications
* MySQL is very fast, reliable, scalable, and easy to use
* MySQL is cross-platform
* MySQL is compliant with the ANSI SQL standard
* MySQL was first released in 1995
* MySQL is developed, distributed, and supported by Oracle Corporation
* MySQL is named after co-founder Monty Widenius's daughter: My

**Q3. What is RDBMS?**

* RDBMS stands for Relational Database Management System.
* RDBMS is a program used to maintain a relational database.
* RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
* It is a program that allows us to create, delete, and update a relational database.
* A Relational Database is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns.
* It is a smaller subset of DBMS which was designed by E.F Codd in the 1970s. The major DBMSs like[SQL, My-SQL](https://www.geeksforgeeks.org/sql-vs-mysql/), and [ORACLE](https://www.geeksforgeeks.org/oracle-architecture/)are all based on the principles of relational DBMS.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Create Database**

The MySQL CREATE DATABASE statement is used to create a new database

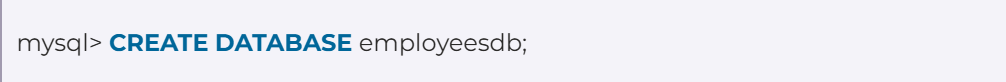
A close-up of a computer screen

Description automatically generated

The parameter descriptions of the above syntax are as follows:

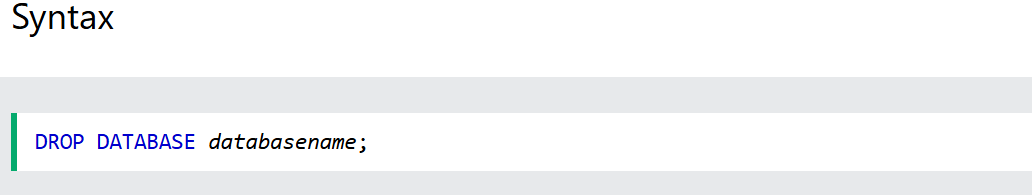
|  |  |
| --- | --- |
| **Parameter** | **Description** |
| database\_name | It is the name of a new database that should be unique in the MySQL server instance. The **IF NOT EXIST** clause avoids an error when we create a database that already exists. |
| charset\_name | It is optional. It is the name of the character set to store every character in a string. MySQL database server supports many character sets. If we do not provide this in the statement, MySQL takes the default character set. |
| collation\_name | It is optional that compares characters in a particular character set. |

**Example**

****

**MySQL DROP Database**

It will delete the database along with all the tables, indexes, and constraints permanently.



**Note:**Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

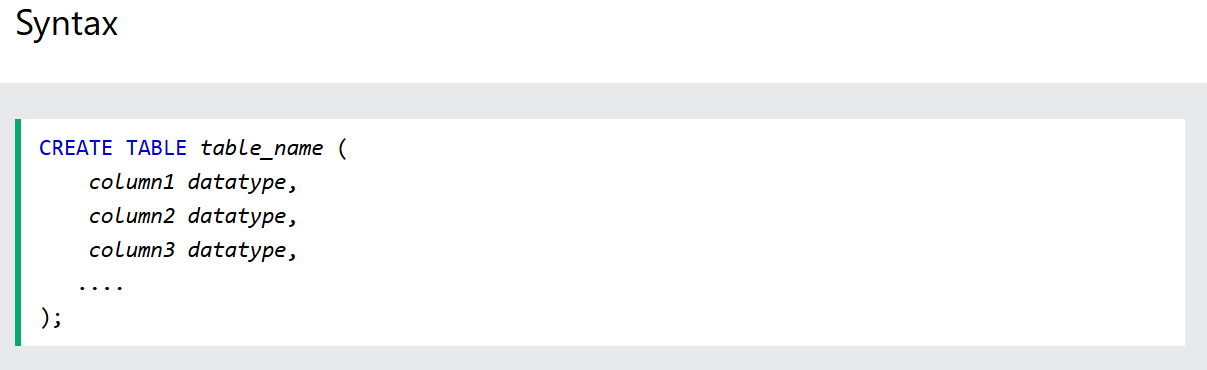
**A white rectangular object with a white stripe

Description automatically generated**

**MySQL CREATE TABLE**

The CREATE TABLE statement is used to create a new table in a database.

* Name of the table
* Names of fields
* Definitions for each field



A white rectangular object with a black border

Description automatically generated

**Constrants in MYSQL**

In MySQL, constraints are rules that limit the types of data that can be inserted into a table. They help maintain the integrity and accuracy of the data.

Here are the main types of constraints you can use in MySQL:

A screenshot of a computer

Description automatically generated

**1. NOT NULL**

* Ensures that a column cannot have a NULL value.



A screenshot of a computer

Description automatically generated

A screenshot of a computer

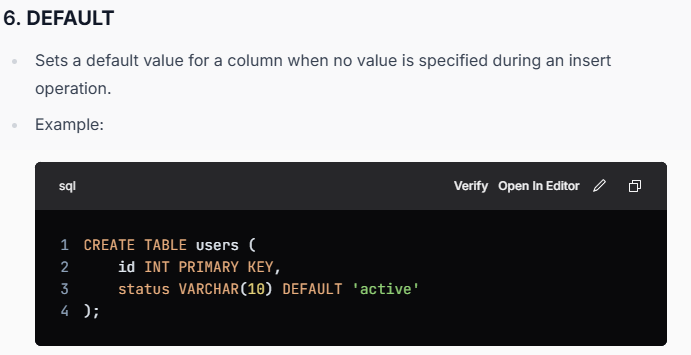
Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated



A screenshot of a computer program

Description automatically generated

**Interview Question**

**1. What is MySQL and How does it differ from other relational databases?**

MySQL is an open-source relational database management system (RDBMS) that is widely used for managing structured data.

It utilizes SQL (Structured Query Language) for querying and managing data. MySQL is known for its reliability, scalability, and performance, making it a popular choice for various applications

**2. How to create a database in MySQL?**

To create a database in MySQL, we can use the CREATE DATABASE statement followed by the name we want to give to our database.

For example:

CREATE DATABASE mydatabase;

**3. Difference between CHAR and VARCHAR data types.**

The **CHAR** and **VARCHAR** data types are both used to store character strings in a database, but they have some key differences in how they store data and their intended use cases. Here are the main differences:

* **CHAR:**
  + Fixed-length data type. When you define a CHAR(n), it always allocates space for n characters. If the string stored is shorter than n, the remaining space is padded with spaces.
  + CHAR datatype can hold a maximum of 255 characters.
  + CHAR is faster.
* **VARCHAR:**
  + Variable-length data type. When you define a VARCHAR(n), it can store up to n characters, but it only uses as much space as needed for the actual string plus a small amount of overhead to store the length of the string.
  + VARCHAR can store up to 4000 characters.
  + VARCHAR is slow as compare to CHAR.

**4. Explain the differences between SQL and MySQL?**

| **SQL** | **MySQL** |
| --- | --- |
| It is a structured query language that manages the relational database management system. | It is a relational database management system that uses SQL. |
| It is not an open-source language. | MySQL is an open-source platform. It allows access to anyone. |
| Used to perform tasks such as querying, updating, and managing data in databases. | Used to create, manage, and maintain databases, and to execute SQL queries. |
| Language (specifically a query language) | Software (database management system) |
| SQL supports [XML](https://www.geeksforgeeks.org/xml-basics) and user defined functions. | It doesn’t support XML and any user defined functions |

**5. What is the MySQL server’s default port?**

3306 is [MySQL server](https://www.geeksforgeeks.org/how-to-stop-mysql-server-on-windows-and-linux)‘s default port.

**6. How many different tables are present in MySQL?**

**There are 5 types of tables present in MySQL.**

* [**Heap**](https://www.geeksforgeeks.org/heap-sort)**table :** Also known as MEMORY tables, HEAP tables store data in memory for fast access. They are temporary and are lost when the server is restarted.
* [**merge**](https://www.geeksforgeeks.org/merge-sort)**table**
* **MyISAM table**
* **INNO DB table**
* **ISAM table**

**7. What is Difference between CHAR\_LENGTH and LENGTH?**

In MySQL, both CHAR\_LENGTH and LENGTH functions are used to determine the size of a string, but they measure different aspects of the string.

Here’s a detailed comparison of the two:

**1. LENGTH() Function**

* **Definition**: The **LENGTH()** function returns the length of a string in bytes. This means that it counts the number of bytes that the string occupies in memory.



**2. CHAR\_LENGTH() Function**

* **Definition**: The **CHAR\_LENGTH()** function (or **CHARACTER\_LENGTH()**, which is synonymous) returns the length of a string in characters.
* This means it counts the number of characters in the string, regardless of how many bytes those characters occupy.

A screenshot of a computer program

Description automatically generated

**8. What is Like Operator and Wildcards?**

**The LIKE Operator**

* **Definition**: The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.

A black and white screen with white text

Description automatically generated

**Wildcards**

Wildcards are special characters that allow you to perform pattern matching with the **LIKE** operator.

There are two primary wildcards in MySQL:

1. **Percent Sign (%)**:

* Represents zero, one, or multiple characters.
* Example:
  + **LIKE 'a%'** matches any string that starts with "a" (e.g., "apple", "abc", "a123").
  + **LIKE '%a'** matches any string that ends with "a" (e.g., "banana", "cuba", "alpha").
  + **LIKE '%a%'** matches any string that contains "a" anywhere (e.g., "banana", "candy", "apple").

2. **Underscore (\_)**

* + Represents a single character.
  + Example:
    - **LIKE 'a\_'** matches any string that starts with "a" and has exactly one character following it (e.g., "ab", "ac", but not "abc").
    - **LIKE '\_a'** matches any string that has exactly one character before "a" (e.g., "ba", "ca", but not "abc").
    - **LIKE 'a\_\_'** matches any string that starts with "a" and has exactly two characters following it (e.g., "abc", "a12").

**9.What is Group by and Having Clause?**

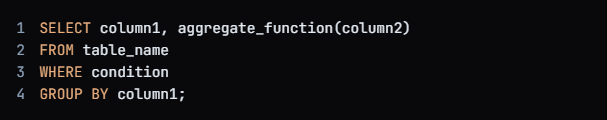
**Group BY**

The GROUP BY clause is used to group rows that share the same value in one or more columns into summary rows, like finding the total number of employees in each department or the average salary per employee for each department.

It is generally used in a SELECT statement.

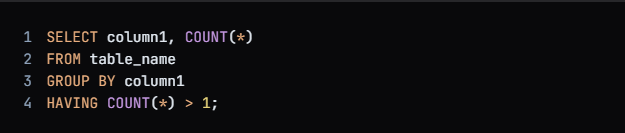
**Key Points about GROUP BY:**

1. **Aggregation**: When you use **GROUP BY**, you typically use aggregate functions like **COUNT()**, **SUM()**, **AVG()**, **MAX()**, and **MIN()** to perform calculations on each group of data.



**Having Clause**

The HAVING clause allows you to apply conditions to the groups created by the **GROUP BY**clause. It is particularly useful when you want to filter data based on aggregate values such as sums, averages, counts, etc.



**10. View**

In MySQL, a view is a virtual table that is defined by a SQL query.

It does not store data itself but provides a way to access data from one or more tables in a structured manner.

Views are useful for simplifying complex queries, restricting data access, or presenting data in a particular format.

A screenshot of a computer

Description automatically generated

**Key Points:**

* A view does not store data; it stores a query to retrieve data.
* Views can simplify complex queries.
* They can be used to present a subset of data from a table.
* Views are read-only in MySQL by default, but they can be made updatable under certain conditions (e.g., when the view references a single table and does not contain aggregate functions or GROUP BY).

A screenshot of a computer

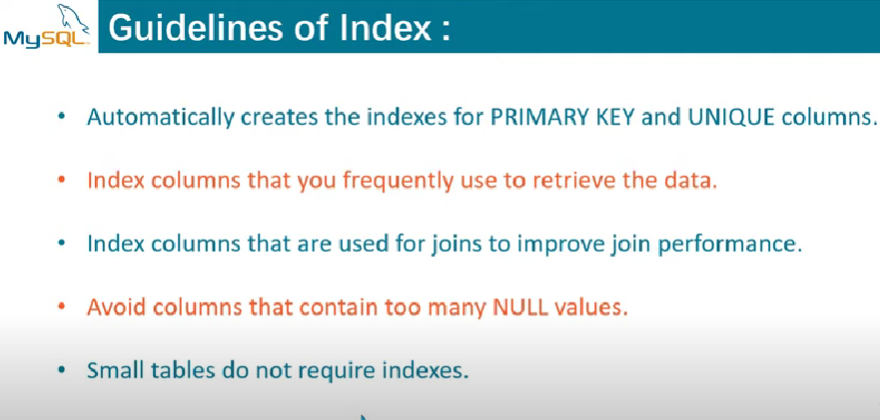
Description automatically generated

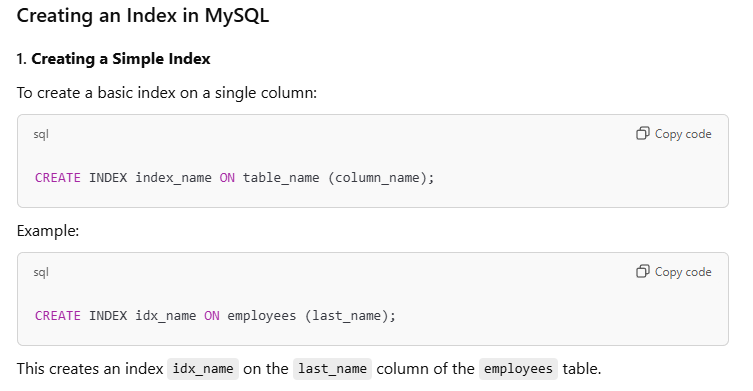
**11. Indexs**

In MySQL, an index is a data structure that improves the speed of data retrieval operations on a table.

Indexes are used to quickly look up data based on a column or a combination of columns.

Think of an index like a book's table of contents, which helps you find the right page quickly without having to read through the entire book.





A screenshot of a chat

Description automatically generated

**When to Use Indexes**

Indexes are helpful in the following situations:

* **Speed up queries**: When querying large tables, indexes can significantly speed up search and retrieval operations.
* **Optimize joins**: Indexes can speed up join operations by indexing the columns used in ON or WHERE clauses.
* **Improve sorting**: If you're frequently sorting by a specific column, creating an index on that column can improve performance.

**When to Avoid Indexes**

* **Write-heavy operations**: Every time you insert, update, or delete data in the table, MySQL must also update the associated indexes. If your table is write-heavy, having too many indexes can degrade performance.
* **Small tables**: For small tables with few records, the overhead of maintaining indexes may not be worth the performance gain.
* **Indexes on columns with high cardinality** (i.e., many unique values): In such cases, an index may not be very efficient, as the database might not benefit much from the index.

**Key Points:**

* **Indexes** improve query performance by allowing faster search, retrieval, and sorting operations.
* You can create indexes on single or multiple columns, with unique or full-text constraints.
* While indexes speed up read operations, they can slow down write operations (inserts, updates, and deletes) since they need to be updated.
* Indexes should be used thoughtfully based on the query patterns and the table's size.

**Drawbacks of Indexing All Columns**

1. **Slower INSERT, UPDATE, DELETE**:
   * Every time you modify data, the database must also update all the indexes.
   * More indexes = more overhead.
2. **Increased Storage Usage**:
   * Indexes consume disk space. Indexing every column can significantly increase storage requirements.
3. **Diminishing Returns**:
   * Not all columns benefit from indexing (e.g., columns with high cardinality like timestamps or unique IDs might not need it unless queried often).
   * Indexing low-selectivity columns (like gender or status with few distinct values) often doesn’t help much.
4. **Complex Query Planner Decisions**:
   * Too many indexes can confuse the query optimizer, leading to suboptimal execution plans.

**🧠 Best Practices**

* **Index columns used in WHERE, JOIN, ORDER BY, and GROUP BY** clauses.
* **Avoid indexing columns that are rarely queried** or frequently updated.
* **Use composite indexes** when queries involve multiple columns together.
* **Monitor performance** using tools like EXPLAIN or ANALYZE.

**12. Type of SLQ Command**

1.**DQL (Data Query Language) :** Used to retrieve data from databases. (SELECT)

2. **DDL (Data Definition Language)** : Used to create, alter, and delete database objects like tables, indexes, etc. (CREATE, DROP, ALTER, RENAME, TRUNCATE)

3**. DML (Data Manipulation Language):** Used to modify the database. (INSERT, UPDATE, DELETE)

4. **DCL (Data Control Language):** Used to grant & revoke permissions. (GRANT, REVOKE)

5. **TCL (Transaction Control Language**): Used to manage transactions. (COMMIT, ROLLBACK, START TRANSACTIONS, SAVEPOINT)

**Storage Engine**

MyISAM and InnoDB are two of the most commonly used **storage engines** in MySQL. Each has different characteristics, strengths, and use cases.

**1. MyISAM:**

MyISAM is the default storage engine in MySQL for many years.

It is primarily designed for **read-heavy** applications and **static data**.

It does not support transactions or foreign key constraints but offers fast data retrieval.

**Key Features of MyISAM:**

* **No Transactions:** MyISAM does not support transactions, which means it cannot ensure data consistency in case of system crashes. There are no BEGIN, COMMIT, or ROLLBACK statements.
* **Table-Level Locking:** MyISAM uses table-level locking, which means that when one query is writing to a table, other queries (including reads) must wait for the write to finish. This can become a bottleneck in write-heavy applications.
* **Faster for Read-Heavy Workloads:** Because MyISAM uses table-level locking, it performs better in environments where read operations dominate, and there are few or no write operations.
* **Full-Text Search Support:** MyISAM supports full-text indexing, which makes it suitable for applications like search engines.
* **Data Integrity and Reliability:** Since MyISAM doesn’t support transactions, it can’t roll back changes in the event of a crash, and it doesn’t support foreign keys, making it less reliable than InnoDB for mission-critical applications.
* **No Foreign Key Constraints:** MyISAM does not support foreign key constraints, which means there is no enforcement of referential integrity between tables.
* **Crash Recovery:** MyISAM lacks automatic crash recovery. If the MySQL server crashes, the database files could become corrupted, although MyISAM includes a repair feature (myisamchk).

**Use Cases for MyISAM:**

* **Read-heavy applications** where performance is critical, and there are fewer or no write operations.
* Applications that require **full-text search capabilities**.
* **Log analysis** and other read-only databases that don't require complex relationships.

**2. InnoDB:**

InnoDB is the default storage engine in MySQL for transactions and data integrity. It is designed for **write-heavy** and **transactional applications** where consistency and reliability are crucial.

**Key Features of InnoDB:**

* **Transactions:** InnoDB supports **ACID** transactions, which means it can guarantee data consistency and rollback any changes in case of failure, using BEGIN, COMMIT, and ROLLBACK.
  + **ACID:**
    - **A**tomicity: All operations within a transaction are completed or none at all.
    - **C**onsistency: The database transitions from one consistent state to another.
    - **I**solation: The operations of one transaction are isolated from others.
    - **D**urability: Changes from a transaction are permanent.
* **Row-Level Locking:** InnoDB uses **row-level locking**, which is more efficient than table-level locking. It allows multiple transactions to modify different rows of the same table concurrently, improving performance in **write-heavy** applications.
* **Foreign Key Constraints:** InnoDB supports **foreign key constraints**, enforcing referential integrity between tables. This ensures that relationships between tables are maintained and prevents data anomalies.
* **Automatic Crash Recovery:** InnoDB has built-in **crash recovery**. If the MySQL server crashes, InnoDB can automatically recover the database to its last consistent state using the transaction log.
* **Support for Large Databases:** InnoDB is designed to handle large databases and supports large table sizes (up to 64 TB) and large numbers of rows.
* **Better for Write-Heavy Workloads:** InnoDB is more suitable for systems with high transaction rates, where data consistency and concurrency are essential.
* **Data Integrity:** InnoDB ensures data integrity with features like transactional support, automatic crash recovery, and support for foreign keys.

**Use Cases for InnoDB:**

* **Applications that require transactions** (e.g., banking systems, e-commerce sites).
* **Write-heavy applications** with many concurrent transactions.
* Applications that need **foreign key relationships** (e.g., complex database designs).

Joins Practice

Table

**CREATE TABLE employees (**

**id INT PRIMARY KEY,**

**name VARCHAR(100),**

**manager\_id INT**

**);**

**A screenshot of a computer

Description automatically generated**

**SELF JOIN**

In MySQL, a self join is a technique where a table is joined with itself. This is useful when we have hierarchical data or need to compare rows within the same table.

**Example:**

**Select e1.name as Employee, e2.name as Manager**

**From employees e1, employees e2**

**Where e1.id = e2.manager\_id**

**Group By**

**SELECT name,SUM(sal) as sale FROM `emp`**

**group by name;**