**UNIT – 5**

**OVERVIEW OF MYSQL**

**MySQL**

MySQL is an open-source Relational Database Management System (RDBMS) that is widely used for managing relational databases**.** It is used to store, retrieve, and manage data in a structured format. It utilizes tables composed of rows and columns to efficiently organize data**.** It is used by many of the most accessed applications including Facebook, Twitter, Netflix, Uber, and booking.com.

**Features of MySQL**

1. **RDBMS –** The database language is based on SQL queries to access or manage records in the table.
2. **Easy to use –** To work with MySQL basic knowledge of SQL is required, we can interact with MySQL by using a few simple SQL statements.
3. **Secure –** MySQL consists of a solid data security layer that protects sensitive data from intruders. Passwords are also encrypted in MySQL.
4. **Open Source –** MySQL is free to use so we can download it from MySQL's official website without any cost.
5. **Scalable -** MySQL supports multithreading which makes it easily scalable. It can handle almost any amount of data, as much as 50 million rows or more. The default file size limit is about 4 GB. However, this can be increased to a theoretical limit of 8 TB of data.
6. **Speed –** MySQL is considered as one of the very fast database languages backed by a large number of benchmark tests.
7. **Compatibility with many OS –** My SQL is compatible to run on many OS such as Windows, Linux, Netware, Unix, etc. MySQL also provides a facility that clients can run on the same computer as the server or on another computer by a LAN or Internet.
8. **Dual password support –** MySQL versions 8.2 provide sport for dual passwords; one is a current password and the second is a secondary password.
9. **Client-Server architecture –** MySQL follows the working of a client-server architecture. There is a database server and many clients which communicate with the server. That is, they can query data, save changes, etc.
10. **GUI support -** MySQL provides a unified visual database GUI tool named as MySQL Workbench to work with database architects, developers and DBAs.

**DATABASE OBJECTS**

Database objects are defined as objects in a database which is used to store or reference the data. Anything that we make with the create command is known as a database object. It can be used to hold and manipulate the data. Various database objects are -

1. **Database –** A database is an organized collection of interrelated data stored in the computer system and usually controlled by a DBMS. In a database, data is modelled in tables, making, querying and processing efficient. SQL is commonly used for this purpose. For example, in bank database, in school, in a grocery store, etc. These are the instances where a large amount of data is to be stored in one place. The following command is used to create a database-

**Syntax –** Create database database\_name;

1. **Table -** A table is a collection of data that is organized in terms of rows and columns in DBMS table is known as a relation and a row is known as a tuple. The following syntax is used to create a table-

**Syntax –**

Create table table\_name

(

Column 1 datatype(size) constraints,

Column 1 datatype(size) constraints,

-

Column n datatype(size) constraints,

);

1. **View -** In MySQL, views are a kind of virtual table. A few also have rows and columns that are in real tables 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 rows of a table or specific rows on the basis of a condition.

**Syntax –**

Create view view\_name as

SELECT column(s)

FROM table\_name(s)

Where condition;

**Example –**

1. From a single table -

Create view StudentDetails as

SELECT name, address

FROM Student

Where SID<5;

1. From two tables -

Create view StudentMarks as

SELECT Student.name, Student.address, Marks.Smarks

FROM Student, Marks

Where Student.name=Marks.name;

1. **Index –** The create index statement is used to create indexes in the tables. Indexes are used to retrieve data from the database more quickly than otherwise. The user can't see the indexes, they are just used to speed up the queries or searches.

**Syntax –**

Create index index\_name on table\_name (column 1, column 2 ……. Column n);

1. **Alias -** The aliases are the temporary names given to tables or columns for the purpose of a particular SQL query. It is used when names of the columns or tables is used other than their original name, but the modified name is temporary. Aliases are created to make tables or columns more readable. These are preferred when more than one tables are involved in the query.

**Syntax-**

1. **For columns**

Select column\_name as alias\_name from table\_name;

1. **For tables**

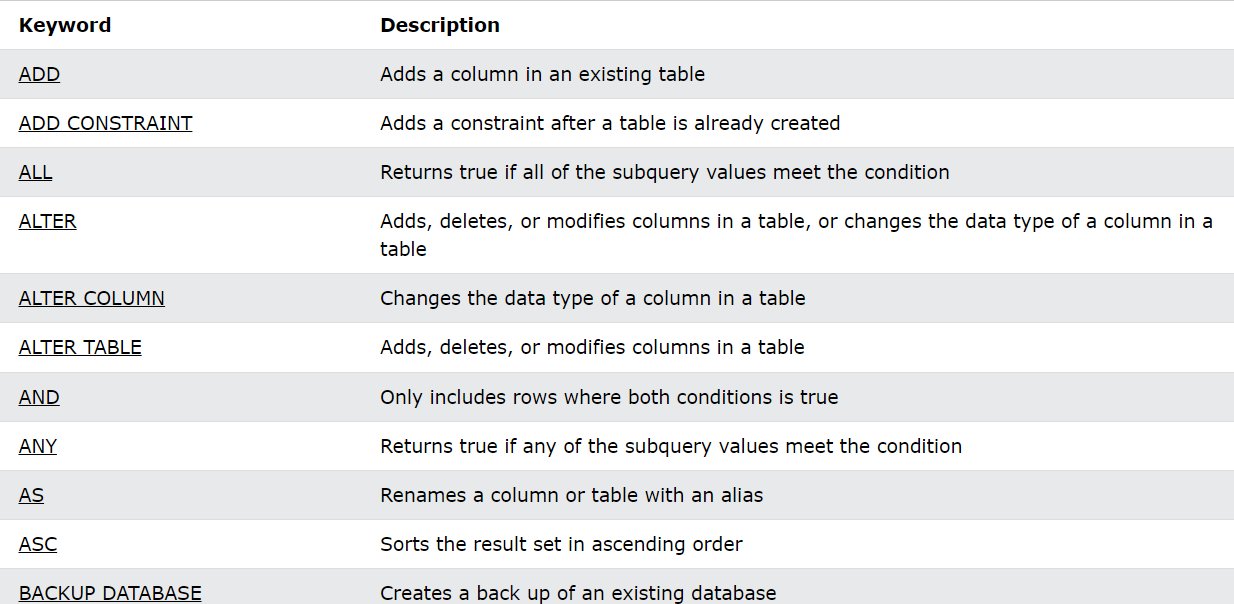
Select column\_name from table\_name as alias\_name;

**MySQL Object Naming**

When naming objects in MySQL, such as databases, tables, columns, indexes, and other database elements, there are several best practices and conventions to follow:

1. **Descriptive and Meaningful Names:** Choose names that clearly describe the purpose or content of the object. For example, ‘customer\_orders’ is more meaningful than ‘co’.
2. **Use Lowercase Letters:** MySQL table names are case-sensitive on Unix-based systems but not on Windows. To avoid issues, it's a common practice to use lowercase letters for all names.
3. **Avoid Reserved Words:** Do not use MySQL reserved keywords for object names. If necessary, use backticks (`) to escape them, but it's better to avoid them altogether.
4. **No Special Characters:** Stick to letters, numbers, and underscores (\_). Avoid using spaces, hyphens, or other special characters.
5. **Length Considerations:** Keep names reasonably short but still descriptive. Long names can be cumbersome to work with, and extremely long names might exceed MySQL's length limits (usually 64 characters for table names).
6. **Singular vs. Plural Names:** Consistently use either singular or plural forms for table names. There are no strict rules, but consistency within a project is crucial. For example, if you name one table customer, name another order instead of orders.
7. **Use Prefixes Wisely:** Prefixes can be useful for categorizing tables (e.g., app\_user and app\_order), but they should be used judiciously to avoid overly verbose names.

**MySQL KEYWORDS**

****

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**USER-DEFINED VARIABLES**

MySQL also supports the concept of user-defined variables, which allow the passing of a value from one statement to another user-defined variable in MySQL is written as

@var\_name, where var\_name is the name of the variable and can contain alphanumeric characters such as (,), (\_),($) etc.

A user-defined variable is session-specific. That is the variable defined by one client is not shared with another client and when the session ends these variables are automatically expired. These are not case-sensitive that is @mark and @MARK are the same. The maximum length of user-defined variable can be 64 characters.

Variable names can include other characters like (! #, ^,-, etc.) In its name, if they are quoted, for example-

@’var@1’ or @’var^2’ or @var3

These variables can take values from the following set of deadlines such as integer, floating point, decimal, fine, your null value.

**Syntax –**

**SET @var\_name = expression;**

**Example-**

mysql > SET @var1 = 2+6;

mysql > select @var1;

mysql > select @var2 :=4;

**DATA TYPES**

An SQL developer must decide what type of data will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside each column, and it also identifies how SQL will interact with the stored data.

In MySQL there are three main data types: string, numeric, and date and time.

1. **STRING DATA TYPES**

|  |  |  |
| --- | --- | --- |
|  | **DATA TYPE** | **DESCRIPTION** |
| **1** | **CHAR(size)** | A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1 |
| **2** | **VARCHAR(size)** | A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535 |
| **3** | **BINARY(size)** | Equal to CHAR() but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1 |
| **4** | **VARBINARY(size)** | Equal to VARCHAR() but stores binary byte strings. The size parameter specifies the maximum column length in bytes. |
| **5** | **BLOB(size)** | For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data |
| **6** | **TINYBLOB** | For BLOBs (Binary Large Objects). Max length: 255 bytes |
| **7** | **MEDIUMBLOB** | For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data |
| **8** | **LONGBLOB** | For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data |
| **9** | **TEXT(size)** | Holds a string with a maximum length of 65,535 bytes. |
| **10** | **TINYTEXT** | Holds a string with a maximum length of 255 characters |
| **11** | **MEDIUMTEXT** | Holds a string with a maximum length of 16,777,215 characters |
| **12** | **LONGTEXT** | Holds a string with a maximum length of 4,294,967,295 characters |

1. **Numeric Data Types**

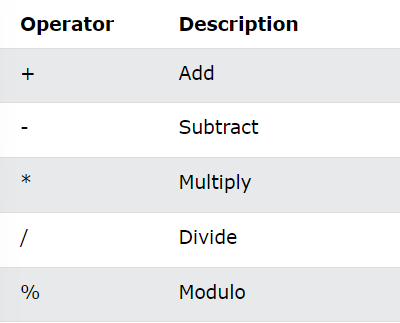
|  |  |  |
| --- | --- | --- |
| **1** | **BIT(size)** | A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1. |
| **2** | **TINYINT(size)** | A very small integer. The signed range is from -128 to 127. The unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255) |
| **3** | **BOOL** | Zero is considered as false, and nonzero values are considered as true. |
| **4** | **BOOLEAN** | Equal to BOOL |
| **5** | **SMALLINT(size)** | A small integer. The signed range is from -32768 to 32767. The unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255) |
| **6** | **MEDIUMINT(size)** | A medium integer. The signed range is from -8388608 to 8388607. The unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255) |
| **7** | **INT(size)** | A medium integer. The signed range is from -2147483648 to 2147483647. The unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255) |
| **8** | **INTEGER(size)** | Equal to INT(size) |
| **9** | **BIGINT(size)** | A large integer. The signed range is from -9223372036854775808 to 9223372036854775807. The unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255) |
| **10** | **FLOAT(size, d)** | A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions |
| **11** | **FLOAT(p)** | A floating point number. MySQL uses the p-value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE() |
| **12** | **DOUBLE(size, d)** | A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter |
| **13** | **DECIMAL(size, d)** | An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0. |
| **14** | **DEC(size, d)** | Equal to DECIMAL(size,d) |

1. **DATE AND TIME DATA TYPES**

|  |  |  |
| --- | --- | --- |
|  | **DATA TYPE** | **DESCRIPTION** |
| **1** | **DATE** | A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31' |
| **2** | **DATETIME(fsp)** | A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time |
| **3** | **TIMESTAMP(fsp)** | A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT\_TIMESTAMP and ON UPDATE CURRENT\_TIMESTAMP in the column definition |
| **4** | **TIME(fsp)** | A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59' |
| **5** | **YEAR** | A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000.  MySQL 8.0 does not support year in two-digit format. |

**MySQL OPERATORS**

1. **MySQL Arithmetic Operators**

****

1. **MySQL Bitwise Operators**

**A screenshot of a phone

Description automatically generated**

1. **MySQL Comparison/Relational Operators**

**A screenshot of a computer

Description automatically generated**

1. **MySQL Compound Operators**

**A table with text on it

Description automatically generated**

1. **Mysql Logical operators**
2. **All –** it returns true if all the subquery value meets the condition.

**Example-**

Select\* from student where sage > ALL (select sage from student where sage >21);

1. **AND –** it returns TRUE if all conditions separated by AND is TRUE.

**Example-**

Select \* from student where address = “Hamirpur” AND country = “India”;

1. **ANY –** Returns TRUE if any of the subquery values meet the condition.

**Example –**

Select \* from student where sage > ANY(select sage from student where sage > 21);

1. **OR –** TRUE if any of the conditions separated by OR is true.

**Example –**

Select \* from student where address = “Hamirpur” OR country = “India”;

1. **IN** – It returns TRUE if an operand is equal to one of a list of expressions.

**Example** –

Select \* from student where address IN (‘Hamirpur’, ‘Chamba’);

1. **BETWEEN –** It returns TRUE if operands is within the range of comparison.

**Example –**

Select sage from student where sage BETWEEN 18 AND 21;

1. **LIKE –** It returns TRUE if operands match a pattern.

**Example –**

Select \* from student where address LIKE ‘S%’ ;

1. **NOT –** It displays a record if the condition is NOT TRUE.

**Example –**

Select \* from student where address NOT like ‘S%’ ;

1. **SOME ­-**  It returns TRUE if any of the subquery values meet the condition.

**Example –**

Select \* from student where sage > SOME (select sage from student where sage >20);

1. **EXISTS –** It returns TRUE if a subquery returns one or more records.

**Example –**

Select \* from student where EXISTS(select sage from student where sage >21);

1. **STRING OPERATORS**
2. **ASCII (**American Standard **) -**