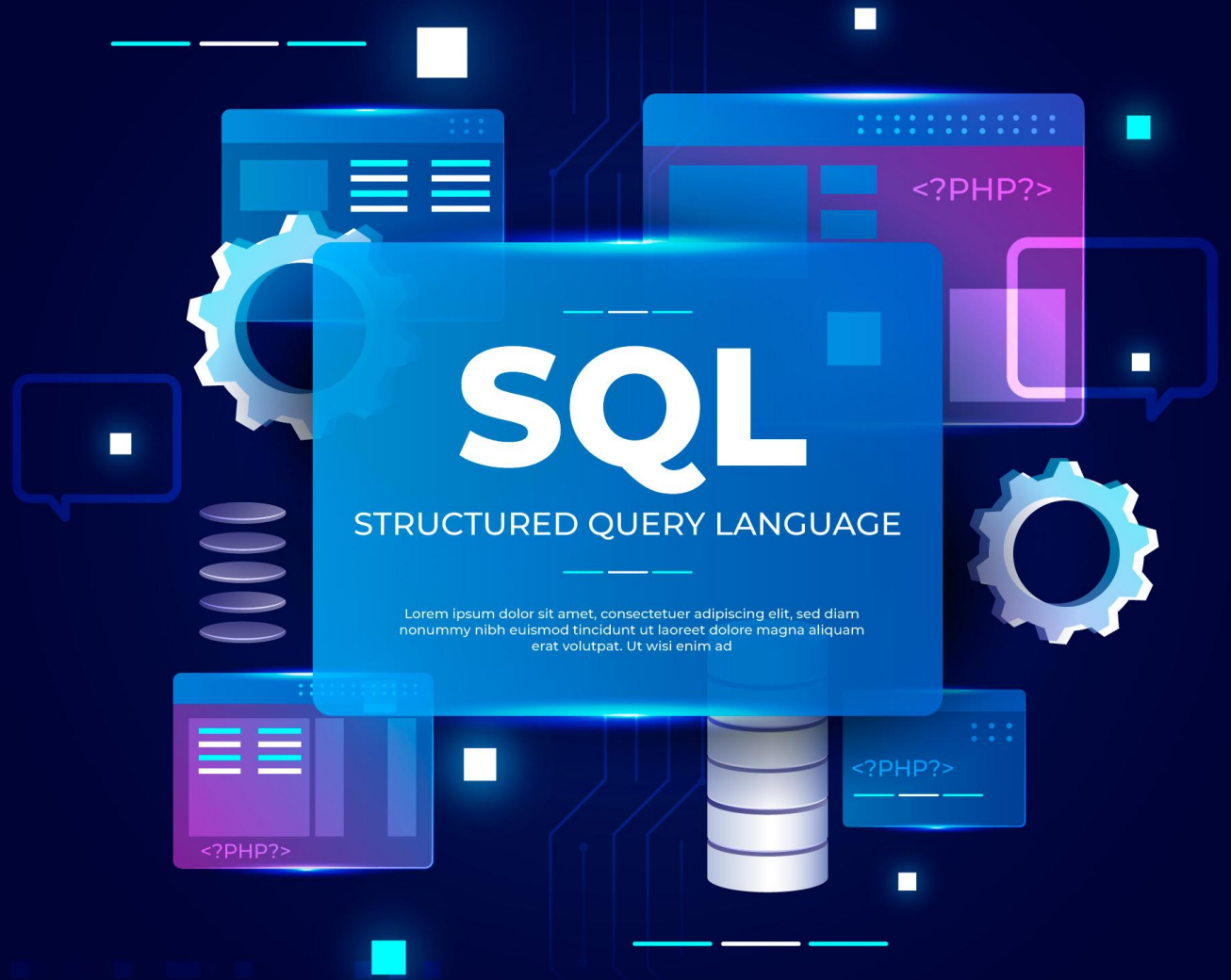

# Must-Know SQL-Related Key Terms

- **Database**

A **database** is a set of data stored in a computer and is usually structured to make the data easily accessible.

- **Relational Database Management System**

A **relational database** is a type of database that allows us to identify and access data in relation to another piece of data in the database. It stores data in rows and columns in a series of tables to make processing and querying efficient.

# Must-Know SQL-Related Key Terms

- **Storage Engine**

A storage engine is a piece of software that a database management system uses to create, read, and update data from a database.

- **Open Source**

Open source means software in which the original source code is freely available to all and may be redistributed and modified.

# What is SQL?

- SQL is the standardized programming language to manage **relational databases**, create SQL databases, and manipulate the data in them by performing different functions.
- SQL was introduced in the **1970s**. The term 'SQL' is pronounced as sequel or *"ess-kew-ell"*.

# What is MySQL?

- MySQL is an open source Relational Database Management System (RDBMS) owned by Oracle.
- MySQL was created by a Swedish company, MySQL AB, founded by **David Axmark**, **Allan Larsson,** and **Michael "Monty" Widenius**. The first version of MySQL appeared on 23 May 1995.
- Its name is a combination of "My", the name of co-founder Michael Widenius' daughter, and "SQL", the abbreviation for **Structured Query Language**.

## What is the major difference between MySQL and SQL?

MySQL is software, but SQL is a database language.

# Definition

- MySQL is a popular choice of database for use in web applications and is a central component of the widely used LAMP, WAMP, and XAMPP.
- LAMP is an acronym for “Linux, Apache, MySQL, Perl/PHP/Python”
- WAMP is an acronym for “Windows, Apache, MySQL, Perl/PHP/Python”
- XAMPP is an acronym for “X for Cross-Platform, A for Apache, M for MySql, P for PHP, and P for Perl.”

# Functions of MySQL

MySQL is the most popular database system used with PHP.

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on several platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

# When is MySQL used?

MySQL can be used for various applications, including data warehousing, e-commerce and logging. However, it’s often found on web servers.

# MySQL DATA TYPES

Properly defining the fields in a table is important to optimizing your database. You should use only the type and size of the field you need to use.

MySQL uses many different data types broken into categories:

- **String**
- **Numeric**
- **Date and time**

# The syntax of data types is as follows:

```
CREATE TABLE table_name (
    column1_name data type(length),
    column2_name data type(length),
);
```

# Example

```
CREATE TABLE products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_item VARCHAR(255) NOT NULL,
    use_by date,
    price int,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

# MySQL String Data Types (Text Formats)

String data types usually store data as long strings of text, for example, feedback or product description. Strings can consist of letters, numbers, or binary media files such as images, photos, or audio files.

The MySQL string data types are divided into:

1. TEXT
2. BLOB
3. CHAR and VARCHAR
4. ENUM

# TEXT Data Type

The MySQL TEXT data type stores long-text strings to display information about the table object, such as product descriptions, blog comments, etc.

| Type | Storage | Maximum number of characters | Overhead storage (in bytes) | Usage |
|---|---|---|---|---|
| TINYTEXT | 255 | 255 | 1 | To store short-text strings such as links, product description or summary |
| TEXT | 64 kB | 65535 | 2 | To store texts such as articles that do not exceed the specified length of characters |
| MEDIUMTEXT | 16 MB | 16777215 | 3 | To store large texts such as whitepapers or books |
| LONGTEXT | 4 GB | 4294967295 | 4 | To store huge texts such as computer programs or applications |

# BLOB Datatype in MySQL

The BLOB data type represents a binary large object and can store binary media data, such as audio or video links, images, or files.

TINYBLOB => 255 bytes + 1 byte

BLOB => 65535 + 2 bytes

MEDIUMBLOB => 16777215 + 3 bytes

LONGBLOB => 4294967295 + 4 bytes

# CHAR and VARCHAR data type

The **CHAR data types** store non-binary strings with a fixed length that reaches 255 characters, while the **VARCHAR data types** store non-binary strings with a variable length having a maximum size of up to 65535 characters.

Example:

name varchar(255)

# ENUM Data Type in MySQL

SQL ENUM data types are strings with enumeration values. ENUM allows you to set a list of predefined values and then choose any of them. If you add an invalid value not included in the list, you will get an empty string.

```
CREATE TABLE clothes (
    product_ID int PRIMARY KEY AUTO_INCREMENT,
    name varchar(255) NOT NULL,
    fabric text NOT NULL,
    size enum ('small', 'medium', 'large') NOT
NULL
);

INSERT INTO clothes (product_ID, name, fabric,
size)
    VALUES (1, 'dresses', 'cotton', 'small');
```

```
INSERT INTO clothes (product_ID, name, fabric,
size)
VALUES (2, 'dresses', 'silk', 'extra large');
```

# MySQL Numeric Data Types (Number Formats)

- **Integers** represent numbers without fractions and can have SIGNED and UNSIGNED attributes. Usually, they may be used for IDs or counting numbers.
- **Decimals** represent numbers with fractions and store exact numeric values in the column. They can be signed and unsigned and are usually used for columns that store monetary values. In comparison with the floating-point numbers, decimals are more accurate.
- **Floating points** represent numbers with fractions but do not store exact numeric values. They can be signed and unsigned. Floating-point numeric values use a double-precision 64-bit format or a single-precision 32-bit format to store data. They may lead to a loss of precision during arithmetic operations.

# SIGNED vs UNSIGNED

UNSIGNED can store only zero and positive numbers in a column.

SIGNED can allow zero, positive, and negative numbers.

```
CREATE TABLE products (
    product_ID int PRIMARY KEY AUTO_INCREMENT,
    product_item varchar(255) NOT NULL,
    category varchar(255) NOT NULL,
    total_amount int UNSIGNED,
);
```

```
INSERT INTO products (product_ID, product_item,
category, total_amount)
VALUES (1, ‘tomatoes', ‘vegetables’, 10);
```

# Boolean Data Type

The boolean data types can only accept either true or false values. In a binary format, true refers to 1 and false – to 0. As a rule, they are used for logical operations.

# Float Data Type

The Float data types represent single-precision approximate numeric values that require 4 bytes for storage. They can have SIGNED and UNSIGNED attributes. When adding a column, you need to set values for the float data type in brackets – FLOAT(m,d) where ‘m‘ is the number of digits in total and ‘d‘ is the number of digits after the decimal point.

| Attribute | Minimum storage size | Maximum storage size |
|---|---|---|
| SIGNED | -3.402823466E+38 | -1.175494351E-38 |
| UNSIGNED | 0 and 1.175494351E-38 | 3.402823466E+38 |

Note: Starting from MySQL version 8.0.17, UNSIGNED is deprecated for the FLOAT and DOUBLE data types.

# Double Data Type

The Double data types refer to the floating-point numeric data types and use 8 bytes to store double-precision values. The syntax for the double data type is DOUBLE PRECISION(m,d) where ‘m‘ is the total number of digits and ‘d‘ is the number of digits following the decimal point.

For example, DOUBLE(7,5) means it will store a value with seven digits and five decimals.

| Attribute | Minimum storage size | Maximum storage size |
|---|---|---|
| SIGNED | -1.7976931348623157E+308 | -2.2250738585072014E-308 |
| UNSIGNED | 0 and 2.2250738585072014E-308 | 1.7976931348623157E+308 |

# Decimal Data Type

The DECIMAL data type can be used to store exact and fixed numeric values. When creating a table column, the syntax for the data type is DECIMAL(p,s), where ‘p‘ stands for precision, the maximum number of digits, and ‘s‘ stands for scale, the number of digits following the decimal.

***decimals*** *are better to use for fixed amounts, such as monetary and financial information (price, salary, etc,). At the same time,* ***float and double*** *– for approximate calculations where rounding values might have a negative impact.*

# MySQL Date & Time Data Types

For managing date and time information in databases, MySQL date types are used that are divided into DATE, TIME, DATETIME, TIMESTAMP, and YEAR.

| Type | Usage | Data type format | Range |
|---|---|---|---|
| **DATE** | Stores only date information in the table column | YYYY-MM-DD format (year, month, and date) | from '1000-01-01' to '9999-12-31' |
| **TIME** | Displays only time | HH:MM:SS format (hours, minutes, and seconds) | from '-838:59:59' to '838:59:59' |
| **DATE TIME** | Stores both date and time in the column | YYYY-MM-DD HH:MM:SS ( year, month, and date, and hours, minutes, and seconds) | from '1000-01-01 00:00:00' to '9999-12-31 23:59:59' |
| **DATE TIME** | Stores both date and time values in the column | YYYY-MM-DD HH:MM:SS ( year, month, and date, and hours, minutes, and seconds) | from '1000-01-01 00:00:00' to '9999-12-31 23:59:59' |
| **TIME STA MP** | Stores both date and time values in the column. Conversion of the value from the zone of the connection server to UTC takes place. | YYYY-MM-DD HH:MM:SS ( year, month, and date, and hours, minutes, and seconds) | from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC |
| **YEAR** | Stores only year values in the column | YYYY (year) | from '1901' to '2155' |

```
CREATE TABLE employees(
    employee_id INT AUTO_INCREMENT,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    date_of_birth date, PRIMARY KEY(employee_id)
);

INSERT INTO employees (employee_id, first_name,
last_name, date_of_birth)
    VALUES (1, 'John', 'Sanders', '2000-01-19');
```

# MySQL Commands and Functions

# Commonly used MySQL commands

- **CREATE TABLE: Creates a new table with specified columns and data types.**

```
CREATE TABLE table_name (
        column1 datatype,
        column2 datatype,
        ...
);
```

# Commonly used MySQL commands

- **INSERT INTO: Adds new records into a table.**

```
CREATE TABLE table_name (
        column1 datatype,
        column2 datatype,
        ...
);
```

# Commonly used MySQL commands

- **SELECT: Retrieves data from one or more tables.**

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

# Commonly used MySQL commands

- **UPDATE: Modifies existing records in a table.**

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

# Commonly used MySQL commands

- **ALTER TABLE: Modifies an existing table (e.g., adds or removes columns).**

  ***ALTER TABLE table_name***
  ***ADD column_name datatype;***

# Commonly used MySQL commands

- **DROP TABLE: Deletes an entire table and its data.**

  ***DROP TABLE table_name;***

# Commonly used MySQL commands

- **CREATE DATABASE: Creates a new database.**

***CREATE DATABASE database_name;***

# Commonly used MySQL commands

- **SHOW TABLES: Displays a list of tables in the current database.**

***SHOW TABLES;***

# Create a table named products to store information about various products.

```
-- Create a table named 'products’
CREATE TABLE products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    category VARCHAR(100),
    price DECIMAL(10, 2),
    in_stock BOOLEAN
);
```

# Data

| product_id | product_name | category | price | in_stock |
|---|---|---|---|---|
| 1 | Laptop | Electronics | 899.99 | 1 |
| 2 | Smartphone | Electronics | 499.99 | 1 |
| 3 | Coffee Maker | Appliances | 59.95 | 0 |
| 4 | Running Shoes | Apparel | 79.99 | 1 |
| 5 | Bluetooth Speaker | Electronics | 39.99 | 1 |

```
-- Insert data into the 'products' table
INSERT INTO products (product_name, category, price, in_stock)
VALUES
    ('Laptop', 'Electronics', 899.99, TRUE),
    ('Smartphone', 'Electronics', 499.99, TRUE),
    ('Coffee Maker', 'Appliances', 59.95, FALSE),
    ('Running Shoes', 'Apparel', 79.99, TRUE),
    ('Bluetooth Speaker', 'Electronics', 39.99, TRUE);
```