MySQL Variables

Variables are used for storing data or information during the execution of a program. It is a way of labeling data with an appropriate name that helps to understand the program more clearly by the reader. The main purpose of the variable is to store data in memory and can be used throughout the program.

MySQL can use variables in **three** different ways, which are given below:

1. User-Defined Variable
2. Local Variable
3. System Variable

User-Defined Variable

Sometimes, we want to pass values from one statement to another statement. The user-defined variable enables us to store a value in one statement and later can refer it to another statement. MySQL
provides a **SET** and **SELECT** statement to declare and initialize a variable. The user-defined variable name starts with **@ symbol**.

The user-defined variables are not case-sensitive such as @name and @NAME; both are the same. A user-defined variable declares by one person cannot visible to another person. We can assign the user-defined variable into limited data types like integer, float, decimal, string, or NULL. The user-defined variable can be a maximum of **64 characters** in length.

Syntax

The following syntax is used to declare a user-defined variable.

1. By using the **SET** statement

```
SET @var_name = value;
```

NOTE: We can use either '=' or ':=' assignment operator with the SET statement.

2. By using the **SELECT** statement

```
SELECT @var_name := value;
```

Example1

Here, we are going to assign a value to a variable by using the SET statement.

```
mysql> SET @name='peter';
```

Then, we can display the above value by using the SELECT statement.

```
mysql> SELECT @name;
```

Output

```
mysql> SET @name = 'peter';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT @name;
+-------+
| @name |
+-------+
| peter |
+-------+
1 row in set (0.00 sec)
```

Example 2

Let us create table **students** in the MySQL database, as shown below:

studentid	firstname	lastname	class	age
1	Rinky	Ponting	12	20
2	Mark	Boucher	11	22
3	Sachin	Tendulkar	10	18
4	Peter	Fleming	10	22
5	Virat	Kohli	12	23
NULL	NULL	NULL	NULL	NULL

Run the following statement to get the maximum age of the student in the 'students' table and assign the age to the user-defined variable **@maxage**.

```
mysql> SELECT @maxage:= MAX(age) FROM students;
```

It will give the following output.

	@maxage:= MAX(age)
▶	23

Now, run the SELECT statement that uses the @maxage variable to return the maximum age of the student.

```
mysql> SELECT firstname, lastname, age FROM students WHERE age = @maxage;
```

After successful execution of the above statement, we will get the following result:

	firstname	lastname	age
▶	Virat	Kohli	23

Example3

If we access the **undeclared** variable, it will give the **NULL** output.

```
Mysql> SELECT @var1;
```

Output

```
mysql> SELECT @var1;
+--------------+
| @var1        |
+--------------+
| 0x           |
+--------------+
1 row in set (0.00 sec)
```

Local Variable

It is a type of variable that is not prefixed by **@ symbol**. The local variable is a strongly typed variable. The scope of the local variable is in a stored program block in which it is declared. MySQL uses the **DECLARE** keyword to specify the local variable. The DECLARE statement also combines a **DEFAULT** clause to provide a default value to a variable. If you do not provide the DEFAULT clause, it will give the initial value **NULL**. It is mainly used in the stored procedure program.

Syntax

We can use the DECLARE statement with the following syntax:

```
DECLARE variable_name datatype(size) [DEFAULT default_value];
```

Let us see the following example to use the local variable.

Example

```
mysql> DECLARE total_price Oct(8,2) DEFAULT 0.0;
```

We can also define two or more variables with the same data type by using a single DECLARE statement.

```
mysql> DECLARE a,b,c INT DEFAULT 0;
```

The below example explains how we can use the DECLARE statement in a stored procedure.

```
DELIMITER //
```

```
Create Procedure Test()
  BEGIN
    DECLARE A INT DEFAULT 100;
    DECLARE B INT;
    DECLARE C INT;
    DECLARE D INT;
    SET B = 90;
    SET C = 45;
    SET D = A + B - C;
    SELECT A, B, C, D;
  END //
DELIMITER ;
```

After successful execution of the above function, call the stored procedure function as below:

```
mysql> CALL Test();
```

It will give the following output:

	A	B	C	D
▶	100	90	45	145

System Variable

System variables are a special class to all program units, which contains **predefined** variables. MySQL contains various system variables that configure its operation, and each system variable contains a default value. We can change some system variables dynamically by using the **SET** statement at runtime. It enables us to modify the server operation without stop and restart it. The system variable can also be used in the expressions.

MySQL server gives a bunch of system variables such as GLOBAL, SESSION, or MIX types. We can see the GLOBAL variable throughout the lifecycle of the server, whereas the SESSION variable remains active for a particular session only.

We can see the names and values of the system variable by using the following ways:

1. To see the current values used by the running server, execute the following command.

```
mysql> SHOW VARIABLES;

OR,

Mysql> SELECT @@var_name;
```

2. When we want to see the values based on its compiled-in defaults, use the following command.

```
mysql> mysqld --verbose --help
```

Example1

```
mysql> SHOW VARIABLES LIKE '%wait_timeout%';
```

Output

```
mysql> SHOW VARIABLES LIKE '%wait_timeout%';
+--------------------------+----------+
| Variable_name            | Value    |
+--------------------------+----------+
| innodb_lock_wait_timeout | 50       |
| lock_wait_timeout        | 31536000 |
| mysqlx_wait_timeout      | 28800    |
| wait_timeout             | 28800    |
+--------------------------+----------+
4 rows in set (0.01 sec)
```

Example 2

```
mysql> SELECT @@key_buffer_size;
```

Output

```
mysql> SELECT @@key_buffer_size;
+-------------------+
| @@key_buffer_size |
+-------------------+
|           8388608 |
+-------------------+
1 row in set (0.00 sec)
```
