How To Perform Basic Queries in MySQL and MariaDB on a Cloud Server

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: http://faq.asphosthelpdesk.com/article.php?id=244

MySQL and MariaDB are relational database management systems. These tools can be used on your server to manage the data from many different programs. Both implement forms of the SQL querying language, and either can be used on a cloud server.

This guide will cover how to query information from your databases by specifying different search criteria. This will help you pull needed data in flexible ways from your databases.

For this guide, we will be using MySQL on an Ubuntu 12.04 cloud server, but the steps involved will work on either MySQL or MariaDB on any modern Linux distribution.

How To Query Information from MySQL and MariaDB

We tell the database software to retrieve information from our databases using the following syntax:

```
SELECT selection_fields FROM data_source WHERE selection_criteria_is_met;
```

Selecting a Data Source to Query in MySQL and MariaDB

We will use the default "mysql" database to practice forming queries:

```
USE mysql;

Database changed
```

View the tables within the "mysql" database to get an idea of the data sources we can query:

```
SHOW TABLES;

+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
. . .
. . .
```

On Ubuntu's default MySQL installation, this command returns 24 results.

Let's choose the "user" table to query results. To get an idea of which categories we can select from, view the table column headers:

```
SHOW COLUMNS FROM user;
```

```
+-------------------------+-----------------------------------+------+-----+---------+-
| Field                   | Type                              | Null | Key |
Default | Extra |
```

```
+-------------------------+-----------------------------------+------+-----+---------+-
| Host                    | char(60)                          | NO   | PRI |
|       |
| User                    | char(16)                          | NO   | PRI |
|       |
| Password                | char(41)                          | NO   |     |
|       |
| Select_priv             | enum('N','Y')                     | NO   |     | N
|       |
. . .
. . .
```

The values in the "Field" column are column headings for the "user" table.

How To Create a Basic Query in MySQL and MariaDB

Let's select some basic information about our users:

```
SELECT user,password,host FROM user;
```

```
+------------------+-------------------------------------------+-----------+
| user             | password                                  | host      |
+------------------+-------------------------------------------+-----------+
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | localhost |
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | 127.0.0.1 |
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | ::1       |
| debian-sys-maint | *9AE1BDEEBCD7F85D6304C3FB18146904CE3496F6 | localhost |
+------------------+-------------------------------------------+-----------+
4 rows in set (0.00 sec)
```

The selection fields we've chosen are three of the columns from the "user" table. Notice how we've separated the different fields with a comma.

Each column that we request is returned, in order, from every record in the table.

How To Use Wildcards in MySQL and MariaDB Queries

The previous example returned a nice set of data. That syntax is useful if you wish to return all data related to specific, known criteria.

The SQL querying language allows for more flexible searches. If we wanted to return everything from the "user" table, we could use the asterisk (*) wildcard, which matches any value:

```
SELECT * FROM user;
```

The results are a bit unwieldy, so they will not be included here. The command will give you every value in every column for every record (line) in the table.

How To Filter Results in MySQL and MariaDB

If we want to only return results that meet a certain criteria, we can filter the results with a "where" filter.

For instance, if we only want to return table data where the user is "debian-sys-maint", we can filter the query like this:

```
SELECT user,password,host FROM user WHERE user = "debian-sys-maint";
```

```
+------------------+-------------------------------------------+-----------+
| user             | password                                  | host      |
+------------------+-------------------------------------------+-----------+
| debian-sys-maint | *9AE1BDEEBCD7F85D6304C3FB18146904CE3496F6 | localhost |
+------------------+-------------------------------------------+-----------+
1 row in set (0.00 sec)
```

In the same way, we can see which users are attached to the "localhost" host:

```
SELECT user,password,host FROM user WHERE host = "localhost";

+------------------+-------------------------------------------+-----------+
| user             | password                                  | host      |
+------------------+-------------------------------------------+-----------+
| debian-sys-maint | *9AE1BDEEBCD7F85D6304C3FB18146904CE3496F6 | localhost |
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | localhost |
+------------------+-------------------------------------------+-----------+
2 rows in set (0.00 sec)
```

If we want to filter by more than one term, we can separate the criteria with "and".

```
SELECT user,password,host FROM user WHERE host = "localhost" AND user = "root";

+------+-------------------------------------------+-----------+
| user | password                                  | host      |
+------+-------------------------------------------+-----------+
| root | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | localhost |
+------+-------------------------------------------+-----------+
1 row in set (0.00 sec)
```

How To Use the Like Comparison Operator

In "where" filters, the percentage sign (%) is used as the "everything" wildcard instead of an asterisk. It will match zero or more characters.

Using this knowledge, we can make use of the "like" comparison operator, which is another way to filter with "where".

The "like" operator compares the values of a certain field to an expression on the right side. For instance, to select the lines that have a user that starts with "d", we can perform the following query:

```
SELECT user,password,host FROM user WHERE user LIKE "d%";

+------------------+-------------------------------------------+-----------+
| user             | password                                  | host      |
+------------------+-------------------------------------------+-----------+
| debian-sys-maint | *9AE1BDEEBCD7F85D6304C3FB18146904CE3496F6 | localhost |
+------------------+-------------------------------------------+-----------+
1 row in set (0.00 sec)
```

You can specify "like" comparisons using "and" or "or" just like you can with equality comparisons:

```
SELECT user,password,host FROM user WHERE user LIKE "d%" OR user LIKE "r%";

+------------------+-------------------------------------------+-----------+
| user             | password                                  | host      |
+------------------+-------------------------------------------+-----------+
```

```
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | localhost |
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | 127.0.0.1 |
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | ::1       |
| debian-sys-maint | *9AE1BDEEBCD7F85D6304C3FB18146904CE3496F6 | localhost |
+------------------+-------------------------------------------+-----------+
4 rows in set (0.00 sec)
```

Conclusion

You now should have a basic understanding of how to retrieve data from your databases.

There are many other queries that we have not covered, but the list that we mentioned is a good start on how to perform information retrieval in MySQL and MariaDB.