Getting Started with MariaDB

Daniel Bartholomew

Chapter No. 4
"MariaDB User Account Management"
In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.4 "MariaDB User Account Management"

A synopsis of the book’s content

Information on where to buy this book

---

About the Author

**Daniel Bartholomew** has been using Linux since 1997 and databases since 1998. He has written dozens of articles for various magazines, including The Linux Journal, Linux Pro, Ubuntu, User and Tux.

Daniel has been involved with the MariaDB project shortly after it began in early 2009. He currently works for SkySQL and splits his time between MariaDB documentation and maintaining the bits and pieces (including build, e-mail, web, and other servers) that keep the MariaDB project running smoothly. In addition to his day-to-day responsibilities, Daniel also serves as the MariaDB release coordinator and has been deeply involved with almost every MariaDB release.

He lives in Raleigh, North Carolina, USA with his lovely wife and four children.

I'd like to thank Amy, Ila, Lizzy, Anthon, and Rachel for their patience with me throughout the writing of this book. Thanks also to Vladislav Vaintroub, Sanja Byelkin, Roger Bartholomew, and others who were very helpful at various points during the project. Lastly, I'd like to thank Monty and the rest of the MariaDB team for the excellent database that they've created.
Getting Started with MariaDB

Databases are all around us. Almost every website we visit and nearly every store we shop at has a database (or several) working quietly behind the scenes. The same goes for banks, hospitals, government agencies, theaters, doctors, amusement parks, and police departments. All use databases to store, sort, and analyze information.

This information comes in many forms and can be anything that can be stored electronically inside a computer. This includes books, catalogs, addresses, names, dates, finances, pictures, money, passwords, documents, preferences, tweets, posts, likes, blogs, articles, and many more. Databases are one of the primary pillars of modern life.

Your posts on Facebook and tweets on Twitter are stored in a database. All your financial information at your bank is stored in a database and so is your purchase history at your favorite online retailer. Your progress in your favorite online game? You guessed it. The record of when you last paid your water bill. You just can't get away from databases. They are quite literally everywhere.

There is a new database that has caught attention of the database community over the past few years like few others have. Its name is MariaDB; it is named after the youngest daughter of its creator, Michael "Monty" Widenius. First released in 2009, MariaDB may be relatively new, but it has a stellar parentage. It's a next-generation version of the popular MySQL database, also created by Monty. (you may have heard of it, but don't worry if you haven't).

MariaDB is open source. This means the source code is freely downloadable and is governed by a license that helps ensure the source code stays free and open to all. The MariaDB developers have also kindly provided installers for various operating systems. Since its first release, MariaDB has gained a large, loyal following quicker than almost any other database. Today it powers tens of thousands of websites, big and small and is the database of choice for many companies in a wide variety of industries around the world, with hundreds of thousands of users.

The great news is that we can install and use it ourselves, right now, on our personal laptop and desktop computers. For all of its power, and make no mistake, MariaDB is a very powerful and capable database; it is very easy to install and use.

This book provides an introduction to MariaDB—enough to get us started. Don't worry if you've never used a database before, they're not that hard to understand.

Before we know it we'll be on our way to becoming an expert database administrator (DBA). But even if we never move beyond just tinkering or playing around with MariaDB, we'll have learned about one of the fundamental technologies of our times.

Not a bad accomplishment over a weekend or two.

For More Information:
What This Book Covers

Chapter 1, Installing MariaDB, explains how to install MariaDB on Windows and Linux.

Chapter 2, Configuring MariaDB, explains the basics of configuring MariaDB, including the location of the configuration files, and how to set common configuration options.

Chapter 3, MariaDB Security, gives out the best practices for MariaDB security and how to easily secure a new MariaDB installation.

Chapter 4, MariaDB User Account Management, explains how to add and administer MariaDB user accounts.

Chapter 5, Using MariaDB, explains the basics of using MariaDB, including adding and dropping databases and tables, and selecting, inserting, and updating data.

Chapter 6, MariaDB Maintenance, explains how to maintain your MariaDB database and keep it running smoothly.

Appendix, MariaDB Next Steps, will provide the user with the locations of official sources of information and documentation.

For More Information:
The root user in our MariaDB database has rights to every database and table, we don't want to use it for day-to-day operations or hand out the login ID and password of root to anyone who doesn't absolutely need to have it. Instead, we want to create users that have specific rights to the specific databases they need to work with.

**User privileges**

The privileges or rights that we can grant to users are many and varied. They break down into three main categories:

- Global administrative privileges
- Database, table, and column privileges
- Miscellaneous privileges
Global administrative privileges

The following table lists the global administrative privileges. Global privileges apply to all databases and tables within those databases, which belong to the entire MariaDB database server.

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE USER</td>
<td>The ability to create a user using the CREATE USER statement.</td>
</tr>
<tr>
<td>FILE</td>
<td>The ability to use the LOAD DATA INFILE statement and the LOAD_FILE() function.</td>
</tr>
<tr>
<td>PROCESS</td>
<td>The ability to use the SHOW PROCESSLIST command.</td>
</tr>
<tr>
<td>RELOAD</td>
<td>The ability to use the FLUSH statement.</td>
</tr>
<tr>
<td>REPLICATION CLIENT</td>
<td>The ability to use the SHOW MASTER STATUS and SHOW SLAVE STATUS commands.</td>
</tr>
<tr>
<td>REPLICATION SLAVE</td>
<td>The ability to get updates made on the replication master server.</td>
</tr>
<tr>
<td>SHOW DATABASES</td>
<td>The ability to list all of the databases on the server.</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>The ability to shut down the server using the mysqladmin shutdown command.</td>
</tr>
<tr>
<td>SUPER</td>
<td>The ability to use superuser statements such as CHANGE MASTER TO..., PURGE LOGS; to SET global variables; and to KILL other users' threads.</td>
</tr>
</tbody>
</table>

Database, table, and column privileges

The following table lists the database and table privileges. These privileges only apply to a specific database or table within a database.

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>The ability to change indexes and tables.</td>
</tr>
<tr>
<td>ALTER ROUTINE</td>
<td>The ability to change or delete procedures and stored functions.</td>
</tr>
<tr>
<td>CREATE</td>
<td>The ability to create databases and tables.</td>
</tr>
<tr>
<td>CREATE ROUTINE</td>
<td>The ability to create procedures and stored functions.</td>
</tr>
<tr>
<td>CREATE TEMPORARY TABLES</td>
<td>The ability to create temporary tables.</td>
</tr>
</tbody>
</table>

For More Information:
Privilege Description

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE VIEW</td>
<td>The ability to create views.</td>
</tr>
<tr>
<td>DELETE</td>
<td>The ability to delete rows from tables.</td>
</tr>
<tr>
<td>DROP</td>
<td>The ability to delete entire databases and tables.</td>
</tr>
<tr>
<td>EVENT</td>
<td>The ability to alter, create, and drop events from the event scheduler.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>The ability to execute stored functions and procedures.</td>
</tr>
<tr>
<td>INDEX</td>
<td>The ability to create or delete indexes.</td>
</tr>
<tr>
<td>INSERT</td>
<td>The ability to insert new rows of data into a table.</td>
</tr>
<tr>
<td>LOCK TABLES</td>
<td>The ability to lock and unlock tables.</td>
</tr>
<tr>
<td>SELECT</td>
<td>The ability to read data from a table.</td>
</tr>
<tr>
<td>SHOW VIEW</td>
<td>The ability to use the SHOW CREATE VIEW statement.</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>The ability to use the CREATE TRIGGER and DROP TRIGGER statements.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>The ability to modify rows in a table.</td>
</tr>
</tbody>
</table>

Column privileges apply to individual columns within a table. There are only three of them: INSERT, UPDATE, and SELECT.

Miscellaneous privileges and limits
The following table lists miscellaneous privileges which don't quite fit into either of the two previous categories.

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USAGE</td>
<td>Grants nothing real, but can be used to change global options for a user.</td>
</tr>
<tr>
<td>ALL PRIVILEGES</td>
<td>Can be used to grant all available privileges to a user. Does not grant the GRANT OPTION privilege. Can be shortened to ALL.</td>
</tr>
<tr>
<td>GRANT OPTION</td>
<td>Gives a user the ability to give other users the privileges they have. This is given at the end of the GRANT statement. See the Granting Permissions section of this chapter for some examples.</td>
</tr>
</tbody>
</table>
There are also several limits we can place on user accounts. These are given in the following:

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_QUERIES_PER_HOUR</td>
<td>The number of SQL statements or queries the user account can issue per hour. This includes updates.</td>
</tr>
<tr>
<td>MAX_UPDATES_PER_HOUR</td>
<td>The number of SQL update statements (not queries) the user account can issue per hour.</td>
</tr>
<tr>
<td>MAX_CONNECTIONS_PER_HOUR</td>
<td>The number of connections the user account can start per hour.</td>
</tr>
<tr>
<td>MAX_USER_CONNECTIONS</td>
<td>The number of simultaneous connections to the database server the user account can have. If set to zero, the number will be equal to the max_connections setting. If the max_connections setting is also zero then there is no limit to the number of simultaneous connections.</td>
</tr>
</tbody>
</table>

Full documentation of the various privileges can be found at [https://mariadb.com/kb/en/grant/](https://mariadb.com/kb/en/grant/).

### Creating users

Creating a user in MariaDB is a two-step process. First, we create the user using the `CREATE USER` statement, and then we give, or `GRANT`, the user the privileges we want them to have. We'll go over the `CREATE USER` statement in this section and the `GRANT` statement in the **Granting permissions** section.

A `CREATE USER` statement has the following pattern:

```
CREATE USER 'username'@'host' IDENTIFIED BY 'password';
```

We customize the username, host, and password parts to the appropriate values. If we don't want to specify a password (not recommended!) then we can drop the `IDENTIFIED BY 'password'` part. This and all SQL statements that we input into MariaDB need to end with a semicolon (`;`).

The host part can be several things. It can be the hostname of the computer the user connects from, the IP address of the computer the user connects from, the network the user connects from, or it can be the wildcard symbol `%`, which means any host.

For More Information:
Here are some examples. This first example user can login from anywhere because of the wildcard character, %, in the host part. The user's password is: bomber.

`CREATE USER 'boyd'@'%' IDENTIFIED BY 'bomber';`

The following three examples demonstrate using various host names. The first specifies the localhost on which means the local server MariaDB is running. The next specifies a single host. The third uses % to specify any subdomain of the example.net domain.

`CREATE USER 'tom'@'localhost' IDENTIFIED BY 'retail';`
`CREATE USER 'richard'@'powr.example.net' IDENTIFIED BY 'nuclear';`
`CREATE USER 'robert'@'%.example.net' IDENTIFIED BY 'pilot';`

Instead of hostnames we can also use IP addresses as shown in the following three examples. The first has an exact IP address identifying a single computer. The second uses a % sign in the last quad of the IP address so any computer where the first three sets of numbers in the IP address match will be able to connect. The third uses a subnet mask, but the end result (in this example at least) is the same as the second.

`CREATE USER 'dallin'@'192.168.1.1' IDENTIFIED BY 'judge';`
`CREATE USER 'russell'@'192.168.1.%' IDENTIFIED BY 'surgeon';`
`CREATE USER 'russell'@'192.168.1.0/255.255.255.0' IDENTIFIED BY 'business';`

One benefit to using IP addresses instead of domain names is that no name resolution or domain validation needs to be made. Such system calls to lookup and check the validity of domains can be costly and might take time and resources better spent on other things. To enforce a no domain names policy, add `skip-name-resolv=1` to the [mysqld] section of the my.cnf or my.ini file.

Complete documentation of the `CREATE USER` statement is available at https://mariadb.com/kb/en/create-user/.

**Granting permissions**

By default, new users do not have permission to do anything except logging in, which is not very useful. So the next thing we need to do is give them the permissions they need. This is done using the `GRANT` statement. Using this statement, we will be able to `GRANT` users the appropriate permissions. The `GRANT` statements have the following basic pattern:

`GRANT <privileges> ON <database> TO <user>;`

---

**For More Information:**

We customize the `<privileges>`, `<database>`, and `<user>` parts as needed. The `<user>` section should match the `'username'@'host'` part of the `CREATE` statement, otherwise, we'll be creating what is essentially a new user. We can also add an `IDENTIFIED BY 'password'` section to the end of the `GRANT` statement if we want to change the password (or add a password to an account that doesn't have one).

Here are some examples. This first one grants all privileges including the grant option on all databases and the user can log in from anywhere. We should not often set up users with such broad authority, and when we do we need to make sure, we use an appropriate `CREATE USER` statement first and assign the user a password (or assign the password here). If the user doesn't exist, the `GRANT` statement will create one, but if the user doesn't exist and our `GRANT` statement doesn't include an `IDENTIFIED BY 'password'` section then the user will be created without a password, so it's a good habit to first create the user with a password, and then grant the user the rights they need.

```
GRANT ALL ON *.* TO 'robert'@'%' WITH GRANT OPTION;
```

The following example is a standard set of permissions for a regular user who needs read and write access to a database called `serv`. If a user just needs read access, we can just assign the user the `SELECT` privilege. By specifying `serv.*` as the database, the user only has these rights on tables in the `serv` database. Multiple privileges are separated by commas.

```
GRANT SELECT,INSERT,UPDATE,DELETE ON serv.* TO 'jeffrey'@'localhost';
```

This following user has read access (SELECT) to just the staff table in the `edu` database, and the user has the `GRANT OPTION` privilege so they can grant that same right to other users.

```
GRANT SELECT ON edu.staff TO 'david'@'localhost' WITH GRANT OPTION;
```

The following example gives a user all rights on the `logan` database. We'll also limit this user to 100 queries per hour, just because we can.

```
GRANT ALL ON logan.* TO 'quentin'@'localhost' WITH MAX QUERIES PER HOUR 100;
```

Complete documentation of the `GRANT` statement is available at https://mariadb.com/kb/en/grant/.
Adding and removing privileges
Sometimes it becomes necessary to remove a privilege or two from a user, or to give them more privileges. Giving additional privileges is easy, just run an additional GRANT statement with the new rights and they will be added. To remove privileges, we use the REVOKE statement. It has the following pattern:

REVOKE <privileges> ON <database> FROM <user>;

To remove a GRANT OPTION privilege, specify it in the privileges section. The following example removes the DELETE and GRANT OPTION permissions from the todd user:

REVOKE DELETE,GRANT OPTION ON cust.* FROM 'todd'@'%';

To remove all privileges from a user ('neil'@'%.example.com' in this example), we use the following special command:

REVOKE ALL,GRANT OPTION FROM 'neil'@'%.example.com';

We, of course, need to customize the user part to match the user for whom we are removing privileges. The previous statement is special in that it must be used as written even if the user doesn't have the GRANT OPTION privilege. If we remove the GRANT OPTION privilege from it the statement won't run.

Complete documentation of the REVOKE statement is available at: https://mariadb.com/kb/en/revoke/.

Showing grants
To show the grants available for a user, we use the SHOW GRANTS command. It has the following pattern:

SHOW GRANTS FOR <user>;

All we have to do is customize the <user> part with the information of the user we want to look at. Here is an example:

SHOW GRANTS FOR 'dieter'@'10.2.200.4';

For More Information:
MariaDB User Account Management

The output of the `SHOW GRANTS` command is a `GRANT` statement that encapsulates all of the user's privileges. This is useful if you want to give another user the exact same privileges. For example, the output of the preceding `SHOW GRANTS` command might be as follows:

```
+------------------------------------------------------+
| Grants for dieter@10.2.200.4                         |
+------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'dieter'@'10.2.200.4' |
+------------------------------------------------------+
```

Changing passwords

To change the password of a user, we use the `SET PASSWORD` statement. It has the following pattern:

```
SET PASSWORD FOR <user> = PASSWORD('<password>');
```

Here is an example:

```
SET PASSWORD FOR 'henry'@'%' = PASSWORD('niftypassword');
```

Complete documentation of the `SET PASSWORD` statement is available at https://mariadb.com/kb/en/set-password/.

Removing users

To remove a user completely, we use the `DROP USER` statement. It has the following pattern:

```
DROP USER <user>;
```

Here is an example:

```
DROP USER 'tom'@'';
```

When a user is dropped, all grants are automatically removed.

Complete documentation of the `DROP USER` statement is available at https://mariadb.com/kb/en/drop-user/.
Summary

In this chapter, we learned about adding and removing users and how to give those users the permissions they need or take them away as needed. Up till now, we’ve only talked about things related to our databases (securing them, managing users, and so on). We haven't actually done anything with the actual data—you know the stuff that databases are good at storing and retrieving. Well, the time to talk about it is now. In the next chapter, *Using MariaDB*, we'll learn some basic SQL commands that we can use with the `mysql` command-line client program to create databases, insert data, read data, and so on.
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.