


[Blog](#) / [Databases](#)

How to: MySQL Create User and Grant Permissions



Limiting access privileges and managing user credentials in MySQL requires a great deal of repetitive and manual effort, especially if you have dozens (or even hundreds) of MySQL instances across multiple servers. Admins need ways to manage MySQL database access with consistency and efficiency.

In this tutorial, we'll walk through the most common MySQL commands and management techniques, including:

- How to **create users** within a MySQL database

- How to **grant and revoke user permissions** within the database management system and to the underlying data
- How to use a centralized interface to provision user access in MySQL

This tutorial focuses on a self-managed MySQL environment where you have root level database access. While many of the commands below may work in a hosted environment (such as AWS RDS or Google's Cloud SQL), there may be restrictions on the commands you can run and how you manage user access in such an environment.

Create MySQL databases and users

Once you have MySQL installed on the server(s) that will host your MySQL environment, you need to create a database and additional user accounts. In order to run the following commands, log into the MySQL instance with the MySQL root account.

Create a MySQL database

To create a database called ``strongdm``, type the following into the MySQL command line:

```
mysql> CREATE DATABASE strongdm;
```

If the database does not already have a unique name for the MySQL instance, MySQL will issue an error message with error code 1007. Add IF NOT EXISTS to the command to prevent this error with the code below:

```
mysql> CREATE DATABASE IF NOT EXISTS strongdm;
```

Delete a MySQL database

To delete a database called ``strongdm``, type the following command into your MySQL command line:

```
mysql> DROP DATABASE strongdm;
```



by [Schuyler Brown](#)Co-Founder / CCO
strongDMLast updated on:
April 14, 2022[Download the PAM eBook PDF](#)

Found in:

[Databases Security](#)**strongDM manages
and audits access to
infrastructure.**Role-based, attribute-based,
& just-in-time access to
infrastructureConnect any person or service
to any infrastructure, anywhere

Logging like you've never seen

**Get a
demo**

Note: This command will **permanently** delete your database and its associated data. The MySQL command line interface will not prompt you to confirm the action, so use this command with care.

Create a new MySQL user account

MySQL defines users with a username and the hostname or IP address that they're using to access the MySQL instance. To create a new user in MySQL, specify the username, the hostname the user can use to access the database management system, and a secure password:

```
1> CREATE USER 'local_user'@'localhost' IDENTIFIED B
```



This command will allow the user with username *local_user* to access the MySQL instance from the local machine (localhost) and prevent the user from accessing it directly from any other machine. Alternatively, you can use a wildcard character (%) in the host definition to grant access to the MySQL instance for a user:

```
mysql> CREATE USER 'subnet_user'@'10.0.0%' IDENTIF
```



In the above example, the `'10.0.0%'` specifies that the user can access the MySQL instance from any client that has an IP address beginning with `'10.0.'`. You may use the wild card at any level of the IP address in the host definition.

To view all users in your MySQL instance, use the SELECT command:

```
mysql> SELECT * FROM mysql.user;
```

MySQL user account management

Without privileges, a newly created user account can connect to the MySQL instance but cannot access any data or perform any actions. Let's look at MySQL privileges more closely:

Understanding privileges in MySQL

In MySQL, a **privilege** is a right to perform an action on a database that must be **granted** to users. This effectively defines the access level that a user has on a database and what they can do within it. We can organize these privileges by scope into levels:



- **Global privileges** apply to all databases on the server. Administrative privileges fall into the global group because they enable a user to manage operations of the MySQL server and aren't specific to a particular database.
- **Database privileges** apply to specific databases in your MySQL instance and all of the objects within those databases (e.g. tables, columns, and views). You can also grant database privileges globally.
- **Proxy privileges** allow a user to act as if they have the privileges granted to another user.
- **Privileges for database objects** (tables, columns, stored routines, views, etc.) can apply to all objects of one type within a particular database or to specific objects, such as a certain table or view. You can also grant database object privileges globally.

Information about MySQL privileges are stored in grant tables in the ``mysql`` database within your MySQL instance, as follows:

Table	Grant Information
<code>user</code>	global privileges
<code>db</code>	database-level privileges
<code>tables_priv</code>	table-level privileges
<code>columns_priv</code>	column-level privileges
<code>procs_priv</code>	stored procedure and function privileges
<code>proxies_priv</code>	proxy-user privileges

Some **common privileges** include:

- ``ALL PRIVILEGES``: The user is granted all privileges except GRANT OPTION and PROXY.
- ``ALTER``: The user can change the structure of a table or database.
- ``CREATE``: The user can create new databases and tables.
- ``DELETE``: The user can delete rows in a table.
- ``INSERT``: The user can add rows to a table.
- ``SELECT``: The user can read rows from a table.
- ``UPDATE``: The user can update rows in a table.

Grant permissions to a MySQL user account

The GRANT statement allows you to set MySQL access permissions, using the following syntax:

```
mysql> GRANT privilege ON privilege_level TO acco
```



Type the following to grant ``SELECT`` and ``INSERT`` privileges to a local user on the ``strongdm`` database:

```
mysql> GRANT SELECT, INSERT ON strongdm.* TO 'loc
```



To create a user with the same privileges as the root user, use the following command, which grants global privileges to the user Janet connecting via localhost:

```
mysql> GRANT ALL ON *.* TO 'janet'@'localhost' WI
```



The WITH GRANT OPTION clause allows the user to grant the privileges they have to other users.

Revoke permissions from a MySQL user account

To remove privileges, use the REVOKE command, which uses syntax similar to the GRANT command. For example, if you wanted to revoke `SELECT` and `INSERT` privileges from a local user on the `strongdm` database, type the following:

```
mysql> REVOKE SELECT, INSERT ON strongdm.* FROM ‘
```



If the user does not actually have the privilege in question, this command will not affect any of their privileges.


Change a MySQL user account password

The syntax to change a user password depends on your version of MySQL. To find out the MySQL version you are running, use the command:

```
mysql> SELECT version();
```


To change a password for MySQL 5.76 or higher, use this command:

```
mysql> ALTER USER 'local_user'@'localhost' IDENTI
```



For older versions of MySQL, use this command instead:

```
mysql> SET PASSWORD FOR 'local_user'@'localhost' =
```



Delete MySQL users

To delete a MySQL user, use the DROP command:

```
mysql> DROP USER 'local_user'@'localhost';
```

Display MySQL user account privileges

To view the privileges of a MySQL user, use the `SHOW GRANTS` command:

```
mysql> SHOW GRANTS FOR 'local_user'@'localhost';
```

Determining if privileges are correct

Occasionally, admins need to review user access to different databases, tables, views, or columns. In addition to checking which grants a user has, you can also look at the privileges that are set on a [specific table](#) or column.

For example, to check the privileges granted on a table named `my_table`, use the following command:

```
mysql> SELECT * FROM 'INFORMATION_SCHEMA'.'TABLE_
```



Similarly, to look at the column privileges:

```
mysql> SELECT * FROM 'INFORMATION_SCHEMA'.'COLUMN
```



Use the SHOW GRANTS command to display all privileges granted to a specific user. If you don't specify a user, the

privileges for the current user will be displayed.

```
mysql> SHOW GRANTS;
```

For a specific user, use:

```
mysql> SHOW GRANTS FOR 'local_user'@'localhost';
```

This command is useful for database auditing. For example, an admin could use it to audit if a user has access to more objects than they should.

In order to show permissions for the user via any host, not just localhost, you'll need to run a few commands:

```
mysql> SELECT CONCAT('SHOW GRANTS FOR'',user, ''
```



This will display all of the SHOW GRANTS commands the user can run, one for each host the user has permission to access the database from. Copy each command and run it to see all the privileges the user has in the MySQL instance.

Effectively managing your MySQL databases and user accounts

Once you set up databases, users, and permissions, consider what the daily management of your MySQL databases, user accounts, and privileges look like. Developers, business users, contractors, vendors, and more all need access. How will you manage database credentials as the infrastructure grows? How will you ensure each user has granular access and only performs certain tasks and nothing more?

A control plane simplifies provisioning access to MySQL databases and reduces strain on admins in several ways:

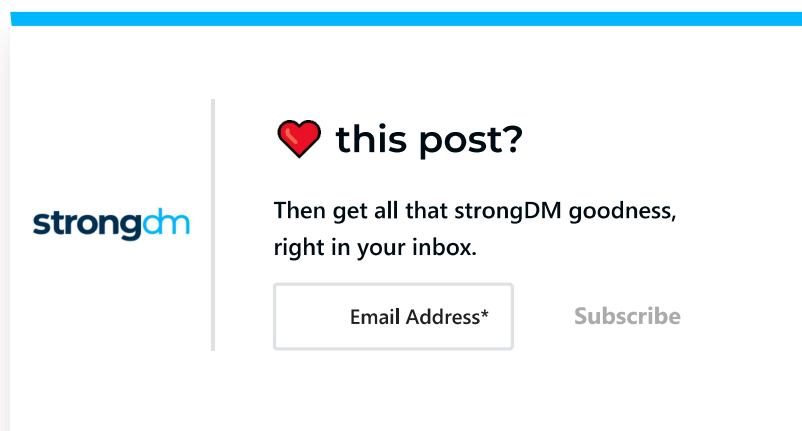
- **Centralized authentication for everyone in the organization.** Regardless of the user type or destination database, strongDM eliminates the need to manually manage authentication credentials for individual users. strongDM integrates directly with your identity provider using OIDC protocols to handle authentication.
- **View and modify access privileges from one place.** The strongDM admin interface provides a role-based system for managing permissions. In the Admin UI, simply drag and drop users to the appropriate roles and their permissions automatically update. Infrastructure administrators can onboard users, view and modify permissions, and revoke access with ease.
- **Auditing of all actions against all hosts.** The strongDM proxy logs all actions — queries, authentications requests, changes to permissions, and more — so you can easily understand who did what, when, and in what order. You can also configure log storage to your specifications with options for encryption, streaming to any log aggregator, and retaining logs locally.

Setting up a control plane for your MySQL databases

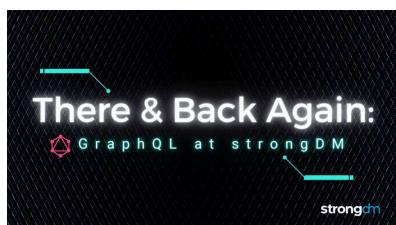
The manual work associated with MySQL account management frustrates many admins. strongDM automates and consolidates access control so that MySQL administrators can efficiently manage user privileges through a control plane. strongDM integrates with any Single Sign-On (SSO) provider and connects to your infrastructure wherever it resides, whether on-prem or in the cloud.

[Try strongDM](#) to see how a centralized control plane can help you efficiently and securely manage your MySQL databases.

To learn more on how strongDM helps companies with managing permissions, make sure to check out our [Managing Permissions Use Case](#).



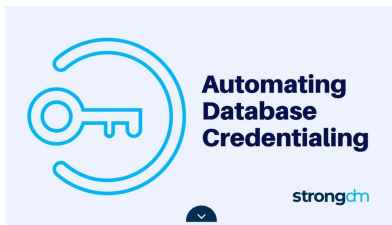
You May Also Like



There and Back Again: GraphQL at strongDM

Our incredible journey from bespoke REST endpoints to GraphQL.

Automating Database Credentialing



Database sprawl is a lot like expanding into the suburbs: your house may be empty at first, but before you know it, you're having to stuff things into your attic.



What is the difference between Proxy and Reverse Proxy? | How To Use Reverse Proxy for Access Management Control

In this post, we'll dissect the two concepts and explain how administrators can use a reverse proxy for easy access management control.



Integrate Active Directory With Any Database or Single Sign-On

Consider this when you choose to integrate Active Directory (AD) with your databases and applications using their native APIs, connectors, or toolkits.



Managing Access to Ephemeral Infrastructure At Scale

Managing a static fleet of strongDM servers is dead simple. You create the server in the strongDM console, place the public key file on the box, and it's done! This scales really well for small deployments, but as your fleet grows, the burden of manual tasks grows with it.



PRODUCT

- Infrastructure Access Platform
- Use Cases
- How It Works
- We ❤️ Your Stack
- Pricing
- Customer Stories
- Compare

DOCS

- Docs Home
- User Guide
- Admin Guide
- API

RESOURCES

- Blog
- Articles
- Videos
- Webinars
- Events
- Podcasts
- Comply

COMPANY

- About Us
- Careers
- Security
- Press

GET STARTED

- Try It Free
- Chat with Us
- Schedule a Demo

