**Безпека у MySQL**

**Table of Contents**

[General Security Issues 4](#_Toc58439515)

[Security Guidelines 4](#_Toc58439516)

[Keeping Passwords Secure 6](#_Toc58439517)

[End-User Guidelines for Password Security 6](#_Toc58439518)

[Administrator Guidelines for Password Security 7](#_Toc58439519)

[Passwords and Logging 7](#_Toc58439520)

[Making MySQL Secure Against Attackers 8](#_Toc58439521)

[Security-Related mysqld Options and Variables 10](#_Toc58439522)

[How to Run MySQL as a Normal User 10](#_Toc58439523)

[Security Considerations for LOAD DATA LOCAL 11](#_Toc58439524)

[Enabling or Disabling Local Data Loading Capability 11](#_Toc58439525)

[Restricting Files Permitted for Local Data Loading 12](#_Toc58439526)

[Client Programming Security Guidelines 13](#_Toc58439527)

[Access Control and Account Management 14](#_Toc58439528)

[Account User Names and Passwords 15](#_Toc58439529)

[Privileges Provided by MySQL 16](#_Toc58439530)

[Summary of Available Privileges 17](#_Toc58439531)

[Static Privilege Descriptions 20](#_Toc58439532)

[Dynamic Privilege Descriptions 25](#_Toc58439533)

[Privilege-Granting Guidelines 29](#_Toc58439534)

[Static Versus Dynamic Privileges 29](#_Toc58439535)

[Migrating Accounts from SUPER to Dynamic Privileges 30](#_Toc58439536)

[Grant Tables 31](#_Toc58439537)

[Grant Table Overview 31](#_Toc58439538)

[The user and db Grant Tables 33](#_Toc58439539)

[The tables\_priv and columns\_priv Grant Tables 36](#_Toc58439540)

[The procs\_priv Grant Table 36](#_Toc58439541)

[The proxies\_priv Grant Table 36](#_Toc58439542)

[The global\_grants Grant Table 37](#_Toc58439543)

[The default\_roles Grant Table 37](#_Toc58439544)

[The role\_edges Grant Table 37](#_Toc58439545)

[The password\_history Grant Table 37](#_Toc58439546)

[Grant Table Scope Column Properties 37](#_Toc58439547)

[Grant Table Privilege Column Properties 38](#_Toc58439548)

[Grant Table Concurrency 38](#_Toc58439549)

[Specifying Account Names 39](#_Toc58439550)

[Specifying Role Names 40](#_Toc58439551)

[Access Control, Stage 1: Connection Verification 41](#_Toc58439552)

[Access Control, Stage 2: Request Verification 44](#_Toc58439553)

[Adding Accounts, Assigning Privileges, and Dropping Accounts 45](#_Toc58439554)

[Creating Accounts and Granting Privileges 46](#_Toc58439555)

[Checking Account Privileges and Properties 47](#_Toc58439556)

[Revoking Account Privileges 47](#_Toc58439557)

[Dropping Accounts 48](#_Toc58439558)

[Reserved Accounts 48](#_Toc58439559)

[Using Roles 48](#_Toc58439560)

[Creating Roles and Granting Privileges to Them 49](#_Toc58439561)

[Defining Mandatory Roles 50](#_Toc58439562)

[Checking Role Privileges 50](#_Toc58439563)

[Activating Roles 51](#_Toc58439564)

[Revoking Roles or Role Privileges 52](#_Toc58439565)

[Dropping Roles 53](#_Toc58439566)

[User and Role Interchangeability 53](#_Toc58439567)

[Account Categories 54](#_Toc58439568)

[System and Regular Accounts 54](#_Toc58439569)

[Operations Affected by the SYSTEM\_USER Privilege 55](#_Toc58439570)

[System and Regular Sessions 55](#_Toc58439571)

[Protecting System Accounts Against Manipulation by Regular Accounts 56](#_Toc58439572)

[Privilege Restriction Using Partial Revokes 57](#_Toc58439573)

[Using Partial Revokes 57](#_Toc58439574)

[Partial Revokes Versus Explicit Schema Grants 61](#_Toc58439575)

[Disabling Partial Revokes 62](#_Toc58439576)

[Partial Revokes and Replication 62](#_Toc58439577)

[When Privilege Changes Take Effect 63](#_Toc58439578)

[Assigning Account Passwords 63](#_Toc58439579)

[Password Management 64](#_Toc58439580)

[Internal Versus External Credentials Storage 65](#_Toc58439581)

[Password Expiration Policy 65](#_Toc58439582)

[Password Reuse Policy 67](#_Toc58439583)

[Password Verification-Required Policy 68](#_Toc58439584)

[Dual Password Support 71](#_Toc58439585)

[Random Password Generation 72](#_Toc58439586)

[Failed-Login Tracking and Temporary Account Locking 73](#_Toc58439587)

[Server Handling of Expired Passwords 74](#_Toc58439588)

[Pluggable Authentication 76](#_Toc58439589)

[Available Authentication Plugins 76](#_Toc58439590)

[Authentication Plugin Usage 77](#_Toc58439591)

[Authentication Plugin Client/Server Compatibility 78](#_Toc58439592)

[Authentication Plugin Connector-Writing Considerations 78](#_Toc58439593)

[Restrictions on Pluggable Authentication 79](#_Toc58439594)

[General Pluggable Authentication Restrictions 79](#_Toc58439595)

[Pluggable Authentication and Third-Party Connectors 79](#_Toc58439596)

[Proxy Users 80](#_Toc58439597)

[Requirements for Proxy User Support 80](#_Toc58439598)

[Simple Proxy User Example 81](#_Toc58439599)

[Preventing Direct Login to Proxied Accounts 82](#_Toc58439600)

[Granting and Revoking the PROXY Privilege 82](#_Toc58439601)

[Default Proxy Users 83](#_Toc58439602)

[Default Proxy User and Anonymous User Conflicts 84](#_Toc58439603)

[Server Support for Proxy User Mapping 85](#_Toc58439604)

[Proxy User System Variables 86](#_Toc58439605)

[Account Locking 87](#_Toc58439606)

[Setting Account Resource Limits 87](#_Toc58439607)

[Troubleshooting Problems Connecting to MySQL 89](#_Toc58439608)

[Based Account Activity Auditing 93](#_Toc58439609)

[Using Encrypted Connections 94](#_Toc58439610)

[Configuring MySQL to Use Encrypted Connections 95](#_Toc58439611)

[Server-Side Startup Configuration for Encrypted Connections 95](#_Toc58439612)

[Server-Side Runtime Configuration and Monitoring for Encrypted Connections 96](#_Toc58439613)

[Client-Side Configuration for Encrypted Connections 98](#_Toc58439614)

[Configuring Encrypted Connections as Mandatory 100](#_Toc58439615)

[Encrypted Connection TLS Protocols and Ciphers 101](#_Toc58439616)

[Supported Connection TLS Protocols 101](#_Toc58439617)

[Connection TLS Protocol Configuration 102](#_Toc58439618)

[Connection Cipher Configuration 103](#_Toc58439619)

[Connection TLS Protocol Negotiation 106](#_Toc58439620)

[Monitoring Current Client Session TLS Protocol and Cipher 107](#_Toc58439621)

[Creating SSL and RSA Certificates and Keys 107](#_Toc58439622)

[Creating SSL and RSA Certificates and Keys using MySQL 107](#_Toc58439623)

[Automatic SSL and RSA File Generation 107](#_Toc58439624)

[Manual SSL and RSA File Generation Using mysql\_ssl\_rsa\_setup 108](#_Toc58439625)

[SSL and RSA File Characteristics 108](#_Toc58439626)

[Creating SSL Certificates and Keys Using openssl 109](#_Toc58439627)

[Example 1: Creating SSL Files from the Command Line on Unix 110](#_Toc58439628)

[Example 2: Creating SSL Files Using a Script on Unix 110](#_Toc58439629)

[Example 3: Creating SSL Files on Windows 113](#_Toc58439630)

[Creating RSA Keys Using openssl 115](#_Toc58439631)

[Connecting to MySQL Remotely from Windows with SSH 115](#_Toc58439632)

When thinking about security within a MySQL installation, you should consider a wide range of possible topics and how they affect the security of your MySQL server and related applications:

* General factors that affect security. These include choosing good passwords, not granting unnecessary privileges to users, ensuring application security by preventing SQL injections and data corruption, and others. See [Section , “General Security Issues”](https://dev.mysql.com/doc/refman/8.0/en/general-security-issues.html).
* Security of the installation itself. The data files, log files, and the all the application files of your installation should be protected to ensure that they are not readable or writable by unauthorized parties. For more information, see [Section , “Postinstallation Setup and Testing”](https://dev.mysql.com/doc/refman/8.0/en/postinstallation.html).
* Access control and security within the database system itself, including the users and databases granted with access to the databases, views and stored programs in use within the database. For more information, see [Section , “Access Control and Account Management”](https://dev.mysql.com/doc/refman/8.0/en/access-control.html).
* The features offered by security-related plugins. See [Section 6.4, “Security Components and Plugins”](https://dev.mysql.com/doc/refman/8.0/en/security-plugins.html).
* Network security of MySQL and your system. The security is related to the grants for individual users, but you may also wish to restrict MySQL so that it is available only locally on the MySQL server host, or to a limited set of other hosts.
* Ensure that you have adequate and appropriate backups of your database files, configuration and log files. Also be sure that you have a recovery solution in place and test that you are able to successfully recover the information from your backups. See [, *Backup and Recovery*](https://dev.mysql.com/doc/refman/8.0/en/backup-and-recovery.html).

## General Security Issues

This section describes general security issues to be aware of and what you can do to make your MySQL installation more secure against attack or misuse. For information specifically about the access control system that MySQL uses for setting up user accounts and checking database access, see [Section , “Postinstallation Setup and Testing”](https://dev.mysql.com/doc/refman/8.0/en/postinstallation.html).

### Security Guidelines

Anyone using MySQL on a computer connected to the Internet should read this section to avoid the most common security mistakes.

In discussing security, it is necessary to consider fully protecting the entire server host (not just the MySQL server) against all types of applicable attacks: eavesdropping, altering, playback, and denial of service. We do not cover all aspects of availability and fault tolerance here.

MySQL uses security based on Access Control Lists (ACLs) for all connections, queries, and other operations that users can attempt to perform. There is also support for SSL-encrypted connections between MySQL clients and servers. Many of the concepts discussed here are not specific to MySQL at all; the same general ideas apply to almost all applications.

When running MySQL, follow these guidelines:

* ***Do not ever give anyone (except MySQL****root****accounts) access to the****user****table in the****mysql****system database!*** This is critical.
* Learn how the MySQL access privilege system works (see [Section 6.2, “Access Control and Account Management”](https://dev.mysql.com/doc/refman/8.0/en/access-control.html)). Use the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements to control access to MySQL. Do not grant more privileges than necessary. Never grant privileges to all hosts.

Checklist:

* + Try mysql -u root. If you are able to connect successfully to the server without being asked for a password, anyone can connect to your MySQL server as the MySQL root user with full privileges! Review the MySQL installation instructions, paying particular attention to the information about setting a root password. See [Section 2.10.4, “Securing the Initial MySQL Account”](https://dev.mysql.com/doc/refman/8.0/en/default-privileges.html).
  + Use the [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) statement to check which accounts have access to what. Then use the [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statement to remove those privileges that are not necessary.
* Do not store cleartext passwords in your database. If your computer becomes compromised, the intruder can take the full list of passwords and use them. Instead, use [SHA2()](https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html#function_sha2) or some other one-way hashing function and store the hash value.

To prevent password recovery using rainbow tables, do not use these functions on a plain password; instead, choose some string to be used as a salt, and use hash(hash(password)+salt) values.

* Do not choose passwords from dictionaries. Special programs exist to break passwords. Even passwords like “xfish98” are very bad. Much better is “duag98” which contains the same word “fish” but typed one key to the left on a standard QWERTY keyboard. Another method is to use a password that is taken from the first characters of each word in a sentence (for example, “Four score and seven years ago” results in a password of “Fsasya”). The password is easy to remember and type, but difficult to guess for someone who does not know the sentence. In this case, you can additionally substitute digits for the number words to obtain the phrase “4 score and 7 years ago”, yielding the password “4sa7ya” which is even more difficult to guess.
* Invest in a firewall. This protects you from at least 50% of all types of exploits in any software. Put MySQL behind the firewall or in a demilitarized zone (DMZ).

Checklist:

* + Try to scan your ports from the Internet using a tool such as nmap. MySQL uses port 3306 by default. This port should not be accessible from untrusted hosts. As a simple way to check whether your MySQL port is open, try the following command from some remote machine, where ***server\_host*** is the host name or IP address of the host on which your MySQL server runs:

shell> telnet server\_host 3306

If **telnet** hangs or the connection is refused, the port is blocked, which is how you want it to be. If you get a connection and some garbage characters, the port is open, and should be closed on your firewall or router, unless you really have a good reason to keep it open.

* Applications that access MySQL should not trust any data entered by users, and should be written using proper defensive programming techniques. See [Section 6.1.7, “Client Programming Security Guidelines”](https://dev.mysql.com/doc/refman/8.0/en/secure-client-programming.html).
* Do not transmit plain (unencrypted) data over the Internet. This information is accessible to everyone who has the time and ability to intercept it and use it for their own purposes. Instead, use an encrypted protocol such as SSL or SSH. MySQL supports internal SSL connections. Another technique is to use SSH port-forwarding to create an encrypted (and compressed) tunnel for the communication.
* Learn to use the **tcpdump** and **strings** utilities. In most cases, you can check whether MySQL data streams are unencrypted by issuing a command like the following:

shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings

This works under Linux and should work with small modifications under other systems.

**Warning**

If you do not see cleartext data, this does not always mean that the information actually is encrypted. If you need high security, consult with a security expert.

### Keeping Passwords Secure

Passwords occur in several contexts within MySQL. The following sections provide guidelines that enable end users and administrators to keep these passwords secure and avoid exposing them. In addition, the validate\_password plugin can be used to enforce a policy on acceptable password. See [Section 6.4.3, “The Password Validation Component”](https://dev.mysql.com/doc/refman/8.0/en/validate-password.html).

#### End-User Guidelines for Password Security

MySQL users should use the following guidelines to keep passwords secure.

When you run a client program to connect to the MySQL server, it is inadvisable to specify your password in a way that exposes it to discovery by other users. The methods you can use to specify your password when you run client programs are listed here, along with an assessment of the risks of each method. In short, the safest methods are to have the client program prompt for the password or to specify the password in a properly protected option file.

* Use the [**mysql\_config\_editor**](https://dev.mysql.com/doc/refman/8.0/en/mysql-config-editor.html) utility, which enables you to store authentication credentials in an encrypted login path file named .mylogin.cnf. The file can be read later by MySQL client programs to obtain authentication credentials for connecting to MySQL Server. See [Section 4.6.7, “**mysql\_config\_editor** — MySQL Configuration Utility”](https://dev.mysql.com/doc/refman/8.0/en/mysql-config-editor.html).
* Use a --password=***password*** or -p***password*** option on the command line. For example:

shell> mysql -u francis -pfrank db\_name

**Warning**

This is convenient but insecure. On some systems, your password becomes visible to system status programs such as **ps** that may be invoked by other users to display command lines. MySQL clients typically overwrite the command-line password argument with zeros during their initialization sequence. However, there is still a brief interval during which the value is visible. Also, on some systems this overwriting strategy is ineffective and the password remains visible to **ps**. (SystemV Unix systems and perhaps others are subject to this problem.)

If your operating environment is set up to display your current command in the title bar of your terminal window, the password remains visible as long as the command is running, even if the command has scrolled out of view in the window content area.

* Use the [--password](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password) or -p option on the command line with no password value specified. In this case, the client program solicits the password interactively:
* shell> mysql -u francis -p db\_name

Enter password: \*\*\*\*\*\*\*\*

The \* characters indicate where you enter your password. The password is not displayed as you enter it.

It is more secure to enter your password this way than to specify it on the command line because it is not visible to other users. However, this method of entering a password is suitable only for programs that you run interactively. If you want to invoke a client from a script that runs noninteractively, there is no opportunity to enter the password from the keyboard. On some systems, you may even find that the first line of your script is read and interpreted (incorrectly) as your password.

* Store your password in an option file. For example, on Unix, you can list your password in the [client] section of the .my.cnf file in your home directory:
* [client]

password=password

To keep the password safe, the file should not be accessible to anyone but yourself. To ensure this, set the file access mode to 400 or 600. For example:

shell> chmod 600 .my.cnf

To name from the command line a specific option file containing the password, use the [--defaults-file=***file\_name***](https://dev.mysql.com/doc/refman/8.0/en/option-file-options.html#option_general_defaults-file) option, where file\_name is the full path name to the file. For example:

shell> mysql --defaults-file=/home/francis/mysql-opts

[Section 4.2.2.2, “Using Option Files”](https://dev.mysql.com/doc/refman/8.0/en/option-files.html), discusses option files in more detail.

On Unix, the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client writes a record of executed statements to a history file (see [Section 4.5.1.3, “mysql Client Logging”](https://dev.mysql.com/doc/refman/8.0/en/mysql-logging.html)). By default, this file is named .mysql\_history and is created in your home directory. Passwords can be written as plain text in SQL statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), so if you use these statements, they are logged in the history file. To keep this file safe, use a restrictive access mode, the same way as described earlier for the .my.cnf file.

If your command interpreter maintains a history, any file in which the commands are saved contains MySQL passwords entered on the command line. For example, **bash** uses ~/.bash\_history. Any such file should have a restrictive access mode.

#### Administrator Guidelines for Password Security

Database administrators should use the following guidelines to keep passwords secure.

MySQL stores passwords for user accounts in the mysql.user system table. Access to this table should never be granted to any nonadministrative accounts.

Account passwords can be expired so that users must reset them. See [Section 6.2.15, “Password Management”](https://dev.mysql.com/doc/refman/8.0/en/password-management.html), and [Section 6.2.16, “Server Handling of Expired Passwords”](https://dev.mysql.com/doc/refman/8.0/en/expired-password-handling.html).

The validate\_password plugin can be used to enforce a policy on acceptable password. See [Section 6.4.3, “The Password Validation Component”](https://dev.mysql.com/doc/refman/8.0/en/validate-password.html).

A user who has access to modify the plugin directory (the value of the [plugin\_dir](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_plugin_dir) system variable) or the my.cnf file that specifies the plugin directory location can replace plugins and modify the capabilities provided by plugins, including authentication plugins.

Files such as log files to which passwords might be written should be protected. See [Section 6.1.2.3, “Passwords and Logging”](https://dev.mysql.com/doc/refman/8.0/en/password-logging.html).

#### Passwords and Logging

Passwords can be written as plain text in SQL statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html). If such statements are logged by the MySQL server as written, passwords in them become visible to anyone with access to the logs.

Statement logging avoids writing passwords as cleartext for the following statements:

CREATE USER ... IDENTIFIED BY ...

ALTER USER ... IDENTIFIED BY ...

SET PASSWORD ...

START SLAVE ... PASSWORD = ...

START REPLICA ... PASSWORD = ...

CREATE SERVER ... OPTIONS(... PASSWORD ...)

ALTER SERVER ... OPTIONS(... PASSWORD ...)

Passwords in those statements are rewritten to not appear literally in statement text written to the general query log, slow query log, and binary log. Rewriting does not apply to other statements. In particular, [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html) or [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html) statements for the mysql.user system table that refer to literal passwords are logged as is, so you should avoid such statements. (Direct modification of grant tables is discouraged, anyway.)

For the general query log, password rewriting can be suppressed by starting the server with the [--log-raw](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_log-raw) option. For security reasons, this option is not recommended for production use. For diagnostic purposes, it may be useful to see the exact text of statements as received by the server.

By default, contents of audit log files produced by the audit log plugin are not encrypted and may contain sensitive information, such as the text of SQL statements. For security reasons, audit log files should be written to a directory accessible only to the MySQL server and to users with a legitimate reason to view the log. See [Section 6.4.5.3, “MySQL Enterprise Audit Security Considerations”](https://dev.mysql.com/doc/refman/8.0/en/audit-log-security.html).

Statements received by the server may be rewritten if a query rewrite plugin is installed (see [Query Rewrite Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-types.html#query-rewrite-plugin-type)). In this case, the [--log-raw](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_log-raw) option affects statement logging as follows:

* Without [--log-raw](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_log-raw), the server logs the statement returned by the query rewrite plugin. This may differ from the statement as received.
* With [--log-raw](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_log-raw), the server logs the original statement as received.

An implication of password rewriting is that statements that cannot be parsed (due, for example, to syntax errors) are not written to the general query log because they cannot be known to be password free. Use cases that require logging of all statements including those with errors should use the [--log-raw](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_log-raw) option, bearing in mind that this also bypasses password rewriting.

Password rewriting occurs only when plain text passwords are expected. For statements with syntax that expect a password hash value, no rewriting occurs. If a plain text password is supplied erroneously for such syntax, the password is logged as given, without rewriting.

To guard log files against unwarranted exposure, locate them in a directory that restricts access to the server and the database administrator. If the server logs to tables in the mysql database, grant access to those tables only to the database administrator.

Replicas store the password for the replication source server in their connection metadata repository, which by default is a table in the mysql database named slave\_master\_info. The use of a file in the data directory for the connection metadata repository is now deprecated, but still possible (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](https://dev.mysql.com/doc/refman/8.0/en/replica-logs.html)). Ensure that the connection metadata repository can be accessed only by the database administrator. An alternative to storing the password in the connection metadata repository is to use the [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) or [START GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html) statement to specify credentials for connecting to the source.

Use a restricted access mode to protect database backups that include log tables or log files containing passwords.

### Making MySQL Secure Against Attackers

When you connect to a MySQL server, you should use a password. The password is not transmitted as cleartext over the connection.

All other information is transferred as text, and can be read by anyone who is able to watch the connection. If the connection between the client and the server goes through an untrusted network, and you are concerned about this, you can use the compressed protocol to make traffic much more difficult to decipher. You can also use MySQL's internal SSL support to make the connection even more secure. See [Section 6.3, “Using Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connections.html). Alternatively, use SSH to get an encrypted TCP/IP connection between a MySQL server and a MySQL client. You can find an Open Source SSH client at <http://www.openssh.org/>, and a comparison of both Open Source and Commercial SSH clients at <http://en.wikipedia.org/wiki/Comparison_of_SSH_clients>.

To make a MySQL system secure, you should strongly consider the following suggestions:

* Require all MySQL accounts to have a password. A client program does not necessarily know the identity of the person running it. It is common for client/server applications that the user can specify any user name to the client program. For example, anyone can use the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) program to connect as any other person simply by invoking it as mysql -u ***other\_user*** ***db\_name*** if ***other\_user*** has no password. If all accounts have a password, connecting using another user's account becomes much more difficult.

For a discussion of methods for setting passwords, see [Section 6.2.14, “Assigning Account Passwords”](https://dev.mysql.com/doc/refman/8.0/en/assigning-passwords.html).

* Make sure that the only Unix user account with read or write privileges in the database directories is the account that is used for running [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html).
* Never run the MySQL server as the Unix root user. This is extremely dangerous, because any user with the [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege is able to cause the server to create files as root (for example, ~root/.bashrc). To prevent this, [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) refuses to run as root unless that is specified explicitly using the [--user=root](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_user) option.

[**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) can (and should) be run as an ordinary, unprivileged user instead. You can create a separate Unix account named mysql to make everything even more secure. Use this account only for administering MySQL. To start [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) as a different Unix user, add a user option that specifies the user name in the [mysqld] group of the my.cnf option file where you specify server options. For example:

[mysqld]

user=mysql

This causes the server to start as the designated user whether you start it manually or by using [**mysqld\_safe**](https://dev.mysql.com/doc/refman/8.0/en/mysqld-safe.html) or [**mysql.server**](https://dev.mysql.com/doc/refman/8.0/en/mysql-server.html). For more details, see [Section 6.1.5, “How to Run MySQL as a Normal User”](https://dev.mysql.com/doc/refman/8.0/en/changing-mysql-user.html).

Running [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) as a Unix user other than root does not mean that you need to change the root user name in the user table. User names for MySQL accounts have nothing to do with user names for Unix accounts.

* Do not grant the [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege to nonadministrative users. Any user that has this privilege can write a file anywhere in the file system with the privileges of the [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) daemon. This includes the server's data directory containing the files that implement the privilege tables. To make [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file)-privilege operations a bit safer, files generated with [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/8.0/en/select-into.html) do not overwrite existing files and are writable by everyone.

The [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege may also be used to read any file that is world-readable or accessible to the Unix user that the server runs as. With this privilege, you can read any file into a database table. This could be abused, for example, by using [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) to load /etc/passwd into a table, which then can be displayed with [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html).

To limit the location in which files can be read and written, set the [secure\_file\_priv](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv) system to a specific directory. See [Section 5.1.8, “Server System Variables”](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html).

* Encrypt binary log files and relay log files. Encryption helps to protect these files and the potentially sensitive data contained in them from being misused by outside attackers, and also from unauthorized viewing by users of the operating system where they are stored. You enable encryption on a MySQL server by setting the [binlog\_encryption](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_encryption) system variable to ON. For more information, see [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](https://dev.mysql.com/doc/refman/8.0/en/replication-binlog-encryption.html).
* Do not grant the [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) or [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege to nonadministrative users. The output of [**mysqladmin processlist**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) and [SHOW PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html) shows the text of any statements currently being executed, so any user who is permitted to see the server process list might be able to see statements issued by other users.

[**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) reserves an extra connection for users who have the [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) or [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege, so that a MySQL root user can log in and check server activity even if all normal connections are in use.

The [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege can be used to terminate client connections, change server operation by changing the value of system variables, and control replication servers.

* Do not permit the use of symlinks to tables. (This capability can be disabled with the [--skip-symbolic-links](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_symbolic-links) option.) This is especially important if you run [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) as root, because anyone that has write access to the server's data directory then could delete any file in the system! See [Section 8.12.2.2, “Using Symbolic Links for MyISAM Tables on Unix”](https://dev.mysql.com/doc/refman/8.0/en/symbolic-links-to-tables.html).
* Stored programs and views should be written using the security guidelines discussed in [Section 25.6, “Stored Object Access Control”](https://dev.mysql.com/doc/refman/8.0/en/stored-objects-security.html).
* If you do not trust your DNS, you should use IP addresses rather than host names in the grant tables. In any case, you should be very careful about creating grant table entries using host name values that contain wildcards.
* If you want to restrict the number of connections permitted to a single account, you can do so by setting the [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) variable in [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html). The [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements also support resource control options for limiting the extent of server use permitted to an account. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), and [Section 13.7.1.1, “ALTER USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html).
* If the plugin directory is writable by the server, it may be possible for a user to write executable code to a file in the directory using [SELECT ... INTO DUMPFILE](https://dev.mysql.com/doc/refman/8.0/en/select.html). This can be prevented by making [plugin\_dir](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_plugin_dir) read only to the server or by setting [secure\_file\_priv](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv) to a directory where [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) writes can be made safely.

### Security-Related mysqld Options and Variables

The following table shows [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) options and system variables that affect security. For descriptions of each of these, see [Section 5.1.7, “Server Command Options”](https://dev.mysql.com/doc/refman/8.0/en/server-options.html), and [Section 5.1.8, “Server System Variables”](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html).

**Table 6.1 Security Option and Variable Summary**

| **Name** | **Cmd-Line** | **Option File** | **System Var** | **Status Var** | **Var Scope** | **Dynamic** |
| --- | --- | --- | --- | --- | --- | --- |
| [**allow-suspicious-udfs**](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_allow-suspicious-udfs) | Yes | Yes |  |  |  |  |
| [**automatic\_sp\_privileges**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_automatic_sp_privileges) | Yes | Yes | Yes |  | Global | Yes |
| [**chroot**](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_chroot) | Yes | Yes |  |  |  |  |
| [**local\_infile**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) | Yes | Yes | Yes |  | Global | Yes |
| [**safe-user-create**](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_safe-user-create) | Yes | Yes |  |  |  |  |
| [**secure\_file\_priv**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv) | Yes | Yes | Yes |  | Global | No |
| [**skip-grant-tables**](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) | Yes | Yes |  |  |  |  |
| [**skip\_name\_resolve**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_skip_name_resolve) | Yes | Yes | Yes |  | Global | No |
| [**skip\_networking**](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_skip_networking) | Yes | Yes | Yes |  | Global | No |
| [**skip\_show\_database**](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-show-database) | Yes | Yes | Yes |  | Global | No |

### How to Run MySQL as a Normal User

On Windows, you can run the server as a Windows service using a normal user account.

On Linux, for installations performed using a MySQL repository or RPM packages, the MySQL server [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) should be started by the local mysql operating system user. Starting by another operating system user is not supported by the init scripts that are included as part of the MySQL repositories.

On Unix (or Linux for installations performed using tar.gz packages) , the MySQL server [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) can be started and run by any user. However, you should avoid running the server as the Unix root user for security reasons. To change [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) to run as a normal unprivileged Unix user ***user\_name***, you must do the following:

1. Stop the server if it is running (use [**mysqladmin shutdown**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html)).
2. Change the database directories and files so that ***user\_name*** has privileges to read and write files in them (you might need to do this as the Unix root user):

shell> chown -R user\_name /path/to/mysql/datadir

If you do not do this, the server cannot access databases or tables when it runs as ***user\_name***.

If directories or files within the MySQL data directory are symbolic links, chown -R might not follow symbolic links for you. If it does not, you must also follow those links and change the directories and files they point to.

1. Start the server as user ***user\_name***. Another alternative is to start [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) as the Unix root user and use the [--user=***user\_name***](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_user) option. [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) starts, then switches to run as the Unix user ***user\_name*** before accepting any connections.
2. To start the server as the given user automatically at system startup time, specify the user name by adding a user option to the [mysqld] group of the /etc/my.cnf option file or the my.cnf option file in the server's data directory. For example:
3. [mysqld]

user=user\_name

If your Unix machine itself is not secured, you should assign passwords to the MySQL root account in the grant tables. Otherwise, any user with a login account on that machine can run the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client with a [--user=root](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_user) option and perform any operation. (It is a good idea to assign passwords to MySQL accounts in any case, but especially so when other login accounts exist on the server host.) See [Section 2.10.4, “Securing the Initial MySQL Account”](https://dev.mysql.com/doc/refman/8.0/en/default-privileges.html).

### Security Considerations for LOAD DATA LOCAL

The [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statement loads a data file into a table. The statement can load a file located on the server host, or, if the LOCAL keyword is specified, on the client host.

The LOCAL version of [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) has two potential security issues:

* Because [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) is an SQL statement, parsing occurs on the server side, and transfer of the file from the client host to the server host is initiated by the MySQL server, which tells the client the file named in the statement. In theory, a patched server could tell the client program to transfer a file of the server's choosing rather than the file named in the statement. Such a server could access any file on the client host to which the client user has read access. (A patched server could in fact reply with a file-transfer request to any statement, not just [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html), so a more fundamental issue is that clients should not connect to untrusted servers.)
* In a Web environment where the clients are connecting from a Web server, a user could use [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) to read any files that the Web server process has read access to (assuming that a user could run any statement against the SQL server). In this environment, the client with respect to the MySQL server actually is the Web server, not a remote program being run by users who connect to the Web server.

To avoid connecting to untrusted servers, clients can establish a secure connection and verify the server identity by connecting using the [--ssl-mode=VERIFY\_IDENTITY](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option and the appropriate CA certificate.

To avoid [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) issues, clients should avoid using LOCAL unless proper client-side precautions have been taken.

For control over local data loading, MySQL permits the capability to be enabled or disabled. In addition, as of MySQL 8.0.21, MySQL enables clients to restrict local data loading operations to files located in a designated directory.

#### Enabling or Disabling Local Data Loading Capability

Adminstrators and applications can configure whether to permit local data loading as follows:

* On the server side:
  + The [local\_infile](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) system variable controls server-side LOCAL capability. Depending on the [local\_infile](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) setting, the server refuses or permits local data loading by clients that request local data loading.
  + By default, [local\_infile](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) is disabled. To explicitly cause the server to refuse or permit [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statements (regardless of how client programs and libraries are configured at build time or runtime), start [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) with [local\_infile](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) disabled or enabled. [local\_infile](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_local_infile) can also be set at runtime.
* On the client side:
  + The [ENABLED\_LOCAL\_INFILE](https://dev.mysql.com/doc/refman/8.0/en/source-configuration-options.html#option_cmake_enabled_local_infile) **CMake** option controls the compiled-in default LOCAL capability for the MySQL client library (see [Section 2.9.7, “MySQL Source-Configuration Options”](https://dev.mysql.com/doc/refman/8.0/en/source-configuration-options.html)). Clients that make no explicit arrangements therefore have LOCAL capability disabled or enabled according to the [ENABLED\_LOCAL\_INFILE](https://dev.mysql.com/doc/refman/8.0/en/source-configuration-options.html#option_cmake_enabled_local_infile) setting specified at MySQL build time.
  + By default, the client library in MySQL binary distributions is compiled with [ENABLED\_LOCAL\_INFILE](https://dev.mysql.com/doc/refman/8.0/en/source-configuration-options.html#option_cmake_enabled_local_infile) disabled. If you compile MySQL from source, configure it with [ENABLED\_LOCAL\_INFILE](https://dev.mysql.com/doc/refman/8.0/en/source-configuration-options.html#option_cmake_enabled_local_infile) disabled or enabled based on whether clients that make no explicit arrangements should have LOCAL capability disabled or enabled.
  + For client programs that use the C API, local data loading capability is determined by the default compiled into the MySQL client library. To enable or disable it explicitly, invoke the [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) C API function to disable or enable the MYSQL\_OPT\_LOCAL\_INFILE option. See [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html).
  + For the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client, local data loading capability is determined by the default compiled into the MySQL client library. To disable or enable it explicitly, use the [--local-infile=0](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_local-infile) or [--local-infile[=1]](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_local-infile) option.
  + For the [**mysqlimport**](https://dev.mysql.com/doc/refman/8.0/en/mysqlimport.html) client, local data loading is not used by default. To disable or enable it explicitly, use the [--local=0](https://dev.mysql.com/doc/refman/8.0/en/mysqlimport.html#option_mysqlimport_local) or [--local[=1]](https://dev.mysql.com/doc/refman/8.0/en/mysqlimport.html#option_mysqlimport_local) option.
  + If you use [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) in Perl scripts or other programs that read the [client] group from option files, you can add a local-infile option setting to that group. To prevent problems for programs that do not understand this option, specify it using the [loose-](https://dev.mysql.com/doc/refman/8.0/en/option-modifiers.html) prefix:
  + [client]

loose-local-infile=0

or:

[client]

loose-local-infile=1

* + In all cases, successful use of a LOCAL load operation by a client also requires that the server permits local loading.

If LOCAL capability is disabled, on either the server or client side, a client that attempts to issue a [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statement receives the following error message:

ERROR 3950 (42000): Loading local data is disabled; this must be

enabled on both the client and server side

#### Restricting Files Permitted for Local Data Loading

As of MySQL 8.0.21, the MySQL client library enables client applications to restrict local data loading operations to files located in a designated directory. Certain MySQL client programs take advantage of this capability.

Client programs that use the C API can control which files to permit for load data loading using the MYSQL\_OPT\_LOCAL\_INFILE and MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR options of the [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) C API function (see [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html)).

The effect of MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR depends on whether LOCAL data loading is enabled or disabled:

* If LOCAL data loading is enabled, either by default in the MySQL client library or by explicitly enabling MYSQL\_OPT\_LOCAL\_INFILE, the MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR option has no effect.
* If LOCAL data loading is disabled, either by default in the MySQL client library or by explicitly disabling MYSQL\_OPT\_LOCAL\_INFILE, the MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR option can be used to designate a permitted directory for locally loaded files. In this case, LOCAL data loading is permitted but restricted to files located in the designated directory. Interpretation of the MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR value is as follows:
  + If the value is the null pointer (the default), it names no directory, with the result that no files are permitted for LOCAL data loading.
  + If the value is a directory path name, LOCAL data loading is permitted but restricted to files located in the named directory. Comparison of the directory path name and the path name of files to be loaded is case-sensitive regardless of the case-sensitivity of the underlying file system.

MySQL client programs use the preceding [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) options as follows:

* The [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client has a [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) option that takes a directory path or an empty string. [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) uses the option value to set the MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR option (with an empty string setting the value to the null pointer). The effect of [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) depends on whether LOCAL data loading is enabled:
  + If LOCAL data loading is enabled, either by default in the MySQL client library or by specifying [--local-infile[=1]](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_local-infile), the [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) option is ignored.
  + If LOCAL data loading is disabled, either by default in the MySQL client library or by specifying [--local-infile=0](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_local-infile), the [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) option applies.

When [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) applies, the option value designates the directory in which local data files must be located. Comparison of the directory path name and the path name of files to be loaded is case-sensitive regardless of the case-sensitivity of the underlying file system. If the option value is the empty string, it names no directory, with the result that no files are permitted for local data loading.

* [**mysqlimport**](https://dev.mysql.com/doc/refman/8.0/en/mysqlimport.html) sets MYSQL\_OPT\_LOAD\_DATA\_LOCAL\_DIR for each file that it processes so that the directory containing the file is the permitted local loading directory.
* For data loading operations corresponding to [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statements, [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) extracts the files from the binary log events, writes them as temporary ffiles to the local file system, and writes [LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statements to cause the files to be loaded. By default, [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) writes these temporary files to an operating system-specific directory. The [--local-load](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html#option_mysqlbinlog_local-load) option can be used to explicitly specify the directory where [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) should prepare local temporary files.

Because other processes can write files to the default system-specific directory, it is advisable to specify the [--local-load](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html#option_mysqlbinlog_local-load) option to [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) to designate a different directory for data files, and then designate that same directory by specifying the [--load-data-local-dir](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_load-data-local-dir) option to [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) when processing the output from [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html).

### Client Programming Security Guidelines

Applications that access MySQL should not trust any data entered by users, who can try to trick your code by entering special or escaped character sequences in Web forms, URLs, or whatever application you have built. Be sure that your application remains secure if a user enters something like ; DROP DATABASE mysql;. This is an extreme example, but large security leaks and data loss might occur as a result of hackers using similar techniques, if you do not prepare for them.

A common mistake is to protect only string data values. Remember to check numeric data as well. If an application generates a query such as SELECT \* FROM table WHERE ID=234 when a user enters the value 234, the user can enter the value 234 OR 1=1 to cause the application to generate the query SELECT \* FROM table WHERE ID=234 OR 1=1. As a result, the server retrieves every row in the table. This exposes every row and causes excessive server load. The simplest way to protect from this type of attack is to use single quotation marks around the numeric constants: SELECT \* FROM table WHERE ID='234'. If the user enters extra information, it all becomes part of the string. In a numeric context, MySQL automatically converts this string to a number and strips any trailing nonnumeric characters from it.

Sometimes people think that if a database contains only publicly available data, it need not be protected. This is incorrect. Even if it is permissible to display any row in the database, you should still protect against denial of service attacks (for example, those that are based on the technique in the preceding paragraph that causes the server to waste resources). Otherwise, your server becomes unresponsive to legitimate users.

Checklist:

* Enable strict SQL mode to tell the server to be more restrictive of what data values it accepts. See [Section 5.1.11, “Server SQL Modes”](https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html).
* Try to enter single and double quotation marks (' and ") in all of your Web forms. If you get any kind of MySQL error, investigate the problem right away.
* Try to modify dynamic URLs by adding %22 ("), %23 (#), and %27 (') to them.
* Try to modify data types in dynamic URLs from numeric to character types using the characters shown in the previous examples. Your application should be safe against these and similar attacks.
* Try to enter characters, spaces, and special symbols rather than numbers in numeric fields. Your application should remove them before passing them to MySQL or else generate an error. Passing unchecked values to MySQL is very dangerous!
* Check the size of data before passing it to MySQL.
* Have your application connect to the database using a user name different from the one you use for administrative purposes. Do not give your applications any access privileges they do not need.

Many application programming interfaces provide a means of escaping special characters in data values. Properly used, this prevents application users from entering values that cause the application to generate statements that have a different effect than you intend:

* MySQL C API: Use the [mysql\_real\_escape\_string\_quote()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-escape-string-quote.html) API call.
* MySQL++: Use the escape and quote modifiers for query streams.
* PHP: Use either the mysqli or pdo\_mysql extensions, and not the older ext/mysql extension. The preferred API's support the improved MySQL authentication protocol and passwords, as well as prepared statements with placeholders. See also [Choosing an API](https://dev.mysql.com/doc/apis-php/en/apis-php-mysqlinfo.api.choosing.html).

If the older ext/mysql extension must be used, then for escaping use the [mysql\_real\_escape\_string\_quote()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-escape-string-quote.html) function and not [mysql\_escape\_string()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-escape-string.html) or addslashes() because only [mysql\_real\_escape\_string\_quote()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-escape-string-quote.html) is character set-aware; the other functions can be “bypassed” when using (invalid) multibyte character sets.

* Perl DBI: Use placeholders or the quote() method.
* Ruby DBI: Use placeholders or the quote() method.
* Java JDBC: Use a PreparedStatement object and placeholders.

Other programming interfaces might have similar capabilities.

## Access Control and Account Management

MySQL enables the creation of accounts that permit client users to connect to the server and access data managed by the server. The primary function of the MySQL privilege system is to authenticate a user who connects from a given host and to associate that user with privileges on a database such as [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html), [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), and [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html). Additional functionality includes the ability to grant privileges for administrative operations.

To control which users can connect, each account can be assigned authentication credentials such as a password. The user interface to MySQL accounts consists of SQL statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html). See [Section 13.7.1, “Account Management Statements”](https://dev.mysql.com/doc/refman/8.0/en/account-management-statements.html).

The MySQL privilege system ensures that all users may perform only the operations permitted to them. As a user, when you connect to a MySQL server, your identity is determined by the host from which you connect and the user name you specify. When you issue requests after connecting, the system grants privileges according to your identity and what you want to do.

MySQL considers both your host name and user name in identifying you because there is no reason to assume that a given user name belongs to the same person on all hosts. For example, the user joe who connects from office.example.com need not be the same person as the user joe who connects from home.example.com. MySQL handles this by enabling you to distinguish users on different hosts that happen to have the same name: You can grant one set of privileges for connections by joe from office.example.com, and a different set of privileges for connections by joe from home.example.com. To see what privileges a given account has, use the [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) statement. For example:

SHOW GRANTS FOR 'joe'@'office.example.com';

SHOW GRANTS FOR 'joe'@'home.example.com';

Internally, the server stores privilege information in the grant tables of the mysql system database. The MySQL server reads the contents of these tables into memory when it starts and bases access-control decisions on the in-memory copies of the grant tables.

MySQL access control involves two stages when you run a client program that connects to the server:

***Stage 1:*** The server accepts or rejects the connection based on your identity and whether you can verify your identity by supplying the correct password.

***Stage 2:*** Assuming that you can connect, the server checks each statement you issue to determine whether you have sufficient privileges to perform it. For example, if you try to select rows from a table in a database or drop a table from the database, the server verifies that you have the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege for the table or the [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) privilege for the database.

For a more detailed description of what happens during each stage, see [Section 6.2.6, “Access Control, Stage 1: Connection Verification”](https://dev.mysql.com/doc/refman/8.0/en/connection-access.html), and [Section 6.2.7, “Access Control, Stage 2: Request Verification”](https://dev.mysql.com/doc/refman/8.0/en/request-access.html). For help in diagnosing privilege-related problems, see [Section 6.2.21, “Troubleshooting Problems Connecting to MySQL”](https://dev.mysql.com/doc/refman/8.0/en/problems-connecting.html).

If your privileges are changed (either by yourself or someone else) while you are connected, those changes do not necessarily take effect immediately for the next statement that you issue. For details about the conditions under which the server reloads the grant tables, see [Section 6.2.13, “When Privilege Changes Take Effect”](https://dev.mysql.com/doc/refman/8.0/en/privilege-changes.html).

There are some things that you cannot do with the MySQL privilege system:

* You cannot explicitly specify that a given user should be denied access. That is, you cannot explicitly match a user and then refuse the connection.
* You cannot specify that a user has privileges to create or drop tables in a database but not to create or drop the database itself.
* A password applies globally to an account. You cannot associate a password with a specific object such as a database, table, or routine.

### Account User Names and Passwords

MySQL stores accounts in the user table of the mysql system database. An account is defined in terms of a user name and the client host or hosts from which the user can connect to the server. For information about account representation in the user table, see [Section 6.2.3, “Grant Tables”](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html).

An account may also have authentication credentials such as a password. The credentials are handled by the account authentication plugin. MySQL supports multiple authentication plugins. Some of them use built-in authentication methods, whereas others enable authentication using external authentication methods. See [Section 6.2.17, “Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html).

There are several distinctions between the way user names and passwords are used by MySQL and your operating system:

* User names, as used by MySQL for authentication purposes, have nothing to do with user names (login names) as used by Windows or Unix. On Unix, most MySQL clients by default try to log in using the current Unix user name as the MySQL user name, but that is for convenience only. The default can be overridden easily, because client programs permit any user name to be specified with a -u or --user option. This means that anyone can attempt to connect to the server using any user name, so you cannot make a database secure in any way unless all MySQL accounts have passwords. Anyone who specifies a user name for an account that has no password can connect successfully to the server.
* MySQL user names are up to 32 characters long. Operating system user names may have a different maximum length.

**Warning**

The MySQL user name length limit is hardcoded in MySQL servers and clients, and trying to circumvent it by modifying the definitions of the tables in the mysql database does not work.

You should never alter the structure of tables in the mysql database in any manner whatsoever except by means of the procedure that is described in [Section 2.11, “Upgrading MySQL”](https://dev.mysql.com/doc/refman/8.0/en/upgrading.html). Attempting to redefine MySQL's system tables in any other fashion results in undefined and unsupported behavior. The server is free to ignore rows that become malformed as a result of such modifications.

* To authenticate client connections for accounts that use built-in authentication methods, the server uses passwords stored in the user table. These passwords are distinct from passwords for logging in to your operating system. There is no necessary connection between the “external” password you use to log in to a Windows or Unix machine and the password you use to access the MySQL server on that machine.

If the server authenticates a client using some other plugin, the authentication method that the plugin implements may or may not use a password stored in the user table. In this case, it is possible that an external password is also used to authenticate to the MySQL server.

* Passwords stored in the user table are encrypted using plugin-specific algorithms.
* If the user name and password contain only ASCII characters, it is possible to connect to the server regardless of character set settings. To enable connections when the user name or password contain non-ASCII characters, client applications should call the [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) C API function with the MYSQL\_SET\_CHARSET\_NAME option and appropriate character set name as arguments. This causes authentication to take place using the specified character set. Otherwise, authentication fails unless the server default character set is the same as the encoding in the authentication defaults.

Standard MySQL client programs support a --default-character-set option that causes [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) to be called as just described. In addition, character set autodetection is supported as described in [Section 10.4, “Connection Character Sets and Collations”](https://dev.mysql.com/doc/refman/8.0/en/charset-connection.html). For programs that use a connector that is not based on the C API, the connector may provide an equivalent to [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) that can be used instead. Check the connector documentation.

The preceding notes do not apply for ucs2, utf16, and utf32, which are not permitted as client character sets.

The MySQL installation process populates the grant tables with an initial root account, as described in [Section 2.10.4, “Securing the Initial MySQL Account”](https://dev.mysql.com/doc/refman/8.0/en/default-privileges.html), which also discusses how to assign a password to it. Thereafter, you normally set up, modify, and remove MySQL accounts using statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html), [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html). See [Section 6.2.8, “Adding Accounts, Assigning Privileges, and Dropping Accounts”](https://dev.mysql.com/doc/refman/8.0/en/creating-accounts.html), and [Section 13.7.1, “Account Management Statements”](https://dev.mysql.com/doc/refman/8.0/en/account-management-statements.html).

To connect to a MySQL server with a command-line client, specify user name and password options as necessary for the account that you want to use:

shell> mysql --user=finley --password db\_name

If you prefer short options, the command looks like this:

shell> mysql -u finley -p db\_name

If you omit the password value following the [--password](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password) or -p option on the command line (as just shown), the client prompts for one. Alternatively, the password can be specified on the command line:

shell> mysql --user=finley --password=password db\_name

shell> mysql -u finley -ppassword db\_name

If you use the -p option, there must be no space between -p and the following password value.

Specifying a password on the command line should be considered insecure. See [Section 6.1.2.1, “End-User Guidelines for Password Security”](https://dev.mysql.com/doc/refman/8.0/en/password-security-user.html). To avoid giving the password on the command line, use an option file or a login path file. See [Section 4.2.2.2, “Using Option Files”](https://dev.mysql.com/doc/refman/8.0/en/option-files.html), and [Section 4.6.7, “**mysql\_config\_editor** — MySQL Configuration Utility”](https://dev.mysql.com/doc/refman/8.0/en/mysql-config-editor.html).

For additional information about specifying user names, passwords, and other connection parameters, see [Section 4.2.4, “Connecting to the MySQL Server Using Command Options”](https://dev.mysql.com/doc/refman/8.0/en/connecting.html).

### Privileges Provided by MySQL

The privileges granted to a MySQL account determine which operations the account can perform. MySQL privileges differ in the contexts in which they apply and at different levels of operation:

* Administrative privileges enable users to manage operation of the MySQL server. These privileges are global because they are not specific to a particular database.
* Database privileges apply to a database and to all objects within it. These privileges can be granted for specific databases, or globally so that they apply to all databases.
* Privileges for database objects such as tables, indexes, views, and stored routines can be granted for specific objects within a database, for all objects of a given type within a database (for example, all tables in a database), or globally for all objects of a given type in all databases.

Privileges also differ in terms of whether they are static (built in to the server) or dynamic (defined at runtime). Whether a privilege is static or dynamic affects its availability to be granted to user accounts and roles. For information about the differences between static and dynamic privileges, see [Static Versus Dynamic Privileges](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#static-dynamic-privileges).)

Information about account privileges is stored in the grant tables in the mysql system database. For a description of the structure and contents of these tables, see [Section 6.2.3, “Grant Tables”](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html). The MySQL server reads the contents of the grant tables into memory when it starts, and reloads them under the circumstances indicated in [Section 6.2.13, “When Privilege Changes Take Effect”](https://dev.mysql.com/doc/refman/8.0/en/privilege-changes.html). The server bases access-control decisions on the in-memory copies of the grant tables.

**Important**

Some MySQL releases introduce changes to the grant tables to add new privileges or features. To make sure that you can take advantage of any new capabilities, update your grant tables to the current structure whenever you upgrade MySQL. See [Section 2.11, “Upgrading MySQL”](https://dev.mysql.com/doc/refman/8.0/en/upgrading.html).

The following sections summarize the available privileges, provide more detailed descriptions of each privilege, and offer usage guidelines.

#### Summary of Available Privileges

The following table shows the static privilege names used in [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements, along with the column name associated with each privilege in the grant tables and the context in which the privilege applies.

**Table 6.2 Permissible Static Privileges for GRANT and REVOKE**

| **Privilege** | **Grant Table Column** | **Context** |
| --- | --- | --- |
| [ALL [PRIVILEGES]](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_all) | Synonym for “all privileges” | Server administration |
| [ALTER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_alter) | Alter\_priv | Tables |
| [ALTER ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_alter-routine) | Alter\_routine\_priv | Stored routines |
| [CREATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create) | Create\_priv | Databases, tables, or indexes |
| [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-role) | Create\_role\_priv | Server administration |
| [CREATE ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-routine) | Create\_routine\_priv | Stored routines |
| [CREATE TABLESPACE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-tablespace) | Create\_tablespace\_priv | Server administration |
| [CREATE TEMPORARY TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-temporary-tables) | Create\_tmp\_table\_priv | Tables |
| [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) | Create\_user\_priv | Server administration |
| [CREATE VIEW](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-view) | Create\_view\_priv | Views |
| [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete) | Delete\_priv | Tables |
| [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) | Drop\_priv | Databases, tables, or views |
| [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop-role) | Drop\_role\_priv | Server administration |
| [EVENT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_event) | Event\_priv | Databases |
| [EXECUTE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_execute) | Execute\_priv | Stored routines |
| [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) | File\_priv | File access on server host |
| [GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_grant-option) | Grant\_priv | Databases, tables, or stored routines |
| [INDEX](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_index) | Index\_priv | Tables |
| [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) | Insert\_priv | Tables or columns |
| [LOCK TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_lock-tables) | Lock\_tables\_priv | Databases |
| [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) | Process\_priv | Server administration |
| [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) | See proxies\_priv table | Server administration |
| [REFERENCES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_references) | References\_priv | Databases or tables |
| [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) | Reload\_priv | Server administration |
| [REPLICATION CLIENT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-client) | Repl\_client\_priv | Server administration |
| [REPLICATION SLAVE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave) | Repl\_slave\_priv | Server administration |
| [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) | Select\_priv | Tables or columns |
| [SHOW DATABASES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_show-databases) | Show\_db\_priv | Server administration |
| [SHOW VIEW](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_show-view) | Show\_view\_priv | Views |
| [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown) | Shutdown\_priv | Server administration |
| [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) | Super\_priv | Server administration |
| [TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_trigger) | Trigger\_priv | Tables |
| [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) | Update\_priv | Tables or columns |
| [USAGE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_usage) | Synonym for “no privileges” | Server administration |

The following table shows the dynamic privilege names used in [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements, along with the context in which the privilege applies.

**Table 6.3 Permissible Dynamic Privileges for GRANT and REVOKE**

| **Privilege** | **Context** |
| --- | --- |
| [APPLICATION\_PASSWORD\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_application-password-admin) | Dual password administration |
| [AUDIT\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_audit-admin) | Audit log administration |
| [BACKUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_backup-admin) | Backup administration |
| [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-admin) | Backup and Replication administration |
| [BINLOG\_ENCRYPTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-encryption-admin) | Backup and Replication administration |
| [CLONE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_clone-admin) | Clone administration |
| [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) | Server administration |
| [ENCRYPTION\_KEY\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_encryption-key-admin) | Server administration |
| [FIREWALL\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_firewall-admin) | Firewall administration |
| [FIREWALL\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_firewall-user) | Firewall administration |
| [FLUSH\_OPTIMIZER\_COSTS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_flush-optimizer-costs) | Server administration |
| [FLUSH\_STATUS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_flush-status) | Server administration |
| [FLUSH\_TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_flush-tables) | Server administration |
| [FLUSH\_USER\_RESOURCES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_flush-user-resources) | Server administration |
| [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) | Replication administration |
| [INNODB\_REDO\_LOG\_ARCHIVE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_innodb-redo-log-archive) | Redo log archiving administration |
| [NDB\_STORED\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_ndb-stored-user) | NDB Cluster |
| [PERSIST\_RO\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_persist-ro-variables-admin) | Server administration |
| [REPLICATION\_APPLIER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-applier) | PRIVILEGE\_CHECKS\_USER for a replication channel |
| [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) | Replication administration |
| [RESOURCE\_GROUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_resource-group-admin) | Resource group administration |
| [RESOURCE\_GROUP\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_resource-group-user) | Resource group administration |
| [ROLE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_role-admin) | Server administration |
| [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) | Server administration |
| [SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_set-user-id) | Server administration |
| [SHOW\_ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_show-routine) | Server administration |
| [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) | Server administration |
| [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin) | Server administration |
| [TABLE\_ENCRYPTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_table-encryption-admin) | Server administration |
| [VERSION\_TOKEN\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_version-token-admin) | Server administration |
| [XA\_RECOVER\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_xa-recover-admin) | Server administration |

#### Static Privilege Descriptions

Static privileges are built in to the server, in contrast to dynamic privileges, which are defined at runtime. The following list describes each static privilege available in MySQL.

Particular SQL statements might have more specific privilege requirements than indicated here. If so, the description for the statement in question provides the details.

* [ALL](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_all), [ALL PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_all)

These privilege specifiers are shorthand for “all privileges available at a given privilege level” (except [GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_grant-option)). For example, granting [ALL](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_all) at the global or table level grants all global privileges or all table-level privileges, respectively.

* [ALTER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_alter)

Enables use of the [ALTER TABLE](https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) statement to change the structure of tables. [ALTER TABLE](https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) also requires the [CREATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create) and [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) privileges. Renaming a table requires [ALTER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_alter) and [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) on the old table, [CREATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create), and [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) on the new table.

* [ALTER ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_alter-routine)

Enables use of statements that alter or drop stored routines (stored procedures and functions). For routines that fall within the scope at which the privilege is granted and for which the user is not the user named as the routine DEFINER, also enables access to routine properties other than the routine definition.

* [CREATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create)

Enables use of statements that create new databases and tables.

* [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-role)

Enables use of the [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) statement. (The [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege also enables use of the [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) statement.) See [Section 6.2.10, “Using Roles”](https://dev.mysql.com/doc/refman/8.0/en/roles.html).

The [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-role) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop-role) privileges are not as powerful as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) because they can be used only to create and drop accounts. They cannot be used as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) can be modify account attributes or rename accounts. See [User and Role Interchangeability](https://dev.mysql.com/doc/refman/8.0/en/roles.html#role-user-interchangeability).

* [CREATE ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-routine)

Enables use of statements that create stored routines (stored procedures and functions). For routines that fall within the scope at which the privilege is granted and for which the user is not the user named as the routine DEFINER, also enables access to routine properties other than the routine definition.

* [CREATE TABLESPACE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-tablespace)

Enables use of statements that create, alter, or drop tablespaces and log file groups.

* [CREATE TEMPORARY TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-temporary-tables)

Enables the creation of temporary tables using the [CREATE TEMPORARY TABLE](https://dev.mysql.com/doc/refman/8.0/en/create-temporary-table.html) statement.

After a session has created a temporary table, the server performs no further privilege checks on the table. The creating session can perform any operation on the table, such as [DROP TABLE](https://dev.mysql.com/doc/refman/8.0/en/drop-table.html), [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html). For more information, see [Section 13.1.20.2, “CREATE TEMPORARY TABLE Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-temporary-table.html).

* [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-user)

Enables use of the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html), [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html), [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html), [RENAME USER](https://dev.mysql.com/doc/refman/8.0/en/rename-user.html), and [REVOKE ALL PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements.

* [CREATE VIEW](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_create-view)

Enables use of the [CREATE VIEW](https://dev.mysql.com/doc/refman/8.0/en/create-view.html) statement.

* [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_delete)

Enables rows to be deleted from tables in a database.

* [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_drop)

Enables use of statements that drop (remove) existing databases, tables, and views. The [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) privilege is required to use the ALTER TABLE ... DROP PARTITION statement on a partitioned table. The [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) privilege is also required for [TRUNCATE TABLE](https://dev.mysql.com/doc/refman/8.0/en/truncate-table.html).

* [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_drop-role)

Enables use of the [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html) statement. (The [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege also enables use of the [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html) statement.) See [Section 6.2.10, “Using Roles”](https://dev.mysql.com/doc/refman/8.0/en/roles.html).

The [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-role) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop-role) privileges are not as powerful as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) because they can be used only to create and drop accounts. They cannot be used as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) can be modify account attributes or rename accounts. See [User and Role Interchangeability](https://dev.mysql.com/doc/refman/8.0/en/roles.html#role-user-interchangeability).

* [EVENT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_event)

Enables use of statements that create, alter, drop, or display events for the Event Scheduler.

* [EXECUTE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_execute)

Enables use of statements that execute stored routines (stored procedures and functions). For routines that fall within the scope at which the privilege is granted and for which the user is not the user named as the routine DEFINER, also enables access to routine properties other than the routine definition.

* [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_file)

Affects the following operations and server behaviors:

* + Enables reading and writing files on the server host using the [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) and [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/8.0/en/select-into.html) statements and the [LOAD\_FILE()](https://dev.mysql.com/doc/refman/8.0/en/string-functions.html#function_load-file) function. A user who has the [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege can read any file on the server host that is either world-readable or readable by the MySQL server. (This implies the user can read any file in any database directory, because the server can access any of those files.)
  + Enables creating new files in any directory where the MySQL server has write access. This includes the server's data directory containing the files that implement the privilege tables.
  + Enables use of the DATA DIRECTORY or INDEX DIRECTORY table option for the [CREATE TABLE](https://dev.mysql.com/doc/refman/8.0/en/create-table.html) statement.

As a security measure, the server does not overwrite existing files.

To limit the location in which files can be read and written, set the [secure\_file\_priv](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv) system variable to a specific directory..

* [GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_grant-option)

Enables you to grant to or revoke from other users those privileges that you yourself possess.

* [INDEX](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_index)

Enables use of statements that create or drop (remove) indexes. [INDEX](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_index) applies to existing tables. If you have the [CREATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create) privilege for a table, you can include index definitions in the [CREATE TABLE](https://dev.mysql.com/doc/refman/8.0/en/create-table.html) statement.

* [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_insert)

Enables rows to be inserted into tables in a database. [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) is also required for the [ANALYZE TABLE](https://dev.mysql.com/doc/refman/8.0/en/analyze-table.html), [OPTIMIZE TABLE](https://dev.mysql.com/doc/refman/8.0/en/optimize-table.html), and [REPAIR TABLE](https://dev.mysql.com/doc/refman/8.0/en/repair-table.html) table-maintenance statements.

* [LOCK TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_lock-tables)

Enables use of explicit [LOCK TABLES](https://dev.mysql.com/doc/refman/8.0/en/lock-tables.html) statements to lock tables for which you have the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege. This includes use of write locks, which prevents other sessions from reading the locked table.

* [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_process)

The [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) privilege controls access to information about threads executing within the server (that is, information about statements being executed by sessions). Thread information available using the [SHOW PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html) statement, the [**mysqladmin processlist**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command, the [INFORMATION\_SCHEMA.PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/information-schema-processlist-table.html) table, and the Performance Schema [processlist](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-processlist-table.html) table is accessible as follows:

* + With the [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) privilege, a user has access to information about all threads, even those belonging to other users.
  + Without the [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) privilege, nonanonymous users have access to information about their own threads but not threads for other users, and anonymous users have no access to thread information.

**Note**

The Performance Schema [threads](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-threads-table.html) table also provides thread information, but table access uses a different privilege model..

The [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) privilege also enables use of the [SHOW ENGINE](https://dev.mysql.com/doc/refman/8.0/en/show-engine.html) statement, access to the INFORMATION\_SCHEMA InnoDB tables (tables with names that begin with INNODB\_), and (as of MySQL 8.0.21) access to the INFORMATION\_SCHEMA [FILES](https://dev.mysql.com/doc/refman/8.0/en/information-schema-files-table.html) table.

* [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_proxy)

Enables one user to impersonate or become known as another user..

* [REFERENCES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_references)

Creation of a foreign key constraint requires the [REFERENCES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_references) privilege for the parent table.

* [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_reload)

The [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) enables the following operations:

* + Use of the [FLUSH](https://dev.mysql.com/doc/refman/8.0/en/flush.html) statement.
  + Use of [**mysqladmin**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) commands that are equivalent to [FLUSH](https://dev.mysql.com/doc/refman/8.0/en/flush.html) operations: flush-hosts, flush-logs, flush-privileges, flush-status, flush-tables, flush-threads, refresh, and reload.

The reload command tells the server to reload the grant tables into memory. flush-privileges is a synonym for reload. The refresh command closes and reopens the log files and flushes all tables. The other flush-***xxx*** commands perform functions similar to refresh, but are more specific and may be preferable in some instances. For example, if you want to flush just the log files, flush-logs is a better choice than refresh.

* + Use of [**mysqldump**](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html) options that perform various [FLUSH](https://dev.mysql.com/doc/refman/8.0/en/flush.html) operations: [--flush-logs](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html#option_mysqldump_flush-logs) and [--master-data](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html#option_mysqldump_master-data).
  + Use of the [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) and [RESET REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) statements.

* [REPLICATION CLIENT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_replication-client)

Enables use of the [SHOW MASTER STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-master-status.html), [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html), and [SHOW BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/show-binary-logs.html) statements. Grant this privilege to accounts that are used by replicas to connect to the current server as their replication source server.

* [REPLICATION SLAVE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_replication-slave)

Enables the account to request updates that have been made to databases on the replication source server, using the [SHOW REPLICAS | SHOW SLAVE HOSTS](https://dev.mysql.com/doc/refman/8.0/en/show-replicas.html), [SHOW RELAYLOG EVENTS](https://dev.mysql.com/doc/refman/8.0/en/show-relaylog-events.html), and [SHOW BINLOG EVENTS](https://dev.mysql.com/doc/refman/8.0/en/show-binlog-events.html) statements. This privilege is also required to use the [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) options [--read-from-remote-server](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html#option_mysqlbinlog_read-from-remote-server) (-R) and [--read-from-remote-master](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html#option_mysqlbinlog_read-from-remote-master). Grant this privilege to accounts that are used by replicas to connect to the current server as their replication source server.

* [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_select)

Enables rows to be selected from tables in a database. [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) statements require the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege only if they actually access tables. Some [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) statements do not access tables and can be executed without permission for any database. For example, you can use [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) as a simple calculator to evaluate expressions that make no reference to tables:

SELECT 1+1;

SELECT PI()\*2;

The [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege is also needed for other statements that read column values. For example, [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) is needed for columns referenced on the right hand side of ***col\_name***=***expr*** assignment in [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html) statements or for columns named in the WHERE clause of [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) or [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html) statements.

The [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege is needed for tables or views used with [EXPLAIN](https://dev.mysql.com/doc/refman/8.0/en/explain.html), including any underlying tables in view definitions.

* [SHOW DATABASES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_show-databases)

Enables the account to see database names by issuing the SHOW DATABASE statement. Accounts that do not have this privilege see only databases for which they have some privileges, and cannot use the statement at all if the server was started with the [--skip-show-database](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-show-database) option.

**Caution**

Because any static global privilege is considered a privilege for all databases, any static global privilege enables a user to see all database names with [SHOW DATABASES](https://dev.mysql.com/doc/refman/8.0/en/show-databases.html) or by examining the [SCHEMATA](https://dev.mysql.com/doc/refman/8.0/en/information-schema-schemata-table.html) table of INFORMATION\_SCHEMA, except databases that have been restricted at the database level by partial revokes.

* [SHOW VIEW](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_show-view)

Enables use of the [SHOW CREATE VIEW](https://dev.mysql.com/doc/refman/8.0/en/show-create-view.html) statement. This privilege is also needed for views used with [EXPLAIN](https://dev.mysql.com/doc/refman/8.0/en/explain.html).

* [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_shutdown)

Enables use of the [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/shutdown.html) and [RESTART](https://dev.mysql.com/doc/refman/8.0/en/restart.html) statements, the [**mysqladmin shutdown**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command, and the [mysql\_shutdown()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-shutdown.html) C API function.

* [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_super)

[SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_super) is a powerful and far-reaching privilege and should not be granted lightly. If an account needs to perform only a subset of [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) operations, it may be possible to achieve the desired privilege set by instead granting one or more dynamic privileges, each of which confers more limited capabilities. See [Dynamic Privilege Descriptions](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#privileges-provided-dynamic).

**Note**

[SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) is deprecated, and you should expect it to be removed in a future version of MySQL. See [Migrating Accounts from SUPER to Dynamic Privileges](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#dynamic-privileges-migration-from-super).

[SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) affects the following operations and server behaviors:

* + Enables system variable changes at runtime:
    - Enables server configuration changes to global system variables with [SET GLOBAL](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) and [SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html).

The corresponding dynamic privilege is [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin).

* + - Enables setting restricted session system variables that require a special privilege.

The corresponding dynamic privilege is [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin).

* + See also [Section 5.1.9.1, “System Variable Privileges”](https://dev.mysql.com/doc/refman/8.0/en/system-variable-privileges.html).
  + Enables changes to global transaction characteristics (see [Section 13.3.7, “SET TRANSACTION Statement”](https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html)).

The corresponding dynamic privilege is [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin).

* + Enables the account to start and stop replication, including Group Replication.

The corresponding dynamic privilege is [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) for regular replication, [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) for Group Replication.

* + Enables use of the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) and [CHANGE REPLICATION FILTER](https://dev.mysql.com/doc/refman/8.0/en/change-replication-filter.html) statements.

The corresponding dynamic privilege is [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin).

* + Enables binary log control by means of the [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) and [BINLOG](https://dev.mysql.com/doc/refman/8.0/en/binlog.html) statements.

The corresponding dynamic privilege is [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-admin).

* + Enables setting the effective authorization ID when executing a view or stored program. A user with this privilege can specify any account in the DEFINER attribute of a view or stored program.

The corresponding dynamic privilege is [SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_set-user-id).

* + Enables use of the [CREATE SERVER](https://dev.mysql.com/doc/refman/8.0/en/create-server.html), [ALTER SERVER](https://dev.mysql.com/doc/refman/8.0/en/alter-server.html), and [DROP SERVER](https://dev.mysql.com/doc/refman/8.0/en/drop-server.html) statements.
  + Enables use of the [**mysqladmin debug**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command.
  + Enables InnoDB encryption key rotation.

The corresponding dynamic privilege is [ENCRYPTION\_KEY\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_encryption-key-admin).

* + Enables execution of Version Tokens user-defined functions.

The corresponding dynamic privilege is [VERSION\_TOKEN\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_version-token-admin).

* + Enables granting and revoking roles, use of the WITH ADMIN OPTION clause of the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement, and nonempty <graphml> element content in the result from the [ROLES\_GRAPHML()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_roles-graphml) function.

The corresponding dynamic privilege is [ROLE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_role-admin).

* + Enables control over client connections not permitted to non-[SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) accounts:
    - Enables use of the [KILL](https://dev.mysql.com/doc/refman/8.0/en/kill.html) statement or [**mysqladmin kill**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command to kill threads belonging to other accounts. (An account can always kill its own threads.)
    - The server does not execute [init\_connect](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_init_connect) system variable content when [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) clients connect.
    - The server accepts one connection from a [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) client even if the connection limit configured by the [max\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_connections) system variable is reached.
    - A server in offline mode ([offline\_mode](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_offline_mode) enabled) does not terminate [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) client connections at the next client request, and accepts new connections from [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) clients.
    - Updates can be performed even when the [read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_read_only) system variable is enabled. This applies to explicit table updates, and to use of account-management statements such as [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) that update tables implicitly.

The corresponding dynamic privilege for the preceding connection-control operations is [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin).

You may also need the [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege to create or alter stored functions if binary logging is enabled, as described in [Section 25.7, “Stored Program Binary Logging”](https://dev.mysql.com/doc/refman/8.0/en/stored-programs-logging.html).

* [TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_trigger)

Enables trigger operations. You must have this privilege for a table to create, drop, execute, or display triggers for that table.

When a trigger is activated (by a user who has privileges to execute [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) statements for the table associated with the trigger), trigger execution requires that the user who defined the trigger still have the [TRIGGER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_trigger) privilege for the table.

* [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_update)

Enables rows to be updated in tables in a database.

* [USAGE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_usage)

This privilege specifier stands for “no privileges.” It is used at the global level with [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) to specify clauses such as WITH GRANT OPTION without naming specific account privileges in the privilege list. [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) displays [USAGE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_usage) to indicate that an account has no privileges at a privilege level.

#### Dynamic Privilege Descriptions

Dynamic privileges are defined at runtime, in contrast to static privileges, which are built in to the server. The following list describes each dynamic privilege available in MySQL.

Most dynamic privileges are defined at server startup. Others are defined by a particular component or plugin, as indicated in the privilege descriptions. In such cases, the privilege is unavailable unless the component or plugin that defines it is enabled.

Particular SQL statements might have more specific privilege requirements than indicated here. If so, the description for the statement in question provides the details.

* [APPLICATION\_PASSWORD\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_application-password-admin) (added in MySQL 8.0.14)

For dual-password capability, this privilege enables use of the RETAIN CURRENT PASSWORD and DISCARD OLD PASSWORD clauses for [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statements that apply to your own account. This privilege is required to manipulate your own secondary password because most users require only one password.

If an account is to be permitted to manipulate secondary passwords for all accounts, it should be granted the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege rather than [APPLICATION\_PASSWORD\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_application-password-admin).

For more information about use of dual passwords, see [Section 6.2.15, “Password Management”](https://dev.mysql.com/doc/refman/8.0/en/password-management.html).

* [AUDIT\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_audit-admin)

Enables audit log configuration. This privilege is defined by the audit\_log plugin; see [Section 6.4.5, “MySQL Enterprise Audit”](https://dev.mysql.com/doc/refman/8.0/en/audit-log.html).

* [BACKUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_backup-admin)

Enables execution of the [LOCK INSTANCE FOR BACKUP](https://dev.mysql.com/doc/refman/8.0/en/lock-instance-for-backup.html) statement and access to the Performance Schema [log\_status](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-log-status-table.html) table.

**Note**

Besides [BACKUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_backup-admin), the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege on the [log\_status](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-log-status-table.html) table is also needed for its access.

The [BACKUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_backup-admin) privilege is automatically granted to users with the [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) privilege when performing an in-place upgrade to MySQL 8.0 from an earlier version.

* [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_binlog-admin)

Enables binary log control by means of the [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) and [BINLOG](https://dev.mysql.com/doc/refman/8.0/en/binlog.html) statements.

* [BINLOG\_ENCRYPTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_binlog-encryption-admin)

Enables setting the system variable [binlog\_encryption](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_encryption), which activates or deactivates encryption for binary log files and relay log files. This ability is not provided by the [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-admin), [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin), or [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) privileges. The related system variable [binlog\_rotate\_encryption\_master\_key\_at\_startup](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_rotate_encryption_master_key_at_startup), which rotates the binary log master key automatically when the server is restarted, does not require this privilege.

* [CLONE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_clone-admin)

Enables execution of the CLONE statements. Includes [BACKUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_backup-admin) and [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown) privileges.

* [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_connection-admin)

Enables use of the [KILL](https://dev.mysql.com/doc/refman/8.0/en/kill.html) statement or [**mysqladmin kill**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command to kill threads belonging to other accounts. (An account can always kill its own threads.)

Enables setting system variables related to client connections, or circumventing restrictions related to client connections. [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) applies to the effects of these system variables:

* + [init\_connect](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_init_connect): The server does not execute [init\_connect](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_init_connect) system variable content when [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) clients connect.
  + [max\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_connections): The server accepts one connection from a [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) client even if the connection limit configured by the [max\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_connections) system variable is reached.
  + [offline\_mode](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_offline_mode): A server in offline mode ([offline\_mode](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_offline_mode) enabled) does not terminate [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) client connections at the next client request, and accepts new connections from [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) clients.
  + [read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_read_only): Updates can be performed even when the [read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_read_only) system variable is enabled. This applies to explicit table updates, and to use of account-management statements such as [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) that update tables implicitly.

Enables setting the [thread\_pool\_max\_transactions\_limit](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_thread_pool_max_transactions_limit) system variable.

* [ENCRYPTION\_KEY\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_encryption-key-admin)

Enables InnoDB encryption key rotation.

* [FIREWALL\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_firewall-admin)

Enables a user to administer firewall rules for any user. This privilege is defined by the MYSQL\_FIREWALL plugin; see [Section 6.4.7, “MySQL Enterprise Firewall”](https://dev.mysql.com/doc/refman/8.0/en/firewall.html).

* [FIREWALL\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_firewall-user)

Enables users to update their own firewall rules. This privilege is defined by the MYSQL\_FIREWALL plugin; see [Section 6.4.7, “MySQL Enterprise Firewall”](https://dev.mysql.com/doc/refman/8.0/en/firewall.html).

* [FLUSH\_OPTIMIZER\_COSTS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_flush-optimizer-costs) (added in MySQL 8.0.23)

Enables use of the [FLUSH OPTIMIZER\_COSTS](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-optimizer-costs) statement.

* [FLUSH\_STATUS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_flush-status) (added in MySQL 8.0.23)

Enables use of the [FLUSH STATUS](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-status) statement.

* [FLUSH\_TABLES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_flush-tables) (added in MySQL 8.0.23)

Enables use of the [FLUSH TABLES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-tables) statement.

* [FLUSH\_USER\_RESOURCES](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_flush-user-resources) (added in MySQL 8.0.23)

Enables use of the [FLUSH USER\_RESOURCES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-user-resources) statement.

* [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_group-replication-admin)

Enables the account to start and stop Group Replication using the [START GROUP REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html) and [STOP GROUP REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/stop-group-replication.html) statements, to change the global setting for the [group\_replication\_consistency](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_consistency) system variable, and to use the [group\_replication\_set\_write\_concurrency()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-maximum-consensus.html#udf_group-replication-set-write-concurrency) and [group\_replication\_set\_communication\_protocol()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-communication-protocol.html#udf_group-replication-set-communication-protocol) UDFs. Grant this privilege to accounts that are used to administer servers that are members of a replication group.

* [INNODB\_REDO\_LOG\_ARCHIVE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_innodb-redo-log-archive)

Enables the account to activate and deactivate redo log archiving.

* [INNODB\_REDO\_LOG\_ENABLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_innodb-redo-log-enable)

Enables use of the [ALTER INSTANCE {ENABLE|DISABLE} INNODB REDO\_LOG](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html) statement to enable or disable redo logging. Introduced in MySQL 8.0.21.

See [Disabling Redo Logging](https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html#innodb-disable-redo-logging).

* [NDB\_STORED\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_ndb-stored-user)

Enables the user or role and its privileges to be shared and synchronized between all NDB-enabled MySQL servers as soon as they join a given NDB Cluster. This privilege is available only if the [NDB](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html) storage engine is enabled.

Any changes to or revocations of privileges made for the given user or role are synchronized immediately with all connected MySQL servers (SQL nodes). You should be aware that there is no guarantee that multiple statements affecting privileges originating from different SQL nodes are executed on all SQL nodes in the same order. For this reason, it is highly recommended that all user administration be done from a single designated SQL node.

NDB\_STORED\_USER is a global privilege and must be granted or revoked using ON \*.\*. Trying to set any other scope for this privilege results in an error. This privilege can be given to most application and administrative users, but it cannot be granted to system reserved accounts such as mysql.session@localhost or mysql.infoschema@localhost.

A user that has been granted the NDB\_STORED\_USER privilege is stored in NDB (and thus shared by all SQL nodes), as is a role with this privilege. A user that is merely granted a role that has NDB\_STORED\_USER is not stored in NDB; each NDB stored user must be granted the privilege explicitly.

For more detailed information about how this works in [NDB](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html), see [Section 23.5.12, “Distributed MySQL Privileges with NDB\_STORED\_USER”](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster-privilege-distribution.html).

The NDB\_STORED\_USER privilege is available beginning with NDB 8.0.18.

* [PERSIST\_RO\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_persist-ro-variables-admin)

For users who also have [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin), [PERSIST\_RO\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_persist-ro-variables-admin) enables use of [SET PERSIST\_ONLY](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) to persist global system variables to the mysqld-auto.cnf option file in the data directory. This statement is similar to [SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) but does not modify the runtime global system variable value. This makes [SET PERSIST\_ONLY](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) suitable for configuring read-only system variables that can be set only at server startup.

See also [Section 5.1.9.1, “System Variable Privileges”](https://dev.mysql.com/doc/refman/8.0/en/system-variable-privileges.html).

* [REPLICATION\_APPLIER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_replication-applier)

Enables the account to act as the PRIVILEGE\_CHECKS\_USER for a replication channel, and to execute BINLOG statements in [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) output. Grant this privilege to accounts that are assigned using [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) to provide a security context for replication channels, and to handle replication errors on those channels. As well as the REPLICATION\_APPLIER privilege, you must also give the account the required privileges to execute the transactions received by the replication channel or contained in the [**mysqlbinlog**](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html) output, for example to update the affected tables. For more information, see [Section 17.3.3, “Replication Privilege Checks”](https://dev.mysql.com/doc/refman/8.0/en/replication-privilege-checks.html).

* [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_replication-slave-admin)

Enables the account to connect to the replication source server, start and stop replication using the [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) and [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) statements, and use the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) and [CHANGE REPLICATION FILTER](https://dev.mysql.com/doc/refman/8.0/en/change-replication-filter.html) statements. Grant this privilege to accounts that are used by replicas to connect to the current server as their replication source server. This privilege does not apply to Group Replication; use GROUP\_REPLICATION\_ADMIN for that.

* [RESOURCE\_GROUP\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_resource-group-admin)

Enables resource group management, consisting of creating, altering, and dropping resource groups, and assignment of threads and statements to resource groups. A user with this privilege can perform any operation relating to resource groups.

* [RESOURCE\_GROUP\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_resource-group-user)

Enables assigning threads and statements to resource groups. A user with this privilege can use the [SET RESOURCE GROUP](https://dev.mysql.com/doc/refman/8.0/en/set-resource-group.html) statement and the [RESOURCE\_GROUP](https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-resource-group) optimizer hint.

* [ROLE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_role-admin)

Enables granting and revoking roles, use of the WITH ADMIN OPTION clause of the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement, and nonempty <graphml> element content in the result from the [ROLES\_GRAPHML()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_roles-graphml) function. Required to set the value of the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable.

* [SERVICE\_CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_service-connection-admin)

Enables connections to the network interface that permits only administrative connections (see [Section 5.1.12.1, “Connection Interfaces”](https://dev.mysql.com/doc/refman/8.0/en/connection-interfaces.html)).

* [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_session-variables-admin) (added in MySQL 8.0.14)

For most system variables, setting the session value requires no special privileges and can be done by any user to affect the current session. For some system variables, setting the session value can have effects outside the current session and thus is a restricted operation. For these, the [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) privilege enables the user to set the session value.

If a system variable is restricted and requires a special privilege to set the session value, the variable description indicates that restriction. Examples include [binlog\_format](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_format), [sql\_log\_bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_sql_log_bin), and [sql\_log\_off](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sql_log_off).

Prior to MySQL 8.0.14 when [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) was added, restricted session system variables can be set only by users who have the [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin) or [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege.

The [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) privilege is a subset of the [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin) and [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privileges. A user who has either of those privileges is also permitted to set restricted session variables and effectively has [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) by implication and need not be granted [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) explicitly.

See also [Section 5.1.9.1, “System Variable Privileges”](https://dev.mysql.com/doc/refman/8.0/en/system-variable-privileges.html).

* [SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_set-user-id)

Enables setting the effective authorization ID when executing a view or stored program. A user with this privilege can specify any account as the DEFINER attribute of a view or stored program.

As of MySQL 8.0.22, [SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_set-user-id) also enables overriding security checks designed to prevent operations that (perhaps inadvertently) cause stored objects to become orphaned or that cause adoption of stored objects that are currently orphaned. For details, see [Orphan Stored Objects](https://dev.mysql.com/doc/refman/8.0/en/stored-objects-security.html#stored-objects-security-orphan-objects).

* [SHOW\_ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_show-routine) (added in MySQL 8.0.20)

Enables a user to access definitions and properties of all stored routines (stored procedures and functions), even those for which the user is not named as the routine DEFINER. This access includes:

* + The contents of the [INFORMATION\_SCHEMA.ROUTINES](https://dev.mysql.com/doc/refman/8.0/en/information-schema-routines-table.html) table.
  + The [SHOW CREATE FUNCTION](https://dev.mysql.com/doc/refman/8.0/en/show-create-function.html) and [SHOW CREATE PROCEDURE](https://dev.mysql.com/doc/refman/8.0/en/show-create-procedure.html) statements.
  + The [SHOW FUNCTION CODE](https://dev.mysql.com/doc/refman/8.0/en/show-function-code.html) and [SHOW PROCEDURE CODE](https://dev.mysql.com/doc/refman/8.0/en/show-procedure-code.html) statements.
  + The [SHOW FUNCTION STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-function-status.html) and [SHOW PROCEDURE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-procedure-status.html) statements.

Prior to MySQL 8.0.20, for a user to access definitions of routines the user did not define, the user must have the global [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege, which is very broad. As of 8.0.20, [SHOW\_ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_show-routine) may be granted instead as a privilege with a more restricted scope that permits access to routine definitions. (That is, an administrator can rescind global [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) from users that do not otherwise require it and grant [SHOW\_ROUTINE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_show-routine) instead.) This enables an account to back up stored routines without requiring a broad privilege.

* [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_system-user) (added in MySQL 8.0.16)

The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege distinguishes system users from regular users:

* + A user with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a system user.
  + A user without the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a regular user.

The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege has an effect on the accounts to which a given user can apply its other privileges, as well as whether the user is protected from other accounts:

* + A system user can modify both system and regular accounts. That is, a user who has the appropriate privileges to perform a given operation on regular accounts is enabled by possession of [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) to also perform the operation on system accounts. A system account can be modified only by system users with appropriate privileges, not by regular users.
  + A regular user with appropriate privileges can modify regular accounts, but not system accounts. A regular account can be modified by both system and regular users with appropriate privileges.

For more information, see [Section 6.2.11, “Account Categories”](https://dev.mysql.com/doc/refman/8.0/en/account-categories.html).

The protection against modification by regular accounts that is afforded to system accounts by the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege does not apply to regular accounts that have privileges on the mysql system schema and thus can directly modify the grant tables in that schema. For full protection, do not grant mysql schema privileges to regular accounts. See [Protecting System Accounts Against Manipulation by Regular Accounts](https://dev.mysql.com/doc/refman/8.0/en/account-categories.html#protecting-system-accounts).

* [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_system-variables-admin)

Affects the following operations and server behaviors:

* + Enables system variable changes at runtime:
    - Enables server configuration changes to global system variables with [SET GLOBAL](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) and [SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html).
    - Enables server configuration changes to global system variables with [SET PERSIST\_ONLY](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html), if the user also has [PERSIST\_RO\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_persist-ro-variables-admin).
    - Enables setting restricted session system variables that require a special privilege. In effect, [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin) implies [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin) without explicitly granting [SESSION\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_session-variables-admin).

See also [Section 5.1.9.1, “System Variable Privileges”](https://dev.mysql.com/doc/refman/8.0/en/system-variable-privileges.html).

* + Enables changes to global transaction characteristics (see [Section 13.3.7, “SET TRANSACTION Statement”](https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html)).

* [TABLE\_ENCRYPTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_table-encryption-admin) (added in MySQL 8.0.16)

Enables a user to override default encryption settings when [table\_encryption\_privilege\_check](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_table_encryption_privilege_check) is enabled; see [Defining an Encryption Default for Schemas and General Tablespaces](https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html#innodb-schema-tablespace-encryption-default).

* [VERSION\_TOKEN\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_version-token-admin)

Enables execution of Version Tokens user-defined functions. This privilege is defined by the version\_tokens plugin; see [Section 5.6.6, “Version Tokens”](https://dev.mysql.com/doc/refman/8.0/en/version-tokens.html).

* [XA\_RECOVER\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html" \l "priv_xa-recover-admin)

Enables execution of the [XA RECOVER](https://dev.mysql.com/doc/refman/8.0/en/xa-statements.html) statement; see [Section 13.3.8.1, “XA Transaction SQL Statements”](https://dev.mysql.com/doc/refman/8.0/en/xa-statements.html).

Prior to MySQL 8.0, any user could execute the [XA RECOVER](https://dev.mysql.com/doc/refman/8.0/en/xa-statements.html) statement to discover the XID values for outstanding prepared XA transactions, possibly leading to commit or rollback of an XA transaction by a user other than the one who started it. In MySQL 8.0, [XA RECOVER](https://dev.mysql.com/doc/refman/8.0/en/xa-statements.html) is permitted only to users who have the [XA\_RECOVER\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_xa-recover-admin) privilege, which is expected to be granted only to administrative users who have need for it. This might be the case, for example, for administrators of an XA application if it has crashed and it is necessary to find outstanding transactions started by the application so they can be rolled back. This privilege requirement prevents users from discovering the XID values for outstanding prepared XA transactions other than their own. It does not affect normal commit or rollback of an XA transaction because the user who started it knows its XID.

#### Privilege-Granting Guidelines

It is a good idea to grant to an account only those privileges that it needs. You should exercise particular caution in granting the [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) and administrative privileges:

* [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) can be abused to read into a database table any files that the MySQL server can read on the server host. This includes all world-readable files and files in the server's data directory. The table can then be accessed using [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) to transfer its contents to the client host.
* [GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_grant-option) enables users to give their privileges to other users. Two users that have different privileges and with the [GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_grant-option) privilege are able to combine privileges.
* [ALTER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_alter) may be used to subvert the privilege system by renaming tables.
* [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown) can be abused to deny service to other users entirely by terminating the server.
* [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) can be used to view the plain text of currently executing statements, including statements that set or change passwords.
* [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) can be used to terminate other sessions or change how the server operates.
* Privileges granted for the mysql system database itself can be used to change passwords and other access privilege information:
  + Passwords are stored encrypted, so a malicious user cannot simply read them to know the plain text password. However, a user with write access to the mysql.user system table authentication\_string column can change an account's password, and then connect to the MySQL server using that account.
  + [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) or [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) granted for the mysql system database enable a user to add privileges or modify existing privileges, respectively.
  + [DROP](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop) for the mysql system database enables a user to remote privilege tables, or even the database itself.

#### Static Versus Dynamic Privileges

MySQL supports static and dynamic privileges:

* Static privileges are built in to the server. They are always available to be granted to user accounts and cannot be unregistered.
* Dynamic privileges can be registered and unregistered at runtime. This affects their availability: A dynamic privilege that has not been registered cannot be granted.

For example, the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) and [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) privileges are static and always available, whereas a dynamic privilege becomes available only if the component that implements it has been enabled.

The remainder of this section describes how dynamic privileges work in MySQL. The discussion uses the term “components” but applies equally to plugins.

**Note**

Server administrators should be aware of which server components define dynamic privileges. For MySQL distributions, documentation of components that define dynamic privileges describes those privileges.

Third-party components may also define dynamic privileges; an administrator should understand those privileges and not install components that might conflict or compromise server operation. For example, one component conflicts with another if both define a privilege with the same name. Component developers can reduce the likelihood of this occurrence by choosing privilege names having a prefix based on the component name.

The server maintains the set of registered dynamic privileges internally in memory. Unregistration occurs at server shutdown.

Normally, a component that defines dynamic privileges registers them when it is installed, during its initialization sequence. When uninstalled, a component does not unregister its registered dynamic privileges. (This is current practice, not a requirement. That is, components could, but do not, unregister at any time privileges they register.)

No warning or error occurs for attempts to register an already registered dynamic privilege. Consider the following sequence of statements:

INSTALL COMPONENT 'my\_component';

UNINSTALL COMPONENT 'my\_component';

INSTALL COMPONENT 'my\_component';

The first [INSTALL COMPONENT](https://dev.mysql.com/doc/refman/8.0/en/install-component.html) statement registers any privileges defined by component my\_component, but [UNINSTALL COMPONENT](https://dev.mysql.com/doc/refman/8.0/en/uninstall-component.html) does not unregister them. For the second [INSTALL COMPONENT](https://dev.mysql.com/doc/refman/8.0/en/install-component.html) statement, the component privileges it registers are found to be already registered, but no warnings or errors occur.

Dynamic privileges apply only at the global level. The server stores information about current assignments of dynamic privileges to user accounts in the mysql.global\_grants system table:

* The server automatically registers privileges named in global\_grants during server startup (unless the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option is given).
* The [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements modify the contents of global\_grants.
* Dynamic privilege assignments listed in global\_grants are persistent. They are not removed at server shutdown.

Example: The following statement grants to user u1 the privileges required to control replication (including Group Replication) on a replica, and to modify system variables:

GRANT REPLICATION\_SLAVE\_ADMIN, GROUP\_REPLICATION\_ADMIN, BINLOG\_ADMIN

ON \*.\* TO 'u1'@'localhost';

Granted dynamic privileges appear in the output from the SHOW GRANTS statement and the INFORMATION\_SCHEMA [USER\_PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/information-schema-user-privileges-table.html) table.

For [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) at the global level, any named privileges not recognized as static are checked against the current set of registered dynamic privileges and granted if found. Otherwise, an error occurs to indicate an unknown privilege identifier.

For [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) the meaning of ALL [PRIVILEGES] at the global level includes all static global privileges, as well as all currently registered dynamic privileges:

* GRANT ALL at the global level grants all static global privileges and all currently registered dynamic privileges. A dynamic privilege registered subsequent to execution of the GRANT statement is not granted retroactively to any account.
* REVOKE ALL at the global level revokes all granted static global privileges and all granted dynamic privileges.

The [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement reads the global\_grants table for dynamic privilege assignments and registers any unregistered privileges found there.

#### Migrating Accounts from SUPER to Dynamic Privileges

In MySQL 8.0, many operations that previously required the [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege are also associated with a dynamic privilege of more limited scope Each such operation can be permitted to an account by granting the associated dynamic privilege rather than [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super). This change improves security by enabling DBAs to avoid granting [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) and tailor user privileges more closely to the operations permitted. [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) is now deprecated; expect it to be removed in a future version of MySQL.

When removal of [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) occurs, operations that formerly required [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) fail unless accounts granted [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) are migrated to the appropriate dynamic privileges. Use the following instructions to accomplish that goal so that accounts are ready prior to [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) removal:

1. Execute this query to identify accounts that are granted [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super):
2. SELECT GRANTEE FROM INFORMATION\_SCHEMA.USER\_PRIVILEGES

WHERE PRIVILEGE\_TYPE = 'SUPER';

1. For each account identified by the preceding query, determine the operations for which it needs [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super). Then grant the dynamic privileges corresponding to those operations, and revoke [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super).

For example, if 'u1'@'localhost' requires [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) for binary log purging and system variable modification, these statements make the required changes to the account:

GRANT BINLOG\_ADMIN, SYSTEM\_VARIABLES\_ADMIN ON \*.\* TO 'u1'@'localhost';

REVOKE SUPER ON \*.\* FROM 'u1'@'localhost';

After you have modified all applicable accounts, the INFORMATION\_SCHEMA query in the first step should produce an empty result set.

### Grant Tables

The mysql system database includes several grant tables that contain information about user accounts and the privileges held by them. This section describes those tables. For information about other tables in the system database, see [Section 5.3, “The mysql System Schema”](https://dev.mysql.com/doc/refman/8.0/en/system-schema.html).

The discussion here describes the underlying structure of the grant tables and how the server uses their contents when interacting with clients. However, normally you do not modify the grant tables directly. Modifications occur indirectly when you use account-management statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) to set up accounts and control the privileges available to each one. See [Section 13.7.1, “Account Management Statements”](https://dev.mysql.com/doc/refman/8.0/en/account-management-statements.html). When you use such statements to perform account manipulations, the server modifies the grant tables on your behalf.

**Note**

Direct modification of grant tables using statements such as [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) is discouraged and done at your own risk. The server is free to ignore rows that become malformed as a result of such modifications.

For any operation that modifies a grant table, the server checks whether the table has the expected structure and produces an error if not. To update the tables to the expected structure, perform the MySQL upgrade procedure..

#### Grant Table Overview

These mysql database tables contain grant information:

* [user](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html#grant-tables-user-db): User accounts, static global privileges, and other nonprivilege columns.

* [global\_grants](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-global-grants" \o "The global_grants Grant Table): Dynamic global privileges.

* [db](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-user-db" \o "The user and db Grant Tables): Database-level privileges.

* [tables\_priv](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-tables-priv-columns-priv" \o "The tables_priv and columns_priv Grant Tables): Table-level privileges.

* [columns\_priv](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-tables-priv-columns-priv" \o "The tables_priv and columns_priv Grant Tables): Column-level privileges.

* [procs\_priv](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-procs-priv" \o "The procs_priv Grant Table): Stored procedure and function privileges.

* [proxies\_priv](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-proxies-priv" \o "The proxies_priv Grant Table): Proxy-user privileges.

* [default\_roles](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-default-roles" \o "The default_roles Grant Table): Default user roles.

* [role\_edges](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-role-edges" \o "The role_edges Grant Table): Edges for role subgraphs.

* [password\_history](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html" \l "grant-tables-password-history" \o "The password_history Grant Table): Password change history.

For information about the differences between static and dynamic global privileges, see [Static Versus Dynamic Privileges](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#static-dynamic-privileges).)

In MySQL 8.0, grant tables use the InnoDB storage engine and are transactional. Before MySQL 8.0, grant tables used the MyISAM storage engine and were nontransactional. This change of grant table storage engine enables an accompanying change to the behavior of account-management statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) or [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html). Previously, an account-management statement that named multiple users could succeed for some users and fail for others. Now, each statement is transactional and either succeeds for all named users or rolls back and has no effect if any error occurs.

Each grant table contains scope columns and privilege columns:

* Scope columns determine the scope of each row in the tables; that is, the context in which the row applies. For example, a user table row with Host and User values of 'h1.example.net' and 'bob' applies to authenticating connections made to the server from the host h1.example.net by a client that specifies a user name of bob. Similarly, a db table row with Host, User, and Db column values of 'h1.example.net', 'bob' and 'reports' applies when bob connects from the host h1.example.net to access the reports database. The tables\_priv and columns\_priv tables contain scope columns indicating tables or table/column combinations to which each row applies. The procs\_priv scope columns indicate the stored routine to which each row applies.
* Privilege columns indicate which privileges a table row grants; that is, which operations it permits to be performed. The server combines the information in the various grant tables to form a complete description of a user's privileges. [Section 6.2.7, “Access Control, Stage 2: Request Verification”](https://dev.mysql.com/doc/refman/8.0/en/request-access.html), describes the rules for this.

In addition, a grant table may contain columns used for purposes other than scope or privilege assessment.

The server uses the grant tables in the following manner:

* The user table scope columns determine whether to reject or permit incoming connections. For permitted connections, any privileges granted in the user table indicate the user's static global privileges. Any privileges granted in this table apply to all databases on the server.

**Caution**

Because any static global privilege is considered a privilege for all databases, any static global privilege enables a user to see all database names with [SHOW DATABASES](https://dev.mysql.com/doc/refman/8.0/en/show-databases.html) or by examining the [SCHEMATA](https://dev.mysql.com/doc/refman/8.0/en/information-schema-schemata-table.html) table of INFORMATION\_SCHEMA, except databases that have been restricted at the database level by partial revokes.

* The global\_grants table lists current assignments of dynamic global privileges to user accounts. For each row, the scope columns determine which user has the privilege named in the privilege column.
* The db table scope columns determine which users can access which databases from which hosts. The privilege columns determine the permitted operations. A privilege granted at the database level applies to the database and to all objects in the database, such as tables and stored programs.
* The tables\_priv and columns\_priv tables are similar to the db table, but are more fine-grained: They apply at the table and column levels rather than at the database level. A privilege granted at the table level applies to the table and to all its columns. A privilege granted at the column level applies only to a specific column.
* The procs\_priv table applies to stored routines (stored procedures and functions). A privilege granted at the routine level applies only to a single procedure or function.
* The proxies\_priv table indicates which users can act as proxies for other users and whether a user can grant the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege to other users.
* The default\_roles and role\_edges tables contain information about role relationships.
* The password\_history table retains previously chosen passwords to enable restrictions on password reuse. See [Section 6.2.15, “Password Management”](https://dev.mysql.com/doc/refman/8.0/en/password-management.html).

The server reads the contents of the grant tables into memory when it starts. You can tell it to reload the tables by issuing a [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement or executing a [**mysqladmin flush-privileges**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) or [**mysqladmin reload**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command. Changes to the grant tables take effect as indicated in [Section 6.2.13, “When Privilege Changes Take Effect”](https://dev.mysql.com/doc/refman/8.0/en/privilege-changes.html).

When you modify an account, it is a good idea to verify that your changes have the intended effect. To check the privileges for a given account, use the [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) statement. For example, to determine the privileges that are granted to an account with user name and host name values of bob and pc84.example.com, use this statement:

SHOW GRANTS FOR 'bob'@'pc84.example.com';

To display nonprivilege properties of an account, use [SHOW CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/show-create-user.html):

SHOW CREATE USER 'bob'@'pc84.example.com';

#### The user and db Grant Tables

The server uses the user and db tables in the mysql database at both the first and second stages of access control. The columns in the user and db tables are shown here.

**Table 6.4 user and db Table Columns**

| **Table Name** | user | db |
| --- | --- | --- |
| ***Scope columns*** | Host | Host |
|  | User | Db |
|  |  | User |
| ***Privilege columns*** | Select\_priv | Select\_priv |
|  | Insert\_priv | Insert\_priv |
|  | Update\_priv | Update\_priv |
|  | Delete\_priv | Delete\_priv |
|  | Index\_priv | Index\_priv |
|  | Alter\_priv | Alter\_priv |
|  | Create\_priv | Create\_priv |
|  | Drop\_priv | Drop\_priv |
|  | Grant\_priv | Grant\_priv |
|  | Create\_view\_priv | Create\_view\_priv |
|  | Show\_view\_priv | Show\_view\_priv |
|  | Create\_routine\_priv | Create\_routine\_priv |
|  | Alter\_routine\_priv | Alter\_routine\_priv |
|  | Execute\_priv | Execute\_priv |
|  | Trigger\_priv | Trigger\_priv |
|  | Event\_priv | Event\_priv |
|  | Create\_tmp\_table\_priv | Create\_tmp\_table\_priv |
|  | Lock\_tables\_priv | Lock\_tables\_priv |
|  | References\_priv | References\_priv |
|  | Reload\_priv |  |
|  | Shutdown\_priv |  |
|  | Process\_priv |  |
|  | File\_priv |  |
|  | Show\_db\_priv |  |
|  | Super\_priv |  |
|  | Repl\_slave\_priv |  |
|  | Repl\_client\_priv |  |
|  | Create\_user\_priv |  |
|  | Create\_tablespace\_priv |  |
|  | Create\_role\_priv |  |
|  | Drop\_role\_priv |  |
| ***Security columns*** | ssl\_type |  |
|  | ssl\_cipher |  |
|  | x509\_issuer |  |
|  | x509\_subject |  |
|  | plugin |  |
|  | authentication\_string |  |
|  | password\_expired |  |
|  | password\_last\_changed |  |
|  | password\_lifetime |  |
|  | account\_locked |  |
|  | Password\_reuse\_history |  |
|  | Password\_reuse\_time |  |
|  | Password\_require\_current |  |
|  | User\_attributes |  |
| ***Resource control columns*** | max\_questions |  |
|  | max\_updates |  |
|  | max\_connections |  |
|  | max\_user\_connections |  |

The user table plugin and authentication\_string columns store authentication plugin and credential information.

The server uses the plugin named in the plugin column of an account row to authenticate connection attempts for the account.

The plugin column must be nonempty. At startup, and at runtime when [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) is executed, the server checks user table rows. For any row with an empty plugin column, the server writes a warning to the error log of this form:

[Warning] User entry 'user\_name'@'host\_name' has an empty plugin

value. The user will be ignored and no one can login with this user

anymore.

To assign a plugin to an account that is missing one, use the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement.

The password\_expired column permits DBAs to expire account passwords and require users to reset their password. The default password\_expired value is 'N', but can be set to 'Y' with the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement. After an account's password has been expired, all operations performed by the account in subsequent connections to the server result in an error until the user issues an [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement to establish a new account password.

**Note**

Although it is possible to “reset” an expired password by setting it to its current value, it is preferable, as a matter of good policy, to choose a different password. DBAs can enforce non-reuse by establishing an appropriate password-reuse policy. See [Password Reuse Policy](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#password-reuse-policy).

password\_last\_changed is a TIMESTAMP column indicating when the password was last changed. The value is non-NULL only for accounts that use a MySQL built-in authentication plugin (mysql\_native\_password, sha256\_password, or caching\_sha2\_password). The value is NULL for other accounts, such as those authenticated using an external authentication system.

password\_last\_changed is updated by the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statements, and by [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements that create an account or change an account password.

password\_lifetime indicates the account password lifetime, in days. If the password is past its lifetime (assessed using the password\_last\_changed column), the server considers the password expired when clients connect using the account. A value of ***N*** greater than zero means that the password must be changed every ***N*** days. A value of 0 disables automatic password expiration. If the value is NULL (the default), the global expiration policy applies, as defined by the [default\_password\_lifetime](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_password_lifetime) system variable.

account\_locked indicates whether the account is locked (see [Section 6.2.19, “Account Locking”](https://dev.mysql.com/doc/refman/8.0/en/account-locking.html)).

Password\_reuse\_history is the value of the PASSWORD HISTORY option for the account, or NULL for the default history.

Password\_reuse\_time is the value of the PASSWORD REUSE INTERVAL option for the account, or NULL for the default interval.

Password\_require\_current (added in MySQL 8.0.13) corresponds to the value of the PASSWORD REQUIRE option for the account, as shown by the following table.

**Table 6.5 Permitted Password\_require\_current Values**

| **Password\_require\_current Value** | **Corresponding PASSWORD REQUIRE Option** |
| --- | --- |
| 'Y' | PASSWORD REQUIRE CURRENT |
| 'N' | PASSWORD REQUIRE CURRENT OPTIONAL |
| NULL | PASSWORD REQUIRE CURRENT DEFAULT |

User\_attributes (added in MySQL 8.0.14) is a JSON-format column that stores account attributes not stored in other columns:

* additional\_password: The secondary password, if any. See [Dual Password Support](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#dual-passwords).
* Restrictions: Restriction lists, if any. Restrictions are added by partial-revoke operations. The attribute value is an array of elements that each have Database and Restrictions keys indicating the name of a restricted database and the applicable restrictions on it (see [Section 6.2.12, “Privilege Restriction Using Partial Revokes”](https://dev.mysql.com/doc/refman/8.0/en/partial-revokes.html)).
* Password\_locking: The conditions for failed-login tracking and temporary account locking, if any (see [Failed-Login Tracking and Temporary Account Locking](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#failed-login-tracking)). The Password\_locking attribute is updated according to the FAILED\_LOGIN\_ATTEMPTS and PASSWORD\_LOCK\_TIME options of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements. The attribute value is a hash with failed\_login\_attempts and password\_lock\_time\_days keys indicating the value of such options as have been specified for the account. If a key is missing, its value is implicitly 0. If a key value is implicitly or explicitly 0, the corresponding capability is disabled. This attribute was added in MySQL 8.0.19.

If no attributes apply, User\_attributes is NULL.

Example: An account that has a secondary password and partially revoked database privileges has additional\_password and Restrictions attributes in the column value:

mysql> SELECT User\_attributes FROM mysql.User WHERE User = 'u'\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

User\_attributes: {"Restrictions":

[{"Database": "mysql", "Privileges": ["SELECT"]}],

"additional\_password": "hashed\_credentials"}

To determine which attributes are present, use the [JSON\_KEYS()](https://dev.mysql.com/doc/refman/8.0/en/json-search-functions.html#function_json-keys) function:

SELECT User, Host, JSON\_KEYS(User\_attributes)

FROM mysql.user WHERE User\_attributes IS NOT NULL;

To extract a particular attribute, such as Restrictions, do this:

SELECT User, Host, User\_attributes->>'$.Restrictions'

FROM mysql.user WHERE User\_attributes->>'$.Restrictions' <> '';

#### The tables\_priv and columns\_priv Grant Tables

During the second stage of access control, the server performs request verification to ensure that each client has sufficient privileges for each request that it issues. In addition to the user and db grant tables, the server may also consult the tables\_priv and columns\_priv tables for requests that involve tables. The latter tables provide finer privilege control at the table and column levels. They have the columns shown in the following table.

**Table 6.6 tables\_priv and columns\_priv Table Columns**

| **Table Name** | tables\_priv | columns\_priv |
| --- | --- | --- |
| ***Scope columns*** | Host | Host |
|  | Db | Db |
|  | User | User |
|  | Table\_name | Table\_name |
|  |  | Column\_name |
| ***Privilege columns*** | Table\_priv | Column\_priv |
|  | Column\_priv |  |
| ***Other columns*** | Timestamp | Timestamp |
|  | Grantor |  |

The Timestamp and Grantor columns are set to the current timestamp and the [CURRENT\_USER](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) value, respectively, but are otherwise unused.

#### The procs\_priv Grant Table

For verification of requests that involve stored routines, the server may consult the procs\_priv table, which has the columns shown in the following table.

**Table 6.7 procs\_priv Table Columns**

| **Table Name** | procs\_priv |
| --- | --- |
| ***Scope columns*** | Host |
|  | Db |
|  | User |
|  | Routine\_name |
|  | Routine\_type |
| ***Privilege columns*** | Proc\_priv |
| ***Other columns*** | Timestamp |
|  | Grantor |

The Routine\_type column is an [ENUM](https://dev.mysql.com/doc/refman/8.0/en/enum.html) column with values of 'FUNCTION' or 'PROCEDURE' to indicate the type of routine the row refers to. This column enables privileges to be granted separately for a function and a procedure with the same name.

The Timestamp and Grantor columns are unused.

#### The proxies\_priv Grant Table

The proxies\_priv table records information about proxy accounts. It has these columns:

* Host, User: The proxy account; that is, the account that has the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for the proxied account.
* Proxied\_host, Proxied\_user: The proxied account.
* Grantor, Timestamp: Unused.
* With\_grant: Whether the proxy account can grant the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege to other accounts.

For an account to be able to grant the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege to other accounts, it must have a row in the proxies\_priv table with With\_grant set to 1 and Proxied\_host and Proxied\_user set to indicate the account or accounts for which the privilege can be granted. For example, the 'root'@'localhost' account created during MySQL installation has a row in the proxies\_priv table that enables granting the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for ''@'', that is, for all users and all hosts. This enables root to set up proxy users, as well as to delegate to other accounts the authority to set up proxy users. See [Section 6.2.18, “Proxy Users”](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html).

#### The global\_grants Grant Table

The global\_grants table lists current assignments of dynamic global privileges to user accounts. The table has these columns:

* USER, HOST: The user name and host name of the account to which the privilege is granted.
* PRIV: The privilege name.
* WITH\_GRANT\_OPTION: Whether the account can grant the privilege to other accounts.

#### The default\_roles Grant Table

The default\_roles table lists default user roles. It has these columns:

* HOST, USER: The account or role to which the default role applies.
* DEFAULT\_ROLE\_HOST, DEFAULT\_ROLE\_USER: The default role.

#### The role\_edges Grant Table

The role\_edges table lists edges for role subgraphs. It has these columns:

* FROM\_HOST, FROM\_USER: The account that is granted a role.
* TO\_HOST, TO\_USER: The role that is granted to the account.
* WITH\_ADMIN\_OPTION: Whether the account can grant the role to and revoke it from other accounts by using WITH ADMIN OPTION.

#### The password\_history Grant Table

The password\_history table contains information about password changes. It has these columns:

* Host, User: The account for which the password change occurred.
* Password\_timestamp: The time when the password change occurred.
* Password: The new password hash value.

The password\_history table accumulates a sufficient number of nonempty passwords per account to enable MySQL to perform checks against both the account password history length and reuse interval. Automatic pruning of entries that are outside both limits occurs when password-change attempts occur.

**Note**

The empty password does not count in the password history and is subject to reuse at any time.

If an account is renamed, its entries are renamed to match. If an account is dropped or its authentication plugin is changed, its entries are removed.

#### Grant Table Scope Column Properties

Scope columns in the grant tables contain strings. The default value for each is the empty string. The following table shows the number of characters permitted in each column.

**Table 6.8 Grant Table Scope Column Lengths**

| **Column Name** | **Maximum Permitted Characters** |
| --- | --- |
| Host, Proxied\_host | 255 (60 prior to MySQL 8.0.17) |
| User, Proxied\_user | 32 |
| Db | 64 |
| Table\_name | 64 |
| Column\_name | 64 |
| Routine\_name | 64 |

Host and Proxied\_host values are converted to lowercase before being stored in the grant tables.

For access-checking purposes, comparisons of User, Proxied\_user, authentication\_string, Db, and Table\_name values are case-sensitive. Comparisons of Host, Proxied\_host, Column\_name, and Routine\_name values are not case-sensitive.

#### Grant Table Privilege Column Properties

The user and db tables list each privilege in a separate column that is declared as ENUM('N','Y') DEFAULT 'N'. In other words, each privilege can be disabled or enabled, with the default being disabled.

The tables\_priv, columns\_priv, and procs\_priv tables declare the privilege columns as [SET](https://dev.mysql.com/doc/refman/8.0/en/set.html) columns. Values in these columns can contain any combination of the privileges controlled by the table. Only those privileges listed in the column value are enabled.

**Table 6.9 Set-Type Privilege Column Values**

| **Table Name** | **Column Name** | **Possible Set Elements** |
| --- | --- | --- |
| tables\_priv | Table\_priv | 'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger' |
| tables\_priv | Column\_priv | 'Select', 'Insert', 'Update', 'References' |
| columns\_priv | Column\_priv | 'Select', 'Insert', 'Update', 'References' |
| procs\_priv | Proc\_priv | 'Execute', 'Alter Routine', 'Grant' |

Only the user and global\_grants tables specify administrative privileges, such as [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload), [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown), and [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin). Administrative operations are operations on the server itself and are not database-specific, so there is no reason to list these privileges in the other grant tables. Consequently, the server need consult only the user and global\_grants tables to determine whether a user can perform an administrative operation.

The [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege also is specified only in the user table. It is not an administrative privilege as such, but a user's ability to read or write files on the server host is independent of the database being accessed.

#### Grant Table Concurrency

As of MySQL 8.0.22, to permit concurrent DML and DDL operations on MySQL grant tables, read operations that previously acquired row locks on MySQL grant tables are executed as non-locking reads. Operations that are performed as non-locking reads on MySQL grant tables include:

* [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) statements and other read-only statements that read data from grant tables through join lists and subqueries, including [SELECT ... FOR SHARE](https://dev.mysql.com/doc/refman/8.0/en/innodb-locking-reads.html) statements, using any transaction isolation level.
* DML operations that read data from grant tables (through join lists or subqueries) but do not modify them, using any transaction isolation level.

Statements that no longer acquire row locks when reading data from grant tables report a warning if executed while using statement-based replication.

When using -[binlog\_format=mixed](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_format), DML operations that read data from grant tables are written to the binary log as row events to make the operations safe for mixed-mode replication.

[SELECT ... FOR SHARE](https://dev.mysql.com/doc/refman/8.0/en/innodb-locking-reads.html) statements that read data from grant tables report a warning. With the FOR SHARE clause, read locks are not supported on grant tables.

DML operations that read data from grant tables and are executed using the SERIALIZABLE isolation level report a warning. Read locks that would normally be acquired when using the SERIALIZABLE isolation level are not supported on grant tables.

### Specifying Account Names

MySQL account names consist of a user name and a host name, which enables creation of distinct accounts for users with the same user name who can connect from different hosts. This section describes how to write account names, including special values and wildcard rules.

MySQL role names are similar to account names, with some differences described at [Section 6.2.5, “Specifying Role Names”](https://dev.mysql.com/doc/refman/8.0/en/role-names.html).

In SQL statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html), account names follow these rules:

* Account name syntax is '***user\_name***'@'***host\_name***'.
* An account name consisting only of a user name is equivalent to '***user\_name***'@'%'. For example, 'me' is equivalent to 'me'@'%'.
* The user name and host name need not be quoted if they are legal as unquoted identifiers. Quotes are necessary to specify a ***user\_name*** string containing special characters (such as space or -), or a ***host\_name*** string containing special characters or wildcard characters (such as . or %). For example, in the account name 'test-user'@'%.com', both the user name and host name parts require quotes.
* Quote user names and host names as identifiers or as strings, using either backticks (`), single quotation marks ('), or double quotation marks ("). For string-quoting and identifier-quoting guidelines, see [Section 9.1.1, “String Literals”](https://dev.mysql.com/doc/refman/8.0/en/string-literals.html), and [Section 9.2, “Schema Object Names”](https://dev.mysql.com/doc/refman/8.0/en/identifiers.html).
* The user name and host name parts, if quoted, must be quoted separately. That is, write 'me'@'localhost', not 'me@localhost'. The latter is actually equivalent to 'me@localhost'@'%'.
* A reference to the [CURRENT\_USER](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) or [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) function is equivalent to specifying the current client's user name and host name literally.

MySQL stores account names in grant tables in the mysql system database using separate columns for the user name and host name parts:

* The user table contains one row for each account. The User and Host columns store the user name and host name. This table also indicates which global privileges the account has.
* Other grant tables indicate privileges an account has for databases and objects within databases. These tables have User and Host columns to store the account name. Each row in these tables associates with the account in the user table that has the same User and Host values.
* For access-checking purposes, comparisons of User values are case-sensitive. Comparisons of Host values are not case sensitive.

For additional detail about the properties of user names and host names as stored in the grant tables, such as maximum length, see [Grant Table Scope Column Properties](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html#grant-tables-scope-column-properties).

User names and host names have certain special values or wildcard conventions, as described following.

The user name part of an account name is either a nonblank value that literally matches the user name for incoming connection attempts, or a blank value (empty string) that matches any user name. An account with a blank user name is an anonymous user. To specify an anonymous user in SQL statements, use a quoted empty user name part, such as ''@'localhost'.

The host name part of an account name can take many forms, and wildcards are permitted:

* A host value can be a host name or an IP address (IPv4 or IPv6). The name 'localhost' indicates the local host. The IP address '127.0.0.1' indicates the IPv4 loopback interface. The IP address '::1' indicates the IPv6 loopback interface.
* The % and \_ wildcard characters are permitted in host name or IP address values. These have the same meaning as for pattern-matching operations performed with the [LIKE](https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html#operator_like) operator. For example, a host value of '%' matches any host name, whereas a value of '%.mysql.com' matches any host in the mysql.com domain. '198.51.100.%' matches any host in the 198.51.100 class C network.

Because IP wildcard values are permitted in host values (for example, '198.51.100.%' to match every host on a subnet), someone could try to exploit this capability by naming a host 198.51.100.somewhere.com. To foil such attempts, MySQL does not perform matching on host names that start with digits and a dot. For example, if a host is named 1.2.example.com, its name never matches the host part of account names. An IP wildcard value can match only IP addresses, not host names.

* For a host value specified as an IPv4 address, a netmask can be given to indicate how many address bits to use for the network number. Netmask notation cannot be used for IPv6 addresses.

The syntax is ***host\_ip***/***netmask***. For example:

CREATE USER 'david'@'198.51.100.0/255.255.255.0';

This enables david to connect from any client host having an IP address ***client\_ip*** for which the following condition is true:

client\_ip & netmask = host\_ip

That is, for the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement just shown:

client\_ip & 255.255.255.0 = 198.51.100.0

IP addresses that satisfy this condition range from 198.51.100.0 to 198.51.100.255.

A netmask typically begins with bits set to 1, followed by bits set to 0. Examples:

* + 198.0.0.0/255.0.0.0: Any host on the 198 class A network
  + 198.51.100.0/255.255.0.0: Any host on the 198.51 class B network
  + 198.51.100.0/255.255.255.0: Any host on the 198.51.100 class C network
  + 198.51.100.1: Only the host with this specific IP address

The server performs matching of host values in account names against the client host using the value returned by the system DNS resolver for the client host name or IP address. Except in the case that the account host value is specified using netmask notation, the server performs this comparison as a string match, even for an account host value given as an IP address. This means that you should specify account host values in the same format used by DNS. Here are examples of problems to watch out for:

* Suppose that a host on the local network has a fully qualified name of host1.example.com. If DNS returns name lookups for this host as host1.example.com, use that name in account host values. If DNS returns just host1, use host1 instead.
* If DNS returns the IP address for a given host as 198.51.100.2, that matches an account host value of 198.51.100.2 but not 198.051.100.2. Similarly, it matches an account host pattern like 198.51.100.% but not 198.051.100.%.

To avoid problems like these, it is advisable to check the format in which your DNS returns host names and addresses. Use values in the same format in MySQL account names.

### Specifying Role Names

MySQL role names refer to roles, which are named collections of privileges. For role usage examples, see [Section 6.2.10, “Using Roles”](https://dev.mysql.com/doc/refman/8.0/en/roles.html).

Role names have syntax and semantics similar to account names; see [Section 6.2.4, “Specifying Account Names”](https://dev.mysql.com/doc/refman/8.0/en/account-names.html). As stored in the grant tables, they have the same properties as account names, which are described in [Grant Table Scope Column Properties](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html#grant-tables-scope-column-properties).

Role names differ from account names in these respects:

* The user part of role names cannot be blank. Thus, there is no “anonymous role” analogous to the concept of “anonymous user.”
* As for an account name, omitting the host part of a role name results in a host part of '%'. But unlike '%' in an account name, a host part of '%' in a role name has no wildcard properties. For example, for a name 'me'@'%' used as a role name, the host part ('%') is just a literal value; it has no “any host” matching property.
* Netmask notation in the host part of a role name has no significance.
* An account name is permitted to be [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) in several contexts. A role name is not.

It is possible for a row in the mysql.user system table to serve as both an account and a role. In this case, any special user or host name matching properties do not apply in contexts for which the name is used as a role name. For example, you cannot execute the following statement with the expectation that it sets the current session roles using all roles that have a user part of myrole and any host name:

SET ROLE 'myrole'@'%';

Instead, the statement sets the active role for the session to the role with exactly the name 'myrole'@'%'.

For this reason, role names are often specified using only the user name part and letting the host name part implicitly be '%'. Specifying a role with a non-'%' host part can be useful if you intend to create a name that works both as a role an as a user account that is permitted to connect from the given host.

### Access Control, Stage 1: Connection Verification

When you attempt to connect to a MySQL server, the server accepts or rejects the connection based on these conditions:

* Your identity and whether you can verify it by supplying the proper credentials.
* Whether your account is locked or unlocked.

The server checks credentials first, then account locking state. A failure at either step causes the server to deny access to you completely. Otherwise, the server accepts the connection, and then enters Stage 2 and waits for requests.

The server performs identity and credentials checking using columns in the user table, accepting the connection only if these conditions are satisfied:

* The client host name and user name match the Host and User columns in some user table row. For the rules governing permissible Host and User values, see [Section 6.2.4, “Specifying Account Names”](https://dev.mysql.com/doc/refman/8.0/en/account-names.html).
* The client supplies the credentials specified in the row (for example, a password), as indicated by the authentication\_string column. Credentials are interpreted using the authentication plugin named in the plugin column.
* The row indicates that the account is unlocked. Locking state is recorded in the account\_locked column, which must have a value of 'N'. Account locking can be set or changed with the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) or [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement.

Your identity is based on two pieces of information:

* Your MySQL user name.
* The client host from which you connect.

If the User column value is nonblank, the user name in an incoming connection must match exactly. If the User value is blank, it matches any user name. If the user table row that matches an incoming connection has a blank user name, the user is considered to be an anonymous user with no name, not a user with the name that the client actually specified. This means that a blank user name is used for all further access checking for the duration of the connection (that is, during Stage 2).

The authentication\_string column can be blank. This is not a wildcard and does not mean that any password matches. It means that the user must connect without specifying a password. The authentication method implemented by the plugin that authenticates the client may or may not use the password in the authentication\_string column. In this case, it is possible that an external password is also used to authenticate to the MySQL server.

Nonblank password values stored in the authentication\_string column of the user table are encrypted. MySQL does not store passwords as cleartext for anyone to see. Rather, the password supplied by a user who is attempting to connect is encrypted (using the password hashing method implemented by the account authentication plugin). The encrypted password then is used during the connection process when checking whether the password is correct. This is done without the encrypted password ever traveling over the connection..

From MySQL's point of view, the encrypted password is the real password, so you should never give anyone access to it. In particular, do not give nonadministrative users read access to tables in the *mysql* system database.

The following table shows how various combinations of User and Host values in the user table apply to incoming connections.

| User**Value** | Host**Value** | **Permissible Connections** |
| --- | --- | --- |
| 'fred' | 'h1.example.net' | fred, connecting from h1.example.net |
| '' | 'h1.example.net' | Any user, connecting from h1.example.net |
| 'fred' | '%' | fred, connecting from any host |
| '' | '%' | Any user, connecting from any host |
| 'fred' | '%.example.net' | fred, connecting from any host in the example.net domain |
| 'fred' | 'x.example.%' | fred, connecting from x.example.net, x.example.com, x.example.edu, and so on; this is probably not useful |
| 'fred' | '198.51.100.177' | fred, connecting from the host with IP address 198.51.100.177 |
| 'fred' | '198.51.100.%' | fred, connecting from any host in the 198.51.100 class C subnet |
| 'fred' | '198.51.100.0/255.255.255.0' | Same as previous example |

It is possible for the client host name and user name of an incoming connection to match more than one row in the user table. The preceding set of examples demonstrates this: Several of the entries shown match a connection from h1.example.net by fred.

When multiple matches are possible, the server must determine which of them to use. It resolves this issue as follows:

* Whenever the server reads the user table into memory, it sorts the rows.
* When a client attempts to connect, the server looks through the rows in sorted order.
* The server uses the first row that matches the client host name and user name.

The server uses sorting rules that order rows with the most-specific Host values first:

* Literal host names and IP addresses are the most specific.
* The specificity of a literal IP address is not affected by whether it has a netmask, so 198.51.100.13 and 198.51.100.0/255.255.255.0 are considered equally specific.
* The pattern '%' means “any host” and is least specific.
* The empty string '' also means “any host” but sorts after '%'.

Rows with the same Host value are ordered with the most-specific User values first (a blank User value means “any user” and is least specific). For rows with equally-specific Host and User values, the order is nondeterministic.

To see how this works, suppose that the user table looks like this:

+-----------+----------+-

| Host | User | ...

+-----------+----------+-

| % | root | ...

| % | jeffrey | ...

| localhost | root | ...

| localhost | | ...

+-----------+----------+-

When the server reads the table into memory, it sorts the rows using the rules just described. The result after sorting looks like this:

+-----------+----------+-

| Host | User | ...

+-----------+----------+-

| localhost | root | ...

| localhost | | ...

| % | jeffrey | ...

| % | root | ...

+-----------+----------+-

When a client attempts to connect, the server looks through the sorted rows and uses the first match found. For a connection from localhost by jeffrey, two of the rows from the table match: the one with Host and User values of 'localhost' and '', and the one with values of '%' and 'jeffrey'. The 'localhost' row appears first in sorted order, so that is the one the server uses.

Here is another example. Suppose that the user table looks like this:

+----------------+----------+-

| Host | User | ...

+----------------+----------+-

| % | jeffrey | ...

| h1.example.net | | ...

+----------------+----------+-

The sorted table looks like this:

+----------------+----------+-

| Host | User | ...

+----------------+----------+-

| h1.example.net | | ...

| % | jeffrey | ...

+----------------+----------+-

The first row matches a connection by any user from h1.example.net, whereas the second row matches a connection by jeffrey from any host.

**Note**

It is a common misconception to think that, for a given user name, all rows that explicitly name that user are used first when the server attempts to find a match for the connection. This is not true. The preceding example illustrates this, where a connection from h1.example.net by jeffrey is first matched not by the row containing 'jeffrey' as the User column value, but by the row with no user name. As a result, jeffrey is authenticated as an anonymous user, even though he specified a user name when connecting.

If you are able to connect to the server, but your privileges are not what you expect, you probably are being authenticated as some other account. To find out what account the server used to authenticate you, use the [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) function. (See [Section 12.16, “Information Functions”](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html).) It returns a value in ***user\_name***@***host\_name*** format that indicates the User and Host values from the matching user table row. Suppose that jeffrey connects and issues the following query:

mysql> SELECT CURRENT\_USER();

+----------------+

| CURRENT\_USER() |

+----------------+

| @localhost |

+----------------+

The result shown here indicates that the matching user table row had a blank User column value. In other words, the server is treating jeffrey as an anonymous user.

Another way to diagnose authentication problems is to print out the user table and sort it by hand to see where the first match is being made.

### Access Control, Stage 2: Request Verification

After the server accepts a connection, it enters Stage 2 of access control. For each request that you issue through the connection, the server determines what operation you want to perform, then checks whether your privileges are sufficient. This is where the privilege columns in the grant tables come into play. These privileges can come from any of the user, global\_grants, db, tables\_priv, columns\_priv, or procs\_priv tables. (You may find it helpful to refer to [Section 6.2.3, “Grant Tables”](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html), which lists the columns present in each grant table.)

The user and global\_grants tables grant global privileges. The rows in these tables for a given account indicate the account privileges that apply on a global basis no matter what the default database is. For example, if the user table grants you the [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete) privilege, you can delete rows from any table in any database on the server host. It is wise to grant privileges in the user table only to people who need them, such as database administrators. For other users, leave all privileges in the user table set to 'N' and grant privileges at more specific levels only (for particular databases, tables, columns, or routines). It is also possible to grant database privileges globally but use partial revokes to restrict them from being exercised on specific databases (see [Section 6.2.12, “Privilege Restriction Using Partial Revokes”](https://dev.mysql.com/doc/refman/8.0/en/partial-revokes.html)).

The db table grants database-specific privileges. Values in the scope columns of this table can take the following forms:

* A blank User value matches the anonymous user. A nonblank value matches literally; there are no wildcards in user names.
* The wildcard characters % and \_ can be used in the Host and Db columns. These have the same meaning as for pattern-matching operations performed with the [LIKE](https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html#operator_like) operator. If you want to use either character literally when granting privileges, you must escape it with a backslash. For example, to include the underscore character (\_) as part of a database name, specify it as \\_ in the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement.
* A '%' or blank Host value means “any host.”
* A '%' or blank Db value means “any database.”

The server reads the db table into memory and sorts it at the same time that it reads the user table. The server sorts the db table based on the Host, Db, and User scope columns. As with the user table, sorting puts the most-specific values first and least-specific values last, and when the server looks for matching rows, it uses the first match that it finds.

The tables\_priv, columns\_priv, and procs\_priv tables grant table-specific, column-specific, and routine-specific privileges. Values in the scope columns of these tables can take the following forms:

* The wildcard characters % and \_ can be used in the Host column. These have the same meaning as for pattern-matching operations performed with the [LIKE](https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html#operator_like) operator.
* A '%' or blank Host value means “any host.”
* The Db, Table\_name, Column\_name, and Routine\_name columns cannot contain wildcards or be blank.

The server sorts the tables\_priv, columns\_priv, and procs\_priv tables based on the Host, Db, and User columns. This is similar to db table sorting, but simpler because only the Host column can contain wildcards.

The server uses the sorted tables to verify each request that it receives. For requests that require administrative privileges such as [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown) or [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload), the server checks only the user and global\_privilege tables because those are the only tables that specify administrative privileges. The server grants access if a row for the account in those tables permits the requested operation and denies access otherwise. For example, if you want to execute [**mysqladmin shutdown**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) but your user table row does not grant the [SHUTDOWN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_shutdown) privilege to you, the server denies access without even checking the db table. (The latter table contains no Shutdown\_priv column, so there is no need to check it.)

For database-related requests ([INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), and so on), the server first checks the user's global privileges in the user table row (less any privilege restrictions imposed by partial revokes). If the row permits the requested operation, access is granted. If the global privileges in the user table are insufficient, the server determines the user's database-specific privileges from the db table:

* The server looks in the db table for a match on the Host, Db, and User columns.
* The Host and User columns are matched to the connecting user's host name and MySQL user name.
* The Db column is matched to the database that the user wants to access.
* If there is no row for the Host and User, access is denied.

After determining the database-specific privileges granted by the db table rows, the server adds them to the global privileges granted by the user table. If the result permits the requested operation, access is granted. Otherwise, the server successively checks the user's table and column privileges in the tables\_priv and columns\_priv tables, adds those to the user's privileges, and permits or denies access based on the result. For stored-routine operations, the server uses the procs\_priv table rather than tables\_priv and columns\_priv.

Expressed in boolean terms, the preceding description of how a user's privileges are calculated may be summarized like this:

global privileges

OR database privileges

OR table privileges

OR column privileges

OR routine privileges

It may not be apparent why, if the global privileges are initially found to be insufficient for the requested operation, the server adds those privileges to the database, table, and column privileges later. The reason is that a request might require more than one type of privilege. For example, if you execute an [INSERT INTO ... SELECT](https://dev.mysql.com/doc/refman/8.0/en/insert-select.html) statement, you need both the [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) and the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privileges. Your privileges might be such that the user table row grants one privilege global and the db table row grants the other specifically for the relevant database. In this case, you have the necessary privileges to perform the request, but the server cannot tell that from either your global or database privileges alone. It must make an access-control decision based on the combined privileges.

### Adding Accounts, Assigning Privileges, and Dropping Accounts

To manage MySQL accounts, use the SQL statements intended for that purpose:

* [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html) create and remove accounts.
* [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) assign privileges to and revoke privileges from accounts.
* [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) displays account privilege assignments.

Account-management statements cause the server to make appropriate modifications to the underlying grant tables, which are discussed in [Section 6.2.3, “Grant Tables”](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html).

**Note**

Direct modification of grant tables using statements such as [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) is discouraged and done at your own risk. The server is free to ignore rows that become malformed as a result of such modifications.

For any operation that modifies a grant table, the server checks whether the table has the expected structure and produces an error if not. To update the tables to the expected structure, perform the MySQL upgrade procedure. See [Section 2.11, “Upgrading MySQL”](https://dev.mysql.com/doc/refman/8.0/en/upgrading.html).

Another option for creating accounts is to use the GUI tool MySQL Workbench. Also, several third-party programs offer capabilities for MySQL account administration. phpMyAdmin is one such program.

This section discusses the following topics:

.

#### Creating Accounts and Granting Privileges

The following examples show how to use the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client program to set up new accounts. These examples assume that the MySQL root account has the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege and all privileges that it grants to other accounts.

At the command line, connect to the server as the MySQL root user, supplying the appropriate password at the password prompt:

shell> mysql -u root -p

Enter password: *(*enter root password here*)*

After connecting to the server, you can add new accounts. The following example uses [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html)and [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements to set up four accounts (where you see '***password***', substitute an appropriate password):

CREATE USER 'finley'@'localhost'

IDENTIFIED BY 'password';

GRANT ALL

ON \*.\*

TO 'finley'@'localhost'

WITH GRANT OPTION;

CREATE USER 'finley'@'%.example.com'

IDENTIFIED BY 'password';

GRANT ALL

ON \*.\*

TO 'finley'@'%.example.com'

WITH GRANT OPTION;

CREATE USER 'admin'@'localhost'

IDENTIFIED BY 'password';

GRANT RELOAD,PROCESS

ON \*.\*

TO 'admin'@'localhost';

CREATE USER 'dummy'@'localhost';

The accounts created by those statements have the following properties:

* Two accounts have a user name of finley. Both are superuser accounts with full global privileges to do anything. The 'finley'@'localhost' account can be used only when connecting from the local host. The 'finley'@'%.example.com' account uses the '%' wildcard in the host part, so it can be used to connect from any host in the example.com domain.

The 'finley'@'localhost' account is necessary if there is an anonymous-user account for localhost. Without the 'finley'@'localhost' account, that anonymous-user account takes precedence when finley connects from the local host and finley is treated as an anonymous user. The reason for this is that the anonymous-user account has a more specific Host column value than the 'finley'@'%' account and thus comes earlier in the user table sort order. (For information about user table sorting, see [Section 6.2.6, “Access Control, Stage 1: Connection Verification”](https://dev.mysql.com/doc/refman/8.0/en/connection-access.html).)

* The 'admin'@'localhost' account can be used only by admin to connect from the local host. It is granted the global [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) and [PROCESS](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_process) administrative privileges. These privileges enable the admin user to execute the [**mysqladmin reload**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html), [**mysqladmin refresh**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html), and [**mysqladmin flush-*xxx***](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) commands, as well as [**mysqladmin processlist**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) . No privileges are granted for accessing any databases. You could add such privileges using [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements.
* The 'dummy'@'localhost' account has no password (which is insecure and not recommended). This account can be used only to connect from the local host. No privileges are granted. It is assumed that you grant specific privileges to the account using [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements.

The previous example grants privileges at the global level. The next example creates three accounts and grants them access at lower levels; that is, to specific databases or objects within databases. Each account has a user name of custom, but the host name parts differ:

CREATE USER 'custom'@'localhost'

IDENTIFIED BY 'password';

GRANT ALL

ON bankaccount.\*

TO 'custom'@'localhost';

CREATE USER 'custom'@'host47.example.com'

IDENTIFIED BY 'password';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP

ON expenses.\*

TO 'custom'@'host47.example.com';

CREATE USER 'custom'@'%.example.com'

IDENTIFIED BY 'password';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP

ON customer.addresses

TO 'custom'@'%.example.com';

The three accounts can be used as follows:

* The 'custom'@'localhost' account has all database-level privileges to access the bankaccount database. The account can be used to connect to the server only from the local host.
* The 'custom'@'host47.example.com' account has specific database-level privileges to access the expenses database. The account can be used to connect to the server only from the host host47.example.com.
* The 'custom'@'%.example.com' account has specific table-level privileges to access the addresses table in the customer database, from any host in the example.com domain. The account can be used to connect to the server from all machines in the domain due to use of the % wildcard character in the host part of the account name.

#### Checking Account Privileges and Properties

To see the privileges for an account, use [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html):

mysql> SHOW GRANTS FOR 'admin'@'localhost';

+-----------------------------------------------------+

| Grants for admin@localhost |

+-----------------------------------------------------+

| GRANT RELOAD, PROCESS ON \*.\* TO 'admin'@'localhost' |

+-----------------------------------------------------+

To see nonprivilege properties for an account, use [SHOW CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/show-create-user.html):

mysql> SET print\_identified\_with\_as\_hex = ON;

mysql> SHOW CREATE USER 'admin'@'localhost'\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE USER for admin@localhost: CREATE USER 'admin'@'localhost'

IDENTIFIED WITH 'caching\_sha2\_password'

AS 0x24412430303524301D0E17054E2241362B1419313C3E44326F294133734B30792F436E77764270373039612E32445250786D43594F45354532324B6169794F47457852796E32

REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK

PASSWORD HISTORY DEFAULT

PASSWORD REUSE INTERVAL DEFAULT

PASSWORD REQUIRE CURRENT DEFAULT

Enabling the [print\_identified\_with\_as\_hex](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_print_identified_with_as_hex) system variable (available as of MySQL 8.0.17) causes [SHOW CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/show-create-user.html) to display hash values that contain unprintable characters as hexadecimal strings rather than as regular string literals.

#### Revoking Account Privileges

To revoke account privileges, use the [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statement. Privileges can be revoked at different levels, just as they can be granted at different levels.

Revoke global privileges:

REVOKE ALL

ON \*.\*

FROM 'finley'@'%.example.com';

REVOKE RELOAD

ON \*.\*

FROM 'admin'@'localhost';

Revoke database-level privileges:

REVOKE CREATE,DROP

ON expenses.\*

FROM 'custom'@'host47.example.com';

Revoke table-level privileges:

REVOKE INSERT,UPDATE,DELETE

ON customer.addresses

FROM 'custom'@'%.example.com';

To check the effect of privilege revocation, use [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html):

mysql> SHOW GRANTS FOR 'admin'@'localhost';

+---------------------------------------------+

| Grants for admin@localhost |

+---------------------------------------------+

| GRANT PROCESS ON \*.\* TO 'admin'@'localhost' |

+---------------------------------------------+

#### Dropping Accounts

To remove an account, use the [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html) statement. For example, to drop some of the accounts created previously:

DROP USER 'finley'@'localhost';

DROP USER 'finley'@'%.example.com';

DROP USER 'admin'@'localhost';

DROP USER 'dummy'@'localhost';

### Reserved Accounts

One part of the MySQL installation process is data directory initialization (see [Section 2.10.1, “Initializing the Data Directory”](https://dev.mysql.com/doc/refman/8.0/en/data-directory-initialization.html)). During data directory initialization, MySQL creates user accounts that should be considered reserved:

* 'root'@'localhost: Used for administrative purposes. This account has all privileges, is a system account, and can perform any operation.

Strictly speaking, this account name is not reserved, in the sense that some installations rename the root account to something else to avoid exposing a highly privileged account with a well-known name.

* 'mysql.sys'@'localhost': Used as the DEFINER for [sys](https://dev.mysql.com/doc/refman/8.0/en/sys-schema.html) schema objects. Use of the mysql.sys account avoids problems that occur if a DBA renames or removes the root account. This account is locked so that it cannot be used for client connections.
* 'mysql.session'@'localhost': Used internally by plugins to access the server. This account is locked so that it cannot be used for client connections. The account is a system account.
* 'mysql.infoschema'@'localhost': Used as the DEFINER for [INFORMATION\_SCHEMA](https://dev.mysql.com/doc/refman/8.0/en/information-schema.html) views. Use of the mysql.infoschema account avoids problems that occur if a DBA renames or removes the root account. This account is locked so that it cannot be used for client connections.

### Using Roles

A MySQL role is a named collection of privileges. Like user accounts, roles can have privileges granted to and revoked from them.

A user account can be granted roles, which grants to the account the privileges associated with each role. This enables assignment of sets of privileges to accounts and provides a convenient alternative to granting individual privileges, both for conceptualizing desired privilege assignments and implementing them.

The following list summarizes role-management capabilities provided by MySQL:

* [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html) create and remove roles.
* [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) assign privileges to revoke privileges from user accounts and roles.
* [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) displays privilege and role assignments for user accounts and roles.
* [SET DEFAULT ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-default-role.html) specifies which account roles are active by default.
* [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) changes the active roles within the current session.
* The [CURRENT\_ROLE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-role) function displays the active roles within the current session.
* The [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) and [activate\_all\_roles\_on\_login](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_activate_all_roles_on_login) system variables enable defining mandatory roles and automatic activation of granted roles when users log in to the server.

For descriptions of individual role-manipulation statements (including the privileges required to use them), see [Section 13.7.1, “Account Management Statements”](https://dev.mysql.com/doc/refman/8.0/en/account-management-statements.html). The following discussion provides examples of role usage. Unless otherwise specified, SQL statements shown here should be executed using a MySQL account with sufficient administrative privileges, such as the root account.

#### Creating Roles and Granting Privileges to Them

Consider this scenario:

* An application uses a database named app\_db.
* Associated with the application, there can be accounts for developers who create and maintain the application, and for users who interact with it.
* Developers need full access to the database. Some users need only read access, others need read/write access.

To avoid granting privileges individually to possibly many user accounts, create roles as names for the required privilege sets. This makes it easy to grant the required privileges to user accounts, by granting the appropriate roles.

To create the roles, use the [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) statement:

CREATE ROLE 'app\_developer', 'app\_read', 'app\_write';

Role names are much like user account names and consist of a user part and host part in '***user\_name***'@'***host\_name***' format. The host part, if omitted, defaults to '%'. The user and host parts can be unquoted unless they contain special characters such as - or %. Unlike account names, the user part of role names cannot be blank. For additional information, see [Section 6.2.5, “Specifying Role Names”](https://dev.mysql.com/doc/refman/8.0/en/role-names.html).

To assign privileges to the roles, execute [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements using the same syntax as for assigning privileges to user accounts:

GRANT ALL ON app\_db.\* TO 'app\_developer';

GRANT SELECT ON app\_db.\* TO 'app\_read';

GRANT INSERT, UPDATE, DELETE ON app\_db.\* TO 'app\_write';

Now suppose that initially you require one developer account, two user accounts that need read-only access, and one user account that needs read/write access. Use [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) to create the accounts:

CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1pass';

CREATE USER 'read\_user1'@'localhost' IDENTIFIED BY 'read\_user1pass';

CREATE USER 'read\_user2'@'localhost' IDENTIFIED BY 'read\_user2pass';

CREATE USER 'rw\_user1'@'localhost' IDENTIFIED BY 'rw\_user1pass';

To assign each user account its required privileges, you could use [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements of the same form as just shown, but that requires enumerating individual privileges for each user. Instead, use an alternative [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) syntax that permits granting roles rather than privileges:

GRANT 'app\_developer' TO 'dev1'@'localhost';

GRANT 'app\_read' TO 'read\_user1'@'localhost', 'read\_user2'@'localhost';

GRANT 'app\_read', 'app\_write' TO 'rw\_user1'@'localhost';

The [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement for the rw\_user1 account grants the read and write roles, which combine to provide the required read and write privileges.

The [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) syntax for granting roles to an account differs from the syntax for granting privileges: There is an ON clause to assign privileges, whereas there is no ON clause to assign roles. Because the syntaxes are distinct, you cannot mix assigning privileges and roles in the same statement. (It is permitted to assign both privileges and roles to an account, but you must use separate [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements, each with syntax appropriate to what is to be granted.) As of MySQL 8.0.16, roles cannot be granted to anonymous users.

A role when created is locked, has no password, and is assigned the default authentication plugin. (These role attributes can be changed later with the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement, by users who have the global [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege.)

While locked, a role cannot be used to authenticate to the server. If unlocked, a role can be used to authenticate. This is because roles and users are both authorization identifiers with much in common and little to distinguish them. See also [User and Role Interchangeability](https://dev.mysql.com/doc/refman/8.0/en/roles.html#role-user-interchangeability).

#### Defining Mandatory Roles

It is possible to specify roles as mandatory by naming them in the value of the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable. The server treats a mandatory role as granted to all users, so that it need not be granted explicitly to any account.

To specify mandatory roles at server startup, define [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) in your server my.cnf file:

[mysqld]

mandatory\_roles='role1,role2@localhost,r3@%.example.com'

To set and persist [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) at runtime, use a statement like this:

SET PERSIST mandatory\_roles = 'role1,role2@localhost,r3@%.example.com';

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value, causing it to carry over to subsequent server restarts. To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the GLOBAL keyword rather than PERSIST. See [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html).

Setting [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) requires the [ROLE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_role-admin) privilege, in addition to the [SYSTEM\_VARIABLES\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-variables-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege) normally required to set a global system variable.

Mandatory roles, like explicitly granted roles, do not take effect until activated (see [Activating Roles](https://dev.mysql.com/doc/refman/8.0/en/roles.html#roles-activating)). At login time, role activation occurs for all granted roles if the [activate\_all\_roles\_on\_login](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_activate_all_roles_on_login) system variable is enabled, or for roles that are set as default roles otherwise. At runtime, [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) activates roles.

Roles named in the value of [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) cannot be revoked with [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) or dropped with [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html) or [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html).

To prevent sessions from being made system sessions by default, a role that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege cannot be listed in the value of the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable:

* If [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) is assigned a role at startup that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, the server writes a message to the error log and exits.
* If [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) is assigned a role at runtime that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, an error occurs and the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) value remains unchanged.

If a role named in [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) is not present in the mysql.user system table, the role is not granted to users. When the server attempts role activation for a user, it does not treat the nonexistent role as mandatory and writes a warning to the error log. If the role is created later and thus becomes valid, [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) may be necessary to cause the server to treat it as mandatory.

.

#### Checking Role Privileges

To verify the privileges assigned to an account, use [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html). For example:

mysql> SHOW GRANTS FOR 'dev1'@'localhost';

+-------------------------------------------------+

| Grants for dev1@localhost |

+-------------------------------------------------+

| GRANT USAGE ON \*.\* TO `dev1`@`localhost` |

| GRANT `app\_developer`@`%` TO `dev1`@`localhost` |

+-------------------------------------------------+

However, that shows each granted role without “expanding” it to the privileges the role represents. To show role privileges as well, add a USING clause naming the granted roles for which to display privileges:

mysql> SHOW GRANTS FOR 'dev1'@'localhost' USING 'app\_developer';

+----------------------------------------------------------+

| Grants for dev1@localhost |

+----------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `dev1`@`localhost` |

| GRANT ALL PRIVILEGES ON `app\_db`.\* TO `dev1`@`localhost` |

| GRANT `app\_developer`@`%` TO `dev1`@`localhost` |

+----------------------------------------------------------+

Verify each other type of user similarly:

mysql> SHOW GRANTS FOR 'read\_user1'@'localhost' USING 'app\_read';

+--------------------------------------------------------+

| Grants for read\_user1@localhost |

+--------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `read\_user1`@`localhost` |

| GRANT SELECT ON `app\_db`.\* TO `read\_user1`@`localhost` |

| GRANT `app\_read`@`%` TO `read\_user1`@`localhost` |

+--------------------------------------------------------+

mysql> SHOW GRANTS FOR 'rw\_user1'@'localhost' USING 'app\_read', 'app\_write';

+------------------------------------------------------------------------------+

| Grants for rw\_user1@localhost |

+------------------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `rw\_user1`@`localhost` |

| GRANT SELECT, INSERT, UPDATE, DELETE ON `app\_db`.\* TO `rw\_user1`@`localhost` |

| GRANT `app\_read`@`%`,`app\_write`@`%` TO `rw\_user1`@`localhost` |

+------------------------------------------------------------------------------+

.

#### Activating Roles

Roles granted to a user account can be active or inactive within account sessions. If a granted role is active within a session, its privileges apply; otherwise, they do not. To determine which roles are active within the current session, use the [CURRENT\_ROLE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-role) function.

By default, granting a role to an account or naming it in the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable value does not automatically cause the role to become active within account sessions. For example, because thus far in the preceding discussion no rw\_user1 roles have been activated, if you connect to the server as rw\_user1 and invoke the [CURRENT\_ROLE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-role) function, the result is NONE (no active roles):

mysql> SELECT CURRENT\_ROLE();

+----------------+

| CURRENT\_ROLE() |

+----------------+

| NONE |

+----------------+

To specify which roles should become active each time a user connects to the server and authenticates, use [SET DEFAULT ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-default-role.html). To set the default to all assigned roles for each account created earlier, use this statement:

SET DEFAULT ROLE ALL TO

'dev1'@'localhost',

'read\_user1'@'localhost',

'read\_user2'@'localhost',

'rw\_user1'@'localhost';

Now if you connect as rw\_user1, the initial value of [CURRENT\_ROLE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-role) reflects the new default role assignments:

mysql> SELECT CURRENT\_ROLE();

+--------------------------------+

| CURRENT\_ROLE() |

+--------------------------------+

| `app\_read`@`%`,`app\_write`@`%` |

+--------------------------------+

To cause all explicitly granted and mandatory roles to be automatically activated when users connect to the server, enable the [activate\_all\_roles\_on\_login](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_activate_all_roles_on_login) system variable. By default, automatic role activation is disabled.

Within a session, a user can execute [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) to change the set of active roles. For example, for rw\_user1:

mysql> SET ROLE NONE; SELECT CURRENT\_ROLE();

+----------------+

| CURRENT\_ROLE() |

+----------------+

| NONE |

+----------------+

mysql> SET ROLE ALL EXCEPT 'app\_write'; SELECT CURRENT\_ROLE();

+----------------+

| CURRENT\_ROLE() |

+----------------+

| `app\_read`@`%` |

+----------------+

mysql> SET ROLE DEFAULT; SELECT CURRENT\_ROLE();

+--------------------------------+

| CURRENT\_ROLE() |

+--------------------------------+

| `app\_read`@`%`,`app\_write`@`%` |

+--------------------------------+

The first [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) statement deactivates all roles. The second makes rw\_user1 effectively read only. The third restores the default roles.

The effective user for stored program and view objects is subject to the DEFINER and SQL SECURITY attributes, which determine whether execution occurs in invoker or definer context

Stored program and view objects that execute in invoker context execute with the roles that are active within the current session.

* Stored program and view objects that execute in definer context execute with the default roles of the user named in their DEFINER attribute. If [activate\_all\_roles\_on\_login](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_activate_all_roles_on_login) is enabled, such objects execute with all roles granted to the DEFINER user, including mandatory roles. For stored programs, if execution should occur with roles different from the default, the program body should execute [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) to activate the required roles.

#### Revoking Roles or Role Privileges

Just as roles can be granted to an account, they can be revoked from an account:

REVOKE role FROM user;

Roles named in the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable value cannot be revoked.

[REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) can also be applied to a role to modify the privileges granted to it. This affects not only the role itself, but any account granted that role. Suppose that you want to temporarily make all application users read only. To do this, use [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) to revoke the modification privileges from the app\_write role:

REVOKE INSERT, UPDATE, DELETE ON app\_db.\* FROM 'app\_write';

As it happens, that leaves the role with no privileges at all, as can be seen using [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) (which demonstrates that this statement can be used with roles, not just users):

mysql> SHOW GRANTS FOR 'app\_write';

+---------------------------------------+

| Grants for app\_write@% |

+---------------------------------------+

| GRANT USAGE ON \*.\* TO `app\_write`@`%` |

+---------------------------------------+

Because revoking privileges from a role affects the privileges for any user who is assigned the modified role, rw\_user1 now has no table modification privileges ([INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update), and [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete) are no longer present):

mysql> SHOW GRANTS FOR 'rw\_user1'@'localhost'

USING 'app\_read', 'app\_write';

+----------------------------------------------------------------+

| Grants for rw\_user1@localhost |

+----------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `rw\_user1`@`localhost` |

| GRANT SELECT ON `app\_db`.\* TO `rw\_user1`@`localhost` |

| GRANT `app\_read`@`%`,`app\_write`@`%` TO `rw\_user1`@`localhost` |

+----------------------------------------------------------------+

In effect, the rw\_user1 read/write user has become a read-only user. This also occurs for any other accounts that are granted the app\_write role, illustrating how use of roles makes it unnecessary to modify privileges for individual accounts.

To restore modification privileges to the role, simply re-grant them:

GRANT INSERT, UPDATE, DELETE ON app\_db.\* TO 'app\_write';

Now rw\_user1 again has modification privileges, as do any other accounts granted the app\_write role.

#### Dropping Roles

To drop roles, use [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html):

DROP ROLE 'app\_read', 'app\_write';

Dropping a role revokes it from every account to which it was granted.

Roles named in the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable value cannot be dropped.

#### User and Role Interchangeability

As has been hinted at earlier for [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html), which displays grants for user accounts or roles, accounts and roles can be used interchangeably.

One difference between roles and users is that [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) creates an authorization identifier that is locked by default, whereas [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) creates an authorization identifier that is unlocked by default. However, distinction is not immutable because a user with appropriate privileges can lock or unlock roles or users after they have been created.

If a database administrator has a preference that a specific authorization identifier must be a role, a name scheme can be used to communicate this intention. For example, you could use a r\_ prefix for all authorization identifiers that you intend to be roles and nothing else.

Another difference between roles and users lies in the privileges available for administering them:

* The [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-role) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop-role) privileges enable only use of the [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html) statements, respectively.
* The [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege enables use of the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/create-role.html), [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/drop-role.html), [DROP USER](https://dev.mysql.com/doc/refman/8.0/en/drop-user.html), [RENAME USER](https://dev.mysql.com/doc/refman/8.0/en/rename-user.html), and [REVOKE ALL PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements.

Thus, the [CREATE ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-role) and [DROP ROLE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_drop-role) privileges are not as powerful as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) and may be granted to users who should only be permitted to create and drop roles, and not perform more general account manipulation.

With regard to privileges and interchangeability of users and roles, you can treat a user account like a role and grant that account to another user or a role. The effect is to grant the account's privileges and roles to the other user or role.

This set of statements demonstrates that you can grant a user to a user, a role to a user, a user to a role, or a role to a role:

CREATE USER 'u1';

CREATE ROLE 'r1';

GRANT SELECT ON db1.\* TO 'u1';

GRANT SELECT ON db2.\* TO 'r1';

CREATE USER 'u2';

CREATE ROLE 'r2';

GRANT 'u1', 'r1' TO 'u2';

GRANT 'u1', 'r1' TO 'r2';

The result in each case is to grant to the grantee object the privileges associated with the granted object. After executing those statements, each of u2 and r2 have been granted privileges from a user (u1) and a role (r1):

mysql> SHOW GRANTS FOR 'u2' USING 'u1', 'r1';

+-------------------------------------+

| Grants for u2@% |

+-------------------------------------+

| GRANT USAGE ON \*.\* TO `u2`@`%` |

| GRANT SELECT ON `db1`.\* TO `u2`@`%` |

| GRANT SELECT ON `db2`.\* TO `u2`@`%` |

| GRANT `u1`@`%`,`r1`@`%` TO `u2`@`%` |

+-------------------------------------+

mysql> SHOW GRANTS FOR 'r2' USING 'u1', 'r1';

+-------------------------------------+

| Grants for r2@% |

+-------------------------------------+

| GRANT USAGE ON \*.\* TO `r2`@`%` |

| GRANT SELECT ON `db1`.\* TO `r2`@`%` |

| GRANT SELECT ON `db2`.\* TO `r2`@`%` |

| GRANT `u1`@`%`,`r1`@`%` TO `r2`@`%` |

+-------------------------------------+

The preceding example is illustrative only, but interchangeability of user accounts and roles has practical application, such as in the following situation: Suppose that a legacy application development project began before the advent of roles in MySQL, so all user accounts associated with the project are granted privileges directly (rather than granted privileges by virtue of being granted roles). One of these accounts is a developer account that was originally granted privileges as follows:

CREATE USER 'old\_app\_dev'@'localhost' IDENTIFIED BY 'old\_app\_devpass';

GRANT ALL ON old\_app.\* TO 'old\_app\_dev'@'localhost';

If this developer leaves the project, it becomes necessary to assign the privileges to another user, or perhaps multiple users if development activies have expanded. Here are some ways to deal with the issue:

* Without using roles: Change the account password so the original developer cannot use it, and have a new developer use the account instead:

ALTER USER 'old\_app\_dev'@'localhost' IDENTIFIED BY 'new\_password';

* Using roles: Lock the account to prevent anyone from using it to connect to the server:

ALTER USER 'old\_app\_dev'@'localhost' ACCOUNT LOCK;

Then treat the account as a role. For each developer new to the project, create a new account and grant to it the original developer account:

CREATE USER 'new\_app\_dev1'@'localhost' IDENTIFIED BY 'new\_password';

GRANT 'old\_app\_dev'@'localhost' TO 'new\_app\_dev1'@'localhost';

The effect is to assign the original developer account privileges to the new account.

### Account Categories

As of MySQL 8.0.16, MySQL incorporates the concept of user account categories, based on the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege.

#### System and Regular Accounts

MySQL incorporates the concept of user account categories, with system and regular users distinguished according to whether they have the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege:

* A user with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a system user.
* A user without the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a regular user.

The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege has an effect on the accounts to which a given user can apply its other privileges, as well as whether the user is protected from other accounts:

* A system user can modify both system and regular accounts. That is, a user who has the appropriate privileges to perform a given operation on regular accounts is enabled by possession of [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) to also perform the operation on system accounts. A system account can be modified only by system users with appropriate privileges, not by regular users.
* A regular user with appropriate privileges can modify regular accounts, but not system accounts. A regular account can be modified by both system and regular users with appropriate privileges.

If a user has the appropriate privileges to perform a given operation on regular accounts, [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) enables the user to also perform the operation on system accounts. [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) does not imply any other privilege, so the ability to perform a given account operation remains predicated on possession of any other required privileges. For example, if a user can grant the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) and [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) privileges to regular accounts, then with [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) the user can also grant [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) and [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) to system accounts.

The distinction between system and regular accounts enables better control over certain account administration issues by protecting accounts that have the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege from accounts that do not have the privilege. For example, the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege enables not only creation of new accounts, but modification and removal of existing accounts. Without the system user concept, a user who has the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege can modify or drop any existing account, including the root account. The concept of system user enables restricting modifications to the root account (itself a system account) so they can be made only by system users. Regular users with the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege can still modify or drop existing accounts, but only regular accounts.

#### Operations Affected by the SYSTEM\_USER Privilege

The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege affects these operations:

* Account manipulation.

Account manipulation includes creating and dropping accounts, granting and revoking privileges, changing account authentication characteristics such as credentials or authentication plugin, and changing other account characteristics such as password expiration policy.

The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is required to manipulate system accounts using account-management statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html). To prevent an account from modifying system accounts this way, make it a regular account by not granting it the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege. (However, to fully protect system accounts against regular accounts, you must also withhold modification privileges for the mysql system schema from regular accounts.

* Killing current sessions and statements executing within them.

To kill a session or statement that is executing with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, your own session must have the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, in addition to any other required privilege ([CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege).

Prior to MySQL 8.0.16, [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege) is sufficient to kill any session or statement.

* Setting the DEFINER attribute for stored objects.

To set the DEFINER attribute for a stored object to an account that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, you must have the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, in addition to any other required privilege ([SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_set-user-id) or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege).

Prior to MySQL 8.0.16, the [SET\_USER\_ID](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_set-user-id) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege) is sufficient to specify any DEFINER value for stored objects.

* Specifying mandatory roles.

A role that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege cannot be listed in the value of the [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles) system variable.

Prior to MySQL 8.0.16, any role can be listed in [mandatory\_roles](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles).

#### System and Regular Sessions

Sessions executing within the server are distinguished as system or regular sessions, similar to the distinction between system and regular users:

* A session that possesses the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a system session.
* A session that does not possess the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is a regular session.

A regular session is able to perform only operations permitted to regular users. A system session is additionally able to perform operations permitted only to system users.

The privileges possessed by a session are those granted directly to its underlying account, plus those granted to all roles currently active within the session. Thus, a session may be a system session because its account has been granted the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege directly, or because the session has activated a role that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege. Roles granted to an account that are not active within the session do not affect session privileges.

Because activating and deactivating roles can change the privileges possessed by sessions, a session may change from a regular session to a system session or vice versa. If a session activates or deactivates a role that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, the appropriate change between regular and system session takes place immediately, for that session only:

* If a regular session activates a role with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, the session becomes a system session.
* If a system session deactivates a role with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, the session becomes a regular session, unless some other role with the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege remains active.

These operations have no effect on existing sessions:

* If the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is granted to or revoked from an account, existing sessions for the account do not change between regular and system sessions. The grant or revoke operation affects only sessions for subsequent connections by the account.
* Statements executed by a stored object invoked within a session execute with the system or regular status of the parent session, even if the object DEFINER attribute names a system account.

Because role activation affects only sessions and not accounts, granting a role that has the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege to a regular account does not protect that account against regular users. The role protects only sessions for the account in which the role has been activated, and protects the session only against being killed by regular sessions.

#### Protecting System Accounts Against Manipulation by Regular Accounts

Account manipulation includes creating and dropping accounts, granting and revoking privileges, changing account authentication characteristics such as credentials or authentication plugin, and changing other account characteristics such as password expiration policy.

Account manipulation can be done two ways:

* By using account-management statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html). This is the preferred method.
* By direct grant-table modification using statements such as [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html) and [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html). This method is discouraged but possible for users with the appropriate privileges on the mysql system schema that contains the grant tables.

To fully protect system accounts against modification by a given account, make it a regular account and do not grant it modification privileges for the mysql schema:

* The [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege is required to manipulate system accounts using account-management statements. To prevent an account from modifying system accounts this way, make it a regular account by not granting [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) to it. This includes not granting [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) to any roles granted to the account.
* Privileges for the mysql schema enable manipulation of system accounts through direct modification of the grant tables, even if the modifying account is a regular account. To restrict unauthorized direct modification of system accounts by a regular account, do not grant modification privileges for the mysql schema to the account (or any roles granted to the account). If a regular account must have global privileges that apply to all schemas, mysql schema modifications can be prevented using privilege restrictions imposed using partial revokes.

**Note**

Unlike withholding the [SYSTEM\_USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_system-user) privilege, which prevents an account from modifying system accounts but not regular accounts, withholding mysql schema privileges prevents an account from modifying system accounts as well as regular accounts. This should not be an issue because, as mentioned, direct grant-table modification is discouraged.

Suppose that you want to create a user u1 who has all privileges on all schemas, except that u1 should be a regular user without the ability to modify system accounts. Assuming that the [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) system variable is enabled, configure u1 as follows:

CREATE USER u1 IDENTIFIED BY 'password';

GRANT ALL ON \*.\* TO u1 WITH GRANT OPTION;

-- GRANT ALL includes SYSTEM\_USER, so at this point

-- u1 can manipulate system or regular accounts

REVOKE SYSTEM\_USER ON \*.\* FROM u1;

-- Revoking SYSTEM\_USER makes u1 a regular user;

-- now u1 can use account-management statements

-- to manipulate only regular accounts

REVOKE ALL ON mysql.\* FROM u1;

-- This partial revoke prevents u1 from directly

-- modifying grant tables to manipulate accounts

To prevent all mysql system schema access by an account, revoke all its privileges on the mysql schema, as just shown. It is also possible to permit partial mysql schema access, such as read-only access. The following example creates an account that has SELECT, INSERT, UPDATE, and DELETE privileges globally for all schemas, but only SELECT for the mysql schema:

CREATE USER u2 IDENTIFIED BY 'password';

GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO u2;

REVOKE INSERT, UPDATE, DELETE ON mysql.\* FROM u2;

Another possibility is to revoke all mysql schema privileges but grant access to specific mysql tables or columns. This can be done even with a partial revoke on mysql. The following statements enable read-only access to u1 within the mysql schema, but only for the db table and the Host and User columns of the user table:

CREATE USER u3 IDENTIFIED BY 'password';

GRANT ALL ON \*.\* TO u3;

REVOKE ALL ON mysql.\* FROM u3;

GRANT SELECT ON mysql.db TO u3;

GRANT SELECT(Host,User) ON mysql.user TO u3;

### Privilege Restriction Using Partial Revokes

Prior to MySQL 8.0.16, it is not possible to grant privileges that apply globally except for certain schemas. As of MySQL 8.0.16, that is possible if the [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) system variable is enabled. Specifically, for users who have privileges at the global level, [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) enables privileges for specific schemas to be revoked while leaving the privileges in place for other schemas. Privilege restrictions thus imposed may be useful for administration of accounts that have global privileges but should not be permitted to access certain schemas. For example, it is possible to permit an account to modify any table except those in the mysql system schema.

**Note**

For brevity, [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statements shown here do not include passwords. For production use, always assign account passwords.

#### Using Partial Revokes

The [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) system variable controls whether privilege restrictions can be placed on accounts. By default, [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) is disabled and attempts to partially revoke global privileges produce an error:

mysql> CREATE USER u1;

mysql> GRANT SELECT, INSERT ON \*.\* TO u1;

mysql> REVOKE INSERT ON world.\* FROM u1;

ERROR 1141 (42000): There is no such grant defined for user 'u1' on host '%'

To permit the [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) operation, enable [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes):

SET PERSIST partial\_revokes = ON;

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value, causing it to carry over to subsequent server restarts. To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the GLOBAL keyword rather than PERSIST. See [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html).

With [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) enabled, the partial revoke succeeds:

mysql> REVOKE INSERT ON world.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+------------------------------------------+

| Grants for u1@% |

+------------------------------------------+

| GRANT SELECT, INSERT ON \*.\* TO `u1`@`%` |

| REVOKE INSERT ON `world`.\* FROM `u1`@`%` |

+------------------------------------------+

[SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) lists partial revokes as [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements in its output. The result indicates that u1 has global [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) and [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) privileges, except that [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) cannot be exercised for tables in the world schema. That is, access by u1 to world tables is read only.

The server records privilege restrictions implemented through partial revokes in the mysql.user system table. If an account has partial revokes, its User\_attributes column value has a Restrictions attribute:

mysql> SELECT User, Host, User\_attributes->>'$.Restrictions'

FROM mysql.user WHERE User\_attributes->>'$.Restrictions' <> '';

+------+------+------------------------------------------------------+

| User | Host | User\_attributes->>'$.Restrictions' |

+------+------+------------------------------------------------------+

| u1 | % | [{"Database": "world", "Privileges": ["INSERT"]}] |

+------+------+------------------------------------------------------+

**Note**

Although partial revokes can be imposed for any schema, privilege restrictions on the mysql system schema in particular are useful as part of a strategy for preventing regular accounts from modifying system accounts. See [Protecting System Accounts Against Manipulation by Regular Accounts](https://dev.mysql.com/doc/refman/8.0/en/account-categories.html#protecting-system-accounts).

Partial revoke operations are subject to these conditions:

* Partial revokes must name the schema literally. Schema names that contain the % or \_ SQL wildcard characters (for example, myschema%) are not permitted.
* It is possible to use partial revokes to place restrictions on nonexistent schemas, but only if the revoked privilege is granted globally. If a privilege is not granted globally, revoking it for a nonexistent schema produces an error.
* Partial revokes apply at the schema level only. You cannot use partial revokes for privileges that apply only globally (such as [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) or [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-admin)), or for table, column, or routine privileges.

As mentioned previously, partial revokes of schema-level privileges appear in [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) output as [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements. This differs from how [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) represents “plain” schema-level privileges:

* When granted, schema-level privileges are represented by their own [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements in the output:
* mysql> CREATE USER u1;
* mysql> GRANT UPDATE ON mysql.\* TO u1;
* mysql> GRANT DELETE ON world.\* TO u1;
* mysql> SHOW GRANTS FOR u1;
* +---------------------------------------+
* | Grants for u1@% |
* +---------------------------------------+
* | GRANT USAGE ON \*.\* TO `u1`@`%` |
* | GRANT UPDATE ON `mysql`.\* TO `u1`@`%` |
* | GRANT DELETE ON `world`.\* TO `u1`@`%` |

+---------------------------------------+

* When revoked, schema-level privileges simply disappear from the output. They do not appear as [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements:
* mysql> REVOKE UPDATE ON mysql.\* FROM u1;
* mysql> REVOKE DELETE ON world.\* FROM u1;
* mysql> SHOW GRANTS FOR u1;
* +--------------------------------+
* | Grants for u1@% |
* +--------------------------------+
* | GRANT USAGE ON \*.\* TO `u1`@`%` |

+--------------------------------+

When a user grants a privilege, any restriction the grantor has on the privilege is inherited by the grantee, unless the grantee already has the privilege without the restriction. Consider the following two users, one of whom has the global [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege:

CREATE USER u1, u2;

GRANT SELECT ON \*.\* TO u2;

Suppose that an administrative user admin has a global but partially revoked [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege:

mysql> CREATE USER admin;

mysql> GRANT SELECT ON \*.\* TO admin WITH GRANT OPTION;

mysql> REVOKE SELECT ON mysql.\* FROM admin;

mysql> SHOW GRANTS FOR admin;

+------------------------------------------------------+

| Grants for admin@% |

+------------------------------------------------------+

| GRANT SELECT ON \*.\* TO `admin`@`%` WITH GRANT OPTION |

| REVOKE SELECT ON `mysql`.\* FROM `admin`@`%` |

+------------------------------------------------------+

If admin grants [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) globally to u1 and u2, the result differs for each user:

* If admin grants [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) globally to u1, who has no [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege to begin with, u1 inherits the admin privilege restriction:
* mysql> GRANT SELECT ON \*.\* TO u1;
* mysql> SHOW GRANTS FOR u1;
* +------------------------------------------+
* | Grants for u1@% |
* +------------------------------------------+
* | GRANT SELECT ON \*.\* TO `u1`@`%` |
* | REVOKE SELECT ON `mysql`.\* FROM `u1`@`%` |

+------------------------------------------+

* On the other hand, u2 already holds a global [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) privilege without restriction. [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) can only add to a grantee's existing privileges, not reduce them, so if admin grants [SELECT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_select) globally to u2, u2 does not inherit the admin restriction:
* mysql> GRANT SELECT ON \*.\* TO u2;
* mysql> SHOW GRANTS FOR u2;
* +---------------------------------+
* | Grants for u2@% |
* +---------------------------------+
* | GRANT SELECT ON \*.\* TO `u2`@`%` |

+---------------------------------+

If a [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement includes an AS ***user*** clause, the privilege restrictions applied are those on the user/role combination specified by the clause, rather than those on the user who executes the statement. For information about the AS clause, see [Section 13.7.1.6, “GRANT Statement”](https://dev.mysql.com/doc/refman/8.0/en/grant.html).

Restrictions on new privileges granted to an account are added to any existing restrictions for that account:

mysql> CREATE USER u1;

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO u1;

mysql> REVOKE INSERT ON mysql.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+---------------------------------------------------------+

| Grants for u1@% |

+---------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE INSERT ON `mysql`.\* FROM `u1`@`%` |

+---------------------------------------------------------+

mysql> REVOKE DELETE, UPDATE ON db2.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+---------------------------------------------------------+

| Grants for u1@% |

+---------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE UPDATE, DELETE ON `db2`.\* FROM `u1`@`%` |

| REVOKE INSERT ON `mysql`.\* FROM `u1`@`%` |

+---------------------------------------------------------+

Aggregation of privilege restrictions applies both when privileges are partially revoked explicitly (as just shown) and when restrictions are inherited implicitly from the user who executes the statement or the user mentioned in an AS ***user*** clause.

If an account has a privilege restriction on a schema:

* The account cannot grant to other accounts a privilege on the restricted schema or any object within it.
* Another account that does not have the restriction can grant privileges to the restricted account for the restricted schema or objects within it. Suppose that an unrestricted user executes these statements:
* CREATE USER u1;
* GRANT SELECT, INSERT, UPDATE ON \*.\* TO u1;
* REVOKE SELECT, INSERT, UPDATE ON mysql.\* FROM u1;
* GRANT SELECT ON mysql.user TO u1; -- grant table privilege

GRANT SELECT(Host,User) ON mysql.db TO u1; -- grant column privileges

The resulting account has these privileges, with the ability to perform limited operations within the restricted schema:

mysql> SHOW GRANTS FOR u1;

+-----------------------------------------------------------+

| Grants for u1@% |

+-----------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE ON \*.\* TO `u1`@`%` |

| REVOKE SELECT, INSERT, UPDATE ON `mysql`.\* FROM `u1`@`%` |

| GRANT SELECT (`Host`, `User`) ON `mysql`.`db` TO `u1`@`%` |

| GRANT SELECT ON `mysql`.`user` TO `u1`@`%` |

+-----------------------------------------------------------+

If an account has a restriction on a global privilege, the restriction is removed by any of these actions:

* Granting the privilege globally to the account by an account that has no restriction on the privilege.
* Granting the privilege at the schema level.
* Revoking the privilege globally.

Consider a user u1 who holds several privileges globally, but with restrictions on [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) and [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete):

mysql> CREATE USER u1;

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO u1;

mysql> REVOKE INSERT, UPDATE, DELETE ON mysql.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+----------------------------------------------------------+

| Grants for u1@% |

+----------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE INSERT, UPDATE, DELETE ON `mysql`.\* FROM `u1`@`%` |

+----------------------------------------------------------+

Granting a privilege globally to u1 from an account with no restriction removes the privilege restriction. For example, to remove the [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) restriction:

mysql> GRANT INSERT ON \*.\* TO u1;

mysql> SHOW GRANTS FOR u1;

+---------------------------------------------------------+

| Grants for u1@% |

+---------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE UPDATE, DELETE ON `mysql`.\* FROM `u1`@`%` |

+---------------------------------------------------------+

Granting a privilege at the schema level to u1 removes the privilege restriction. For example, to remove the [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) restriction:

mysql> GRANT UPDATE ON mysql.\* TO u1;

mysql> SHOW GRANTS FOR u1;

+---------------------------------------------------------+

| Grants for u1@% |

+---------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE DELETE ON `mysql`.\* FROM `u1`@`%` |

+---------------------------------------------------------+

Revoking a global privilege removes the privilege, including any restrictions on it. For example, to remove the [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete) restriction (at the cost of removing all [DELETE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_delete) access):

mysql> REVOKE DELETE ON \*.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+-------------------------------------------------+

| Grants for u1@% |

+-------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE ON \*.\* TO `u1`@`%` |

+-------------------------------------------------+

If an account has a privilege at both the global and schema levels, you must revoke it at the schema level twice to effect a partial revoke. Suppose that u1 has these privileges, where [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) is held both globally and on the world schema:

mysql> CREATE USER u1;

mysql> GRANT SELECT, INSERT ON \*.\* TO u1;

mysql> GRANT INSERT ON world.\* TO u1;

mysql> SHOW GRANTS FOR u1;

+-----------------------------------------+

| Grants for u1@% |

+-----------------------------------------+

| GRANT SELECT, INSERT ON \*.\* TO `u1`@`%` |

| GRANT INSERT ON `world`.\* TO `u1`@`%` |

+-----------------------------------------+

Revoking [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) on world revokes the schema-level privilege ([SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) no longer displays the schema-level [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement):

mysql> REVOKE INSERT ON world.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+-----------------------------------------+

| Grants for u1@% |

+-----------------------------------------+

| GRANT SELECT, INSERT ON \*.\* TO `u1`@`%` |

+-----------------------------------------+

Revoking [INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) on world again performs a partial revoke of the global privilege ([SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) now includes a schema-level [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statement):

mysql> REVOKE INSERT ON world.\* FROM u1;

mysql> SHOW GRANTS FOR u1;

+------------------------------------------+

| Grants for u1@% |

+------------------------------------------+

| GRANT SELECT, INSERT ON \*.\* TO `u1`@`%` |

| REVOKE INSERT ON `world`.\* FROM `u1`@`%` |

+------------------------------------------+

#### Partial Revokes Versus Explicit Schema Grants

To provide access to accounts for some schemas but not others, partial revokes provide an alternative to the approach of explicitly granting schema-level access without granting global privileges. The two approaches have different advantages and disadvantages.

Granting schema-level privileges and not global privileges:

* Adding a new schema: The schema is inaccessible to existing accounts by default. For any account to which the schema should be accessible, the DBA must grant schema-level access.
* Adding a new account: The DBA must grant schema-level access for each schema to which the account should have access.

Granting global privileges in conjunction with partial revokes:

* Adding a new schema: The schema is accessible to existing accounts that have global privileges. For any such account to which the schema should be inaccessible, the DBA must add a partial revoke.
* Adding a new account: The DBA must grant the global privileges, plus a partial revoke on each restricted schema.

The approach that uses explicit schema-level grant is more convenient for accounts for which access is limited to a few schemas. The approach that uses partial revokes is more convenient for accounts with broad access to all schemas except a few.

#### Disabling Partial Revokes

Once enabled, [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) cannot be disabled if any account has privilege restrictions. If any such account exists, disabling [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) fails:

* For attempts to disable [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) at startup, the server logs an error message and enables [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes).
* For attempts to disable [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) at runtime, an error occurs and the [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) value remains unchanged.

To disable [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) when restrictions exist, the restrictions first must be removed:

1. Determine which accounts have partial revokes:
2. SELECT User, Host, User\_attributes->>'$.Restrictions'

FROM mysql.user WHERE User\_attributes->>'$.Restrictions' <> '';

1. For each such account, remove its privilege restrictions. Suppose that the previous step shows account u1 to have these restrictions:

[{"Database": "world", "Privileges": ["INSERT", "DELETE"]

Restriction removal can be done various ways:

* + Grant the privileges globally, without restrictions:

GRANT INSERT, DELETE ON \*.\* TO u1;

* + Grant the privileges at the schema level:

GRANT INSERT, DELETE ON world.\* TO u1;

* + Revoke the privileges globally (assuming that they are no longer needed):

REVOKE INSERT, DELETE ON \*.\* FROM u1;

* + Remove the account itself (assuming that it is no longer needed):

DROP USER u1;

After all privilege restrictions are removed, it is possible to disable partial revokes:

SET PERSIST partial\_revokes = OFF;

#### Partial Revokes and Replication

In replication scenarios, if [partial\_revokes](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_partial_revokes) is enabled on any host, it must be enabled on all hosts. Otherwise, [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) statements to partially revoke a global privilege do not have the same effect for all hosts on which replication occurs, potentially resulting in replication inconsistencies or errors.

### When Privilege Changes Take Effect

If the [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) server is started without the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option, it reads all grant table contents into memory during its startup sequence. The in-memory tables become effective for access control at that point.

If you modify the grant tables indirectly using an account-management statement, the server notices these changes and loads the grant tables into memory again immediately. Account-management statements are described in [Section 13.7.1, “Account Management Statements”](https://dev.mysql.com/doc/refman/8.0/en/account-management-statements.html). Examples include [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html), [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html), [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html), and [RENAME USER](https://dev.mysql.com/doc/refman/8.0/en/rename-user.html).

If you modify the grant tables directly using statements such as [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) (which is not recommended), the changes have no effect on privilege checking until you either tell the server to reload the tables or restart it. Thus, if you change the grant tables directly but forget to reload them, the changes have no effect until you restart the server. This may leave you wondering why your changes seem to make no difference!

To tell the server to reload the grant tables, perform a flush-privileges operation. This can be done by issuing a [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement or by executing a [**mysqladmin flush-privileges**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) or [**mysqladmin reload**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command.

A grant table reload affects privileges for each existing client session as follows:

* Table and column privilege changes take effect with the client's next request.
* Database privilege changes take effect the next time the client executes a USE ***db\_name*** statement.

**Note**

Client applications may cache the database name; thus, this effect may not be visible to them without actually changing to a different database.

* Global privileges and passwords are unaffected for a connected client. These changes take effect only in sessions for subsequent connections.

Changes to the set of active roles within a session take effect immediately, for that session only. The [SET ROLE](https://dev.mysql.com/doc/refman/8.0/en/set-role.html) statement performs session role activation and deactivation (see [Section 13.7.1.11, “SET ROLE Statement”](https://dev.mysql.com/doc/refman/8.0/en/set-role.html)).

If the server is started with the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option, it does not read the grant tables or implement any access control. Any user can connect and perform any operation, which is insecure. To cause a server thus started to read the tables and enable access checking, flush the privileges.

### Assigning Account Passwords

Required credentials for clients that connect to the MySQL server can include a password. This section describes how to assign passwords for MySQL accounts.

MySQL stores credentials in the user table in the mysql system database. Operations that assign or modify passwords are permitted only to users with the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege, or, alternatively, privileges for the mysql database ([INSERT](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_insert) privilege to create new accounts, [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) privilege to modify existing accounts). If the [read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_read_only) system variable is enabled, use of account-modification statements such as [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) or [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) additionally requires the [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege).

The discussion here summarizes syntax only for the most common password-assignment statements. For complete details on other possibilities, see [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [Section 13.7.1.1, “ALTER USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), and [Section 13.7.1.10, “SET PASSWORD Statement”](https://dev.mysql.com/doc/refman/8.0/en/set-password.html).

MySQL uses plugins to perform client authentication; see [Section 6.2.17, “Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html). In password-assigning statements, the authentication plugin associated with an account performs any hashing required of a cleartext password specified. This enables MySQL to obfuscate passwords prior to storing them in the mysql.user system table. For the statements described here, MySQL automatically hashes the password specified. There are also syntax for [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) that permits hashed values to be specified literally. For details, see the descriptions of those statements.

To assign a password when you create a new account, use [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and include an IDENTIFIED BY clause:

CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'password';

[CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) also supports syntax for specifying the account authentication plugin. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html).

To assign or change a password for an existing account, use the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement with an IDENTIFIED BY clause:

ALTER USER 'jeffrey'@'localhost' IDENTIFIED BY 'password';

If you are not connected as an anonymous user, you can change your own password without naming your own account literally:

ALTER USER USER() IDENTIFIED BY 'password';

To change an account password from the command line, use the [**mysqladmin**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command:

mysqladmin -u user\_name -h host\_name password "password"

The account for which this command sets the password is the one with a row in the mysql.user system table that matches ***user\_name*** in the User column and the client host from which you connect in the Host column.

**Warning**

Setting a password using [**mysqladmin**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) should be considered insecure. On some systems, your password becomes visible to system status programs such as **ps** that may be invoked by other users to display command lines. MySQL clients typically overwrite the command-line password argument with zeros during their initialization sequence. However, there is still a brief interval during which the value is visible. Also, on some systems this overwriting strategy is ineffective and the password remains visible to **ps**. (SystemV Unix systems and perhaps others are subject to this problem.)

If you are using MySQL Replication, be aware that, currently, a password used by a replica as part of a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement is effectively limited to 32 characters in length; if the password is longer, any excess characters are truncated. This is not due to any limit imposed by MySQL Server generally, but rather is an issue specific to MySQL Replication. (For more information, see Bug #43439.)

### Password Management

MySQL supports these password-management capabilities:

* Password expiration, to require passwords to be changed periodically.
* Password reuse restrictions, to prevent old passwords from being chosen again.
* Password verification, to require that password changes also specify the current password to be replaced.
* Dual passwords, to enable clients to connect using either a primary or secondary password.
* Password strength assessment, to require strong passwords.
* Random password generation, as an alternative to requiring explicit administrator-specified literal passwords.
* Password failure tracking, to enable temporary account locking after too many consecutive incorrect-password login failures.

**Important**

MySQL implements password-management capabilities using tables in the mysql system database. If you upgrade MySQL from an earlier version, your system tables might not be up to date. In that case, the server writes messages similar to these to the error log during the startup process (the exact numbers may vary):

[ERROR] Column count of mysql.user is wrong. Expected

49, found 47. The table is probably corrupted

[Warning] ACL table mysql.password\_history missing.

Some operations may fail.

To correct the issue, perform the MySQL upgrade procedure.

#### Internal Versus External Credentials Storage

Some authentication plugins store account credentials internally to MySQL, in the mysql.user system table:

* mysql\_native\_password
* caching\_sha2\_password
* sha256\_password

Most discussion in this section applies to such authentication plugins because most password-management capabilities described here are based on internal credentials storage handled by MySQL itself. Other authentication plugins store account credentials externally to MySQL. For accounts that use plugins that perform authentication against an external credentials system, password management must be handled externally against that system as well.

The exception is that the options for failed-login tracking and temporary account locking apply to all accounts, not just accounts that use internal credentials storage, because MySQL is able to assess the status of login attempts for any account no matter whether it uses internal or external credentials storage.

For information about individual authentication plugins, see [Section 6.4.1, “Authentication Plugins”](https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html).

#### Password Expiration Policy

MySQL enables database administrators to expire account passwords manually, and to establish a policy for automatic password expiration. Expiration policy can be established globally, and individual accounts can be set to either defer to the global policy or override the global policy with specific per-account behavior.

To expire an account password manually, use the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement:

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE;

This operation marks the password expired in the corresponding row in the mysql.user system table.

Password expiration according to policy is automatic and is based on password age, which for a given account is assessed from the date and time of its most recent password change. The mysql.user system table indicates for each account when its password was last changed, and the server automatically treats the password as expired at client connection time if its age is greater than its permitted lifetime. This works with no explicit manual password expiration.

To establish automatic password-expiration policy globally, use the [default\_password\_lifetime](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_password_lifetime) system variable. Its default value is 0, which disables automatic password expiration. If the value of [default\_password\_lifetime](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_password_lifetime) is a positive integer ***N***, it indicates the permitted password lifetime, such that passwords must be changed every ***N*** days.

Examples:

* To establish a global policy that passwords have a lifetime of approximately six months, start the server with these lines in a server my.cnf file:
* [mysqld]

default\_password\_lifetime=180

* To establish a global policy such that passwords never expire, set [default\_password\_lifetime](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_password_lifetime) to 0:
* [mysqld]

default\_password\_lifetime=0

* [default\_password\_lifetime](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_password_lifetime) can also be set and persisted at runtime:
* SET PERSIST default\_password\_lifetime = 180;

SET PERSIST default\_password\_lifetime = 0;

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value to carry over to subsequent server restarts; see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html). To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the GLOBAL keyword rather than PERSIST.

The global password-expiration policy applies to all accounts that have not been set to override it. To establish policy for individual accounts, use the PASSWORD EXPIRE option of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), and [Section 13.7.1.1, “ALTER USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html).

Example account-specific statements:

* Require the password to be changed every 90 days:
* CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;

This expiration option overrides the global policy for all accounts named by the statement.

* Disable password expiration:
* CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE NEVER;

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE NEVER;

This expiration option overrides the global policy for all accounts named by the statement.

* Defer to the global expiration policy for all accounts named by the statement:
* CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE DEFAULT;

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE DEFAULT;

When a client successfully connects, the server determines whether the account password has expired:

* The server checks whether the password has been manually expired.
* Otherwise, the server checks whether the password age is greater than its permitted lifetime according to the automatic password expiration policy. If so, the server considers the password expired.

If the password is expired (whether manually or automatically), the server either disconnects the client or restricts the operations permitted to it (see [Section 6.2.16, “Server Handling of Expired Passwords”](https://dev.mysql.com/doc/refman/8.0/en/expired-password-handling.html)). Operations performed by a restricted client result in an error until the user establishes a new account password:

mysql> SELECT 1;

ERROR 1820 (HY000): You must reset your password using ALTER USER

statement before executing this statement.

mysql> ALTER USER USER() IDENTIFIED BY 'password';

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT 1;

+---+

| 1 |

+---+

| 1 |

+---+

1 row in set (0.00 sec)

After the client resets the password, the server restores normal access for the session, as well as for subsequent connections that use the account. It is also possible for an administrative user to reset the account password, but any existing restricted sessions for that account remain restricted. A client using the account must disconnect and reconnect before statements can be executed successfully.

**Note**

Although it is possible to “reset” an expired password by setting it to its current value, it is preferable, as a matter of good policy, to choose a different password. DBAs can enforce non-reuse by establishing an appropriate password-reuse policy. See [Password Reuse Policy](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#password-reuse-policy).

#### Password Reuse Policy

MySQL enables restrictions to be placed on reuse of previous passwords. Reuse restrictions can be established based on number of password changes, time elapsed, or both. Reuse policy can be established globally, and individual accounts can be set to either defer to the global policy or override the global policy with specific per-account behavior.

The password history for an account consists of passwords it has been assigned in the past. MySQL can restrict new passwords from being chosen from this history:

* If an account is restricted on the basis of number of password changes, a new password cannot be chosen from a specified number of the most recent passwords. For example, if the minimum number of password changes is set to 3, a new password cannot be the same as any of the most recent 3 passwords.
* If an account is restricted based on time elapsed, a new password cannot be chosen from passwords in the history that are newer than a specified number of days. For example, if the password reuse interval is set to 60, a new password must not be among those previously chosen within the last 60 days.

**Note**

The empty password does not count in the password history and is subject to reuse at any time.

To establish password-reuse policy globally, use the [password\_history](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_history) and [password\_reuse\_interval](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_reuse_interval) system variables.

Examples:

* To prohibit reusing any of the last 6 passwords or passwords newer than 365 days, put these lines in the server my.cnf file:
* [mysqld]
* password\_history=6

password\_reuse\_interval=365

* To set and persist the variables at runtime, use statements like this:
* SET PERSIST password\_history = 6;

SET PERSIST password\_reuse\_interval = 365;

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value to carry over to subsequent server restarts; see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html). To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the GLOBAL keyword rather than PERSIST.

The global password-reuse policy applies to all accounts that have not been set to override it. To establish policy for individual accounts, use the PASSWORD HISTORY and PASSWORD REUSE INTERVAL options of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), and [Section 13.7.1.1, “ALTER USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html).

Example account-specific statements:

* Require a minimum of 5 password changes before permitting reuse:
* CREATE USER 'jeffrey'@'localhost' PASSWORD HISTORY 5;

ALTER USER 'jeffrey'@'localhost' PASSWORD HISTORY 5;

This history-length option overrides the global policy for all accounts named by the statement.

* Require a minimum of 365 days elapsed before permitting reuse:
* CREATE USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL 365 DAY;

ALTER USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL 365 DAY;

This time-elapsed option overrides the global policy for all accounts named by the statement.

* To combine both types of reuse restrictions, use PASSWORD HISTORY and PASSWORD REUSE INTERVAL together:
* CREATE USER 'jeffrey'@'localhost'
* PASSWORD HISTORY 5
* PASSWORD REUSE INTERVAL 365 DAY;
* ALTER USER 'jeffrey'@'localhost'
* PASSWORD HISTORY 5

PASSWORD REUSE INTERVAL 365 DAY;

These options override both global policy reuse restrictions for all accounts named by the statement.

* Defer to the global policy for both types of reuse restrictions:
* CREATE USER 'jeffrey'@'localhost'
* PASSWORD HISTORY DEFAULT
* PASSWORD REUSE INTERVAL DEFAULT;
* ALTER USER 'jeffrey'@'localhost'
* PASSWORD HISTORY DEFAULT

PASSWORD REUSE INTERVAL DEFAULT;

#### Password Verification-Required Policy

As of MySQL 8.0.13, it is possible to require that attempts to change an account password be verified by specifying the current password to be replaced. This enables DBAs to prevent users from changing a password without proving that they know the current password. Such changes could otherwise occur, for example, if one user walks away from a terminal session temporarily without logging out, and a malicious user uses the session to change the original user's MySQL password. This can have unfortunate consequences:

* The original user becomes unable to access MySQL until the account password is reset by an administrator.
* Until the password reset occurs, the malicious user can access MySQL with the benign user's changed credentials.

Password-verification policy can be established globally, and individual accounts can be set to either defer to the global policy or override the global policy with specific per-account behavior.

For each account, its mysql.user row indicates whether there is an account-specific setting requiring verification of the current password for password change attempts. The setting is established by the PASSWORD REQUIRE option of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements:

* If the account setting is PASSWORD REQUIRE CURRENT, password changes must specify the current password.
* If the account setting is PASSWORD REQUIRE CURRENT OPTIONAL, password changes may but need not specify the current password.
* If the account setting is PASSWORD REQUIRE CURRENT DEFAULT, the [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) system variable determines the verification-required policy for the account:
  + If [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) is enabled, password changes must specify the current password.
  + If [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) is disabled, password changes may but need not specify the current password.

In other words, if the account setting is not PASSWORD REQUIRE CURRENT DEFAULT, the account setting takes precedence over the global policy established by the [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) system variable. Otherwise, the account defers to the [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) setting.

By default, password verification is optional: [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) is disabled and accounts created with no PASSWORD REQUIRE option default to PASSWORD REQUIRE CURRENT DEFAULT.

The following table shows how per-account settings interact with [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) system variable values to determine account password verification-required policy.

**Table 6.10 Password-Verification Policy**

| **Per-Account Setting** | **password\_require\_current System Variable** | **Password Changes Require Current Password?** |
| --- | --- | --- |
| PASSWORD REQUIRE CURRENT | OFF | Yes |
| PASSWORD REQUIRE CURRENT | ON | Yes |
| PASSWORD REQUIRE CURRENT OPTIONAL | OFF | No |
| PASSWORD REQUIRE CURRENT OPTIONAL | ON | No |
| PASSWORD REQUIRE CURRENT DEFAULT | OFF | No |
| PASSWORD REQUIRE CURRENT DEFAULT | ON | Yes |

**Note**

Privileged users can change any account password without specifying the current password, regardless of the verification-required policy. A privileged user is one who has the global [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege or the [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_update) privilege for the mysql system database.

To establish password-verification policy globally, use the [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) system variable. Its default value is OFF, so it is not required that account password changes specify the current password.

Examples:

* To establish a global policy that password changes must specify the current password, start the server with these lines in a server my.cnf file:
* [mysqld]

password\_require\_current=ON

* To set and persist [password\_require\_current](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_password_require_current) at runtime, use a statement such as one of these:
* SET PERSIST password\_require\_current = ON;

SET PERSIST password\_require\_current = OFF;

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value to carry over to subsequent server restarts; see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html). To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the GLOBAL keyword rather than PERSIST.

The global password verification-required policy applies to all accounts that have not been set to override it. To establish policy for individual accounts, use the PASSWORD REQUIRE options of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), and [Section 13.7.1.1, “ALTER USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html).

Example account-specific statements:

* Require that password changes specify the current password:
* CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT;

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT;

This verification option overrides the global policy for all accounts named by the statement.

* Do not require that password changes specify the current password (the current password may but need not be given):
* CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT OPTIONAL;

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT OPTIONAL;

This verification option overrides the global policy for all accounts named by the statement.

* Defer to the global password verification-required policy for all accounts named by the statement:
* CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT DEFAULT;

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT DEFAULT;

Verification of the current password comes into play when a user changes a password using the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) or [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statement. The examples use [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), which is preferred over [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html), but the principles described here are the same for both statements.

In password-change statements, a REPLACE clause specifies the current password to be replaced. Examples:

* Change the current user's password:

ALTER USER USER() IDENTIFIED BY 'auth\_string' REPLACE 'current\_auth\_string';

* Change a named user's password:
* ALTER USER 'jeffrey'@'localhost'
* IDENTIFIED BY 'auth\_string'

REPLACE 'current\_auth\_string';

* Change a named user's authentication plugin and password:
* ALTER USER 'jeffrey'@'localhost'
* IDENTIFIED WITH caching\_sha2\_password BY 'auth\_string'

REPLACE 'current\_auth\_string';

The REPLACE clause works like this:

* REPLACE must be given if password changes for the account are required to specify the current password, as verification that the user attempting to make the change actually knows the current password.
* REPLACE is optional if password changes for the account may but need not specify the current password.
* If REPLACE is specified, it must specify the correct current password, or an error occurs. This is true even if REPLACE is optional.
* REPLACE can be specified only when changing the account password for the current user. (This means that in the examples just shown, the statements that explicitly name the account for jeffrey fail unless the current user is jeffrey.) This is true even if the change is attempted for another user by a privileged user; however, such a user can change any password without specifying REPLACE.
* REPLACE is omitted from the binary log to avoid writing cleartext passwords to it.

#### Dual Password Support

As of MySQL 8.0.14, user accounts are permitted to have dual passwords, designated as primary and secondary passwords. Dual-password capability makes it possible to seamlessly perform credential changes in scenarios like this:

* A system has a large number of MySQL servers, possibly involving replication.
* Multiple applications connect to different MySQL servers.
* Periodic credential changes must be made to the account or accounts used by the applications to connect to the servers.

Consider how a credential change must be performed in the preceding type of scenario when an account is permitted only a single password. In this case, there must be close cooperation in the timing of when the account password change is made and propagated throughout all servers, and when all applications that use the account are updated to use the new password. This process may involve downtime during which servers or applications are unavailable.

With dual passwords, credential changes can be made more easily, in phases, without requiring close cooperation, and without downtime:

1. For each affected account, establish a new primary password on the servers, retaining the current password as the secondary password. This enables servers to recognize either the primary or secondary password for each account, while applications can continue to connect to the servers using the same password as previously (which is now the secondary password).
2. After the password change has propagated to all servers, modify applications that use any affected account to connect using the account primary password.
3. After all applications have been migrated from the secondary passwords to the primary passwords, the secondary passwords are no longer needed and can be discarded. After this change has propagated to all servers, only the primary password for each account can be used to connect. The credential change is now complete.

MySQL implements dual-password capability with syntax that saves and discards secondary passwords:

* The RETAIN CURRENT PASSWORD clause for the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statements saves an account current password as its secondary password when you assign a new primary password.
* The DISCARD OLD PASSWORD clause for [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) discards an account secondary password, leaving only the primary password.

Suppose that, for the previously described credential-change scenario, an account named 'appuser1'@'host1.example.com' is used by applications to connect to servers, and that the account password is to be changed from '***password\_a***' to '***password\_b***'.

To perform this change of credentials, use ALTER USER as follows:

1. On each server that is not a replica, establish '***password\_b***' as the new appuser1 primary password, retaining the current password as the secondary password:
2. ALTER USER 'appuser1'@'host1.example.com'
3. IDENTIFIED BY 'password\_b'

RETAIN CURRENT PASSWORD;

1. Wait for the password change to replicate throughout the system to all replicas.
2. Modify each application that uses the appuser1 account so that it connects to the servers using a password of '***password\_b***' rather than '***password\_a***'.
3. At this point, the secondary password is no longer needed. On each server that is not a replica, discard the secondary password:
4. ALTER USER 'appuser1'@'host1.example.com'

DISCARD OLD PASSWORD;

1. After the discard-password change has replicated to all replicas, the credential change is complete.

The RETAIN CURRENT PASSWORD and DISCARD OLD PASSWORD clauses have the following effects:

* RETAIN CURRENT PASSWORD retains an account current password as its secondary password, replacing any existing secondary password. The new password becomes the primary password, but clients can use the account to connect to the server using either the primary or secondary password. (Exception: If the new password specified by the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) or [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statement is empty, the secondary password becomes empty as well, even if RETAIN CURRENT PASSWORD is given.)
* If you specify RETAIN CURRENT PASSWORD for an account that has an empty primary password, the statement fails.
* If an account has a secondary password and you change its primary password without specifying RETAIN CURRENT PASSWORD, the secondary password remains unchanged.
* For [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), if you change the authentication plugin assigned to the account, the secondary password is discarded. If you change the authentication plugin and also specify RETAIN CURRENT PASSWORD, the statement fails.
* For [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), DISCARD OLD PASSWORD discards the secondary password, if one exists. The account retains only its primary password, and clients can use the account to connect to the server only with the primary password.

Statements that modify secondary passwords require these privileges:

* The [APPLICATION\_PASSWORD\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_application-password-admin) privilege is required to use the RETAIN CURRENT PASSWORD or DISCARD OLD PASSWORD clause for [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statements that apply to your own account. The privilege is required to manipulate your own secondary password because most users require only one password.
* If an account is to be permitted to manipulate secondary passwords for all accounts, it should be granted the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_create-user) privilege rather than [APPLICATION\_PASSWORD\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_application-password-admin).

#### Random Password Generation

As of MySQL 8.0.18, the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), and [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statements have the capability of generating random passwords for user accounts, as an alternative to requiring explicit administrator-specified literal passwords. See the description of each statement for details about the syntax. This section describes the characteristics common to generated random passwords.

By default, generated random passwords have a length of 20 characters. This length is controlled by the [generated\_random\_password\_length](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_generated_random_password_length) system variable, which has a range from 5 to 255.

For each account for which a statement generates a random password, the statement stores the password in the mysql.user system table, hashed appropriately for the account authentication plugin. The statement also returns the cleartext password in a row of a result set to make it available to the user or application executing the statement. The result set columns are named user, host, and generated password, indicating the user name and host name values that identify the affected row in the mysql.user system table, and the cleartext generated password.

mysql> CREATE USER

'u1'@'localhost' IDENTIFIED BY RANDOM PASSWORD,

'u2'@'%.example.com' IDENTIFIED BY RANDOM PASSWORD,

'u3'@'%.org' IDENTIFIED BY RANDOM PASSWORD;

+------+---------------+----------------------+

| user | host | generated password |

+------+---------------+----------------------+

| u1 | localhost | BA;42VpXqQ@i+y{&TDFF |

| u2 | %.example.com | YX5>XRAJRP@>sn9azmD4 |

| u3 | %.org | ;GfD44l,)C}PI/6)4TwZ |

+------+---------------+----------------------+

mysql> ALTER USER

'u1'@'localhost' IDENTIFIED BY RANDOM PASSWORD,

'u2'@'%.example.com' IDENTIFIED BY RANDOM PASSWORD;

+------+---------------+----------------------+

| user | host | generated password |

+------+---------------+----------------------+

| u1 | localhost | yhXBrBp.;Y6abB)e\_UWr |

| u2 | %.example.com | >M-vmjp9DTY6}hkp,RcC |

+------+---------------+----------------------+

mysql> SET PASSWORD FOR 'u3'@'%.org' TO RANDOM;

+------+-------+----------------------+

| user | host | generated password |

+------+-------+----------------------+

| u3 | %.org | o(.\_oNn)d;FC<vJIDg9M |

+------+-------+----------------------+

A [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), or [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html) statement that generates a random password for an account is written to the binary log as a [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) or [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement with an IDENTIFIED WITH ***auth\_plugin*** AS '***auth\_string***', clause, where ***auth\_plugin*** is the account authentication plugin and '***auth\_string***' is the account hashed password value.

If the validate\_password component is installed, the policy that it implements has no effect on generated passwords. (The purpose of password validation is to help humans create better passwords.)

#### Failed-Login Tracking and Temporary Account Locking

As of MySQL 8.0.19, administrators can configure user accounts such that too many consecutive login failures cause temporary account locking.

“Login failure” in this context means failure of the client to provide a correct password during a connection attempt. It does not include failure to connect for reasons such as unknown user or network issues. For accounts that have dual passwords (see [Dual Password Support](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#dual-passwords)), either account password counts as correct.

The required number of login failures and the lock time are configurable per account, using the FAILED\_LOGIN\_ATTEMPTS and PASSWORD\_LOCK\_TIME options of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements. Examples:

CREATE USER 'u1'@'localhost' IDENTIFIED BY 'password'

FAILED\_LOGIN\_ATTEMPTS 3 PASSWORD\_LOCK\_TIME 3;

ALTER USER 'u2'@'localhost'

FAILED\_LOGIN\_ATTEMPTS 4 PASSWORD\_LOCK\_TIME UNBOUNDED;

When too many consecutive login failures occur, the client receives an error that looks like this:

ERROR 3957 (HY000): Access denied for user user.

Account is blocked for D day(s) (R day(s) remaining)

due to N consecutive failed logins.

Use the options as follows:

* FAILED\_LOGIN\_ATTEMPTS ***N***

This option indicates whether to track account login attempts that specify an incorrect password. The number ***N*** specifies how many consecutive incorrect passwords cause temporary account locking.

* PASSWORD\_LOCK\_TIME {***N*** | UNBOUNDED}

This option indicates how long to lock the account after too many consecutive login attempts provide an incorrect password. The value is a number ***N*** to specify the number of days the account remains locked, or UNBOUNDED to specify that when an account enters the temporarily locked state, the duration of that state is unbounded and does not end until the account is unlocked. The conditions under which unlocking occurs are described later.

Permitted values of ***N*** for each option are in the range from 0 to 32767. A value of 0 disables the option.

Failed-login tracking and temporary account locking have these characteristics:

* For failed-login tracking and temporary locking to occur for an account, its FAILED\_LOGIN\_ATTEMPTS and PASSWORD\_LOCK\_TIME options both must be nonzero.
* For [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), if FAILED\_LOGIN\_ATTEMPTS or PASSWORD\_LOCK\_TIME is not specified, its implicit default value is 0 for all accounts named by the statement. This means that failed-login tracking and temporary account locking are disabled. (These implicit defaults also apply to accounts created prior to the introduction of failed-login tracking.)
* For [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), if FAILED\_LOGIN\_ATTEMPTS or PASSWORD\_LOCK\_TIME is not specified, its value remains unchanged for all accounts named by the statement.
* For temporary account locking to occur, password failures must be consecutive. Any successful login that occurs prior to reaching the FAILED\_LOGIN\_ATTEMPTS value for failed logins causes failure counting to reset. For example, if FAILED\_LOGIN\_ATTEMPTS is 4 and three consecutive password failures have occurred, one more failure is necessary for locking to begin. But if the next login succeeds, failed-login counting for the account is reset so that four consecutive failures are again required for locking.
* Once temporary locking begins, successful login cannot occur even with the correct password until either the lock duration has passed or the account is unlocked by one of the account-reset methods listed in the following discussion.

When the server reads the grant tables, it initializes state information for each account regarding whether failed-login tracking is enabled, whether the account is currently temporarily locked and when locking began if so, and the number of failures before temporary locking occurs if the account is not locked.

An account's state information can be reset, which means that failed-login counting is reset, and the account is unlocked if currently temporarily locked. Account resets can be global for all accounts or per account:

* A global reset of all accounts occurs for any of these conditions:
  + A server restart.
  + Execution of [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges). (Starting the server with [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) causes the grant tables not to be read, which disables failed-login tracking. In this case, the first execution of [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) causes the server to read the grant tables and enable failed-login tracking, in addition to resetting all accounts.)
* A per-account reset occurs for any of these conditions:
  + Sucessful login for the account.
  + The lock duration passes. In this case, failed-login counting resets at the time of the next login attempt.
  + Execution of an [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement for the account that sets either FAILED\_LOGIN\_ATTEMPTS or PASSWORD\_LOCK\_TIME (or both) to any value (including the current option value), or execution of an [ALTER USER ... UNLOCK](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement for the account.

Other [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements for the account have no effect on its current failed-login count or its locking state.

Failed-login tracking is tied to the login account that is used to check credentials. If user proxying is in use, tracking occurs for the proxy user, not the proxied user. That is, tracking is tied to the account indicated by [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user), not the account indicated by [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user)..

### Server Handling of Expired Passwords

MySQL provides password-expiration capability, which enables database administrators to require that users reset their password. Passwords can be expired manually, and on the basis of a policy for automatic expiration (see [Section 6.2.15, “Password Management”](https://dev.mysql.com/doc/refman/8.0/en/password-management.html)).

The [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement enables account password expiration. For example:

ALTER USER 'myuser'@'localhost' PASSWORD EXPIRE;

For each connection that uses an account with an expired password, the server either disconnects the client or restricts the client to “sandbox mode,” in which the server permits the client to perform only those operations necessary to reset the expired password. Which action is taken by the server depends on both client and server settings, as discussed later.

If the server disconnects the client, it returns an [ER\_MUST\_CHANGE\_PASSWORD\_LOGIN](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_must_change_password_login) error:

shell> mysql -u myuser -p

Password: \*\*\*\*\*\*

ERROR 1862 (HY000): Your password has expired. To log in you must

change it using a client that supports expired passwords.

If the server restricts the client to sandbox mode, these operations are permitted within the client session:

* The client can reset the account password with [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) or [SET PASSWORD](https://dev.mysql.com/doc/refman/8.0/en/set-password.html). After that has been done, the server restores normal access for the session, as well as for subsequent connections that use the account.

**Note**

Although it is possible to “reset” an expired password by setting it to its current value, it is preferable, as a matter of good policy, to choose a different password. DBAs can enforce non-reuse by establishing an appropriate password-reuse policy. See [Password Reuse Policy](https://dev.mysql.com/doc/refman/8.0/en/password-management.html#password-reuse-policy).

* The client can use the [SET](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) statement.

For any operation not permitted within the session, the server returns an [ER\_MUST\_CHANGE\_PASSWORD](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_must_change_password) error:

mysql> USE performance\_schema;

ERROR 1820 (HY000): You must reset your password using ALTER USER

statement before executing this statement.

mysql> SELECT 1;

ERROR 1820 (HY000): You must reset your password using ALTER USER

statement before executing this statement.

That is what normally happens for interactive invocations of the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client because by default such invocations are put in sandbox mode. To resume normal functioning, select a new password.

For noninteractive invocations of the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client (for example, in batch mode), the server normally disconnects the client if the password is expired. To permit noninteractive [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) invocations to stay connected so that the password can be changed (using the statements permitted in sandbox mode), add the [--connect-expired-password](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_connect-expired-password) option to the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) command.

As mentioned previously, whether the server disconnects an expired-password client or restricts it to sandbox mode depends on a combination of client and server settings. The following discussion describes the relevant settings and how they interact.

**Note**

This discussion applies only for accounts with expired passwords. If a client connects using a nonexpired password, the server handles the client normally.

On the client side, a given client indicates whether it can handle sandbox mode for expired passwords. For clients that use the C client library, there are two ways to do this:

* Pass the MYSQL\_OPT\_CAN\_HANDLE\_EXPIRED\_PASSWORDS flag to [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html) prior to connecting:
* bool arg = 1;
* mysql\_options(mysql,
* MYSQL\_OPT\_CAN\_HANDLE\_EXPIRED\_PASSWORDS,

&arg);

This is the technique used within the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client, which enables MYSQL\_OPT\_CAN\_HANDLE\_EXPIRED\_PASSWORDS if invoked interactively or with the [--connect-expired-password](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_connect-expired-password) option.

* Pass the CLIENT\_CAN\_HANDLE\_EXPIRED\_PASSWORDS flag to [mysql\_real\_connect()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html) at connect time:
* MYSQL mysql;
* mysql\_init(&mysql);
* if (!mysql\_real\_connect(&mysql,
* host, user, password, db,
* port, unix\_socket,
* CLIENT\_CAN\_HANDLE\_EXPIRED\_PASSWORDS))
* {
* ... handle error ...

}

Other MySQL Connectors have their own conventions for indicating readiness to handle sandbox mode. See the documentation for the Connector in which you are interested.

On the server side, if a client indicates that it can handle expired passwords, the server puts it in sandbox mode.

If a client does not indicate that it can handle expired passwords (or uses an older version of the client library that cannot so indicate), the server action depends on the value of the [disconnect\_on\_expired\_password](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_disconnect_on_expired_password) system variable:

* If [disconnect\_on\_expired\_password](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_disconnect_on_expired_password) is enabled (the default), the server disconnects the client with an [ER\_MUST\_CHANGE\_PASSWORD\_LOGIN](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_must_change_password_login) error.
* If [disconnect\_on\_expired\_password](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_disconnect_on_expired_password) is disabled, the server puts the client in sandbox mode.

### Pluggable Authentication

When a client connects to the MySQL server, the server uses the user name provided by the client and the client host to select the appropriate account row from the mysql.user system table. The server then authenticates the client, determining from the account row which authentication plugin applies to the client:

* If the server cannot find the plugin, an error occurs and the connection attempt is rejected.
* Otherwise, the server invokes that plugin to authenticate the user, and the plugin returns a status to the server indicating whether the user provided the correct password and is permitted to connect.

Pluggable authentication enables these important capabilities:

* **Choice of authentication methods.** Pluggable authentication makes it easy for DBAs to choose and change the authentication method used for individual MySQL accounts.
* **External authentication.** Pluggable authentication makes it possible for clients to connect to the MySQL server with credentials appropriate for authentication methods that store credentials elsewhere than in the mysql.user system table. For example, plugins can be created to use external authentication methods such as PAM, Windows login IDs, LDAP, or Kerberos.
* **Proxy users:** If a user is permitted to connect, an authentication plugin can return to the server a user name different from the name of the connecting user, to indicate that the connecting user is a proxy for another user (the proxied user). While the connection lasts, the proxy user is treated, for purposes of access control, as having the privileges of the proxied user. In effect, one user impersonates another. For more information, see [Section 6.2.18, “Proxy Users”](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html).

**Note**

If you start the server with the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option, authentication plugins are not used even if loaded because the server performs no client authentication and permits any client to connect. Because this is insecure, if the server is started with the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option, it also disables remote connections by enabling [skip\_networking](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_skip_networking).

#### Available Authentication Plugins

MySQL 8.0 provides these authentication plugins:

* A plugin that performs native authentication; that is, authentication based on the password hashing method in use from before the introduction of pluggable authentication in MySQL. The mysql\_native\_password plugin implements authentication based on this native password hashing method. See [Section 6.4.1.1, “Native Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/native-pluggable-authentication.html).
* Plugins that perform authentication using SHA-256 password hashing. This is stronger encryption than that available with native authentication. See [Section 6.4.1.3, “SHA-256 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/sha256-pluggable-authentication.html), and [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/caching-sha2-pluggable-authentication.html).
* A client-side plugin that sends the password to the server without hashing or encryption. This plugin is used in conjunction with server-side plugins that require access to the password exactly as provided by the client user. See [Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/cleartext-pluggable-authentication.html).
* A plugin that performs external authentication using PAM (Pluggable Authentication Modules), enabling MySQL Server to use PAM to authenticate MySQL users. This plugin supports proxy users as well. See [Section 6.4.1.5, “PAM Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/pam-pluggable-authentication.html).
* A plugin that performs external authentication on Windows, enabling MySQL Server to use native Windows services to authenticate client connections. Users who have logged in to Windows can connect from MySQL client programs to the server based on the information in their environment without specifying an additional password. This plugin supports proxy users as well. See [Section 6.4.1.6, “Windows Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/windows-pluggable-authentication.html).
* Plugins that perform authentication using LDAP (Lightweight Directory Access Protocol) to authenticate MySQL users by accessing directory services such as X.500. These plugins support proxy users as well. See [Section 6.4.1.7, “LDAP Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/ldap-pluggable-authentication.html).
* A plugin that prevents all client connections to any account that uses it. Use cases for this plugin include proxied accounts that should never permit direct login but are accessed only through proxy accounts and accounts that must be able to execute stored programs and views with elevated privileges without exposing those privileges to ordinary users. See [Section 6.4.1.8, “No-Login Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/no-login-pluggable-authentication.html).
* A plugin that authenticates clients that connect from the local host through the Unix socket file. See [Section 6.4.1.9, “Socket Peer-Credential Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/socket-pluggable-authentication.html).
* A test plugin that checks account credentials and logs success or failure to the server error log. This plugin is intended for testing and development purposes, and as an example of how to write an authentication plugin. See [Section 6.4.1.10, “Test Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/test-pluggable-authentication.html).

**Note**

For information about current restrictions on the use of pluggable authentication, including which connectors support which plugins, see [Restrictions on Pluggable Authentication](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html#pluggable-authentication-restrictions).

Third-party connector developers should read that section to determine the extent to which a connector can take advantage of pluggable authentication capabilities and what steps to take to become more compliant.

If you are interested in writing your own authentication plugins, see [Writing Authentication Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/writing-authentication-plugins.html).

#### Authentication Plugin Usage

This section provides general instructions for installing and using authentication plugins. For instructions specific to a given plugin, see the section that describes that plugin under [Section 6.4.1, “Authentication Plugins”](https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html).

In general, pluggable authentication uses a pair of corresponding plugins on the server and client sides, so you use a given authentication method like this:

* If necessary, install the plugin library or libraries containing the appropriate plugins. On the server host, install the library containing the server-side plugin, so that the server can use it to authenticate client connections. Similarly, on each client host, install the library containing the client-side plugin for use by client programs. Authentication plugins that are built in need not be installed.
* For each MySQL account that you create, specify the appropriate server-side plugin to use for authentication. If the account is to use the default authentication plugin, the account-creation statement need not specify the plugin explicitly. The [default\_authentication\_plugin](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_authentication_plugin) system variable configures the default authentication plugin.
* When a client connects, the server-side plugin tells the client program which client-side plugin to use for authentication.

In the case that an account uses an authentication method that is the default for both the server and the client program, the server need not communicate to the client which client-side plugin to use, and a round trip in client/server negotiation can be avoided.

For standard MySQL clients such as [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) and [**mysqladmin**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html), the [--default-auth=***plugin\_name***](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_default-auth) option can be specified on the command line as a hint about which client-side plugin the program can expect to use, although the server overrides this if the server-side plugin associated with the user account requires a different client-side plugin.

If the client program does not find the client-side plugin library file, specify a [--plugin-dir=***dir\_name***](https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_plugin-dir) option to indicate the plugin library directory location.

#### Authentication Plugin Client/Server Compatibility

Pluggable authentication enables flexibility in the choice of authentication methods for MySQL accounts, but in some cases client connections cannot be established due to authentication plugin incompatibility between the client and server.

The general compatibility principle for a successful client connection to a given account on a given server is that the client and server both must support the authentication method required by the account. Because authentication methods are implemented by authentication plugins, the client and server both must support the authentication plugin required by the account.

Authentication plugin incompatibilities can arise in various ways. Examples:

* Connect using a MySQL 5.7 client from 5.7.22 or lower to a MySQL 8.0 server account that authenticates with caching\_sha2\_password. This fails because the 5.7 client does not recognize the plugin, which was introduced in MySQL 8.0. (This issue is addressed in MySQL 5.7 as of 5.7.23, when caching\_sha2\_password client-side support was added to the MySQL client library and client programs.)
* Connect using a MySQL 5.5 client to a MySQL 5.6 server account that authenticates with sha256\_password. This fails because the 5.5 client does not recognize the plugin, which was introduced in MySQL 5.6.
* Connect using a MySQL 5.7 client to a pre-5.7 server account that authenticates with mysql\_old\_password. This fails for multiple reasons. First, such a connection requires --secure-auth=0, which is no longer a supported option. Even were it supported, the 5.7 client does not recognize the plugin because it was removed in MySQL 5.7.
* Connect using a MySQL 5.7 client from a Community distribution to a MySQL 5.7 Enterprise server account that authenticates using one of the Enterprise-only LDAP authentication plugins. This fails because the Community client does not have access to the Enterprise plugin.

In general, these compatibility issues do not arise when connections are made between a client and server from the same MySQL distribution. When connections are made between a client and server from different MySQL series, issues can arise. These issues are inherent in the development process when MySQL introduces new authentication plugins or removes old ones. To minimize the potential for incompatibilities, regularly upgrade the server, clients, and connectors on a timely basis.

#### Authentication Plugin Connector-Writing Considerations

Various implementations of the MySQL client/server protocol exist. The libmysqlclient C API client library is one implementation. Some MySQL connectors (typically those not written in C) provide their own implementation. However, not all protocol implementations handle plugin authentication the same way. This section describes an authentication issue that protocol implementors should take into account.

In the client/server protocol, the server tells connecting clients which authentication plugin it considers the default. If the protocol implementation used by the client tries to load the default plugin and that plugin does not exist on the client side, the load operation fails. This is an unnecessary failure if the default plugin is not the plugin actually required by the account to which the client is trying to connect.

If a client/server protocol implementation does not have its own notion of default authentication plugin and always tries to load the default plugin specified by the server, it fails with an error if that plugin is not available.

To avoid this problem, the protocol implementation used by the client should have its own default plugin and should use it as its first choice (or, alternatively, fall back to this default in case of failure to load the default plugin specified by the server). Example:

* In MySQL 5.7, libmysqlclient uses as its default choice either mysql\_native\_password or the plugin specified through the MYSQL\_DEFAULT\_AUTH option for [mysql\_options()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-options.html).
* When a 5.7 client tries to connect to an 8.0 server, the server specifies caching\_sha2\_password as its default authentication plugin, but the client still sends credential details per either mysql\_native\_password or whatever is specified through MYSQL\_DEFAULT\_AUTH.
* The only time the client loads the plugin specified by the server is for a change-plugin request, but in that case it can be any plugin depending on the user account. In this case, the client must try to load the plugin, and if that plugin is not available, an error is not optional.

#### Restrictions on Pluggable Authentication

The first part of this section describes general restrictions on the applicability of the pluggable authentication framework described at [Section 6.2.17, “Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html). The second part describes how third-party connector developers can determine the extent to which a connector can take advantage of pluggable authentication capabilities and what steps to take to become more compliant.

The term “native authentication” used here refers to authentication against passwords stored in the mysql.user system table. This is the same authentication method provided by older MySQL servers, before pluggable authentication was implemented. “Windows native authentication” refers to authentication using the credentials of a user who has already logged in to Windows, as implemented by the Windows Native Authentication plugin (“Windows plugin” for short).

##### General Pluggable Authentication Restrictions

* ***Connector/C++:*** Clients that use this connector can connect to the server only through accounts that use native authentication.

Exception: A connector supports pluggable authentication if it was built to link to libmysqlclient dynamically (rather than statically) and it loads the current version of libmysqlclient if that version is installed, or if the connector is recompiled from source to link against the current libmysqlclient.

For information about writing connectors to handle informatin from the server about the default server-side authentication plugin, see [Authentication Plugin Connector-Writing Considerations](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html#pluggable-authentication-connector-writing).

* ***Connector/NET:*** Clients that use Connector/NET can connect to the server through accounts that use native authentication or Windows native authentication.
* ***Connector/PHP:*** Clients that use this connector can connect to the server only through accounts that use native authentication, when compiled using the MySQL native driver for PHP (mysqlnd).
* ***Windows native authentication:*** Connecting through an account that uses the Windows plugin requires Windows Domain setup. Without it, NTLM authentication is used and then only local connections are possible; that is, the client and server must run on the same computer.
* ***Proxy users:*** Proxy user support is available to the extent that clients can connect through accounts authenticated with plugins that implement proxy user capability (that is, plugins that can return a user name different from that of the connecting user). For example, the PAM and Windows plugins support proxy users. The mysql\_native\_password and sha256\_password authentication plugins do not support proxy users by default, but can be configured to do so; see [Server Support for Proxy User Mapping](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html#proxy-users-server-user-mapping).
* ***Replication***: Replicas can not only employ replication user accounts using native authentication, but can also connect through replication user accounts that use nonnative authentication if the required client-side plugin is available. If the plugin is built into libmysqlclient, it is available by default. Otherwise, the plugin must be installed on the replica side in the directory named by the replica's [plugin\_dir](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_plugin_dir) system variable.
* [*FEDERATED*](https://dev.mysql.com/doc/refman/8.0/en/federated-storage-engine.html)***tables:*** A [FEDERATED](https://dev.mysql.com/doc/refman/8.0/en/federated-storage-engine.html) table can access the remote table only through accounts on the remote server that use native authentication.

##### Pluggable Authentication and Third-Party Connectors

Third-party connector developers can use the following guidelines to determine readiness of a connector to take advantage of pluggable authentication capabilities and what steps to take to become more compliant:

* An existing connector to which no changes have been made uses native authentication and clients that use the connector can connect to the server only through accounts that use native authentication. However, you should test the connector against a recent version of the server to verify that such connections still work without problem.

Exception: A connector might work with pluggable authentication without any changes if it links to libmysqlclient dynamically (rather than statically) and it loads the current version of libmysqlclient if that version is installed.

* To take advantage of pluggable authentication capabilities, a connector that is libmysqlclient-based should be relinked against the current version of libmysqlclient. This enables the connector to support connections though accounts that require client-side plugins now built into libmysqlclient (such as the cleartext plugin needed for PAM authentication and the Windows plugin needed for Windows native authentication). Linking with a current libmysqlclient also enables the connector to access client-side plugins installed in the default MySQL plugin directory (typically the directory named by the default value of the local server's [plugin\_dir](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_plugin_dir) system variable).

If a connector links to libmysqlclient dynamically, it must be ensured that the newer version of libmysqlclient is installed on the client host and that the connector loads it at runtime.

* Another way for a connector to support a given authentication method is to implement it directly in the client/server protocol. Connector/NET uses this approach to provide support for Windows native authentication.
* If a connector should be able to load client-side plugins from a directory different from the default plugin directory, it must implement some means for client users to specify the directory. Possibilities for this include a command-line option or environment variable from which the connector can obtain the directory name. Standard MySQL client programs such as [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) and [**mysqladmin**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) implement a --plugin-dir option.
* Proxy user support by a connector depends, as described earlier in this section, on whether the authentication methods that it supports permit proxy users.

### Proxy Users

The MySQL server authenticates client connections using authentication plugins. The plugin that authenticates a given connection may request that the connecting (external) user be treated as a different user for privilege-checking purposes. This enables the external user to be a proxy for the second user; that is, to assume the privileges of the second user:

* The external user is a “proxy user” (a user who can impersonate or become known as another user).
* The second user is a “proxied user” (a user whose identity and privileges can be assumed by a proxy user).

This section describes how the proxy user capability works. For general information about authentication plugins, see [Section 6.2.17, “Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/pluggable-authentication.html). For information about specific plugins, see [Section 6.4.1, “Authentication Plugins”](https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html). For information about writing authentication plugins that support proxy users, see [Implementing Proxy User Support in Authentication Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/writing-authentication-plugins-proxy-users.html).

**Note**

One administrative benefit to be gained by proxying is that the DBA can set up a single account with a set of privileges and then enable multiple proxy users to have those privileges without having to assign the privileges individually to each of those users. As an alternative to proxy users, DBAs may find that roles provide a suitable way to map users onto specific sets of named privileges. Each user can be granted a given single role to, in effect, be granted the appropriate set of privileges. See [Section 6.2.10, “Using Roles”](https://dev.mysql.com/doc/refman/8.0/en/roles.html).

#### Requirements for Proxy User Support

For proxying to occur for a given authentication plugin, these conditions must be satisfied:

* Proxying must be supported, either by the plugin itself, or by the MySQL server on behalf of the plugin. In the latter case, server support may need to be enabled explicitly; see [Server Support for Proxy User Mapping](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html#proxy-users-server-user-mapping).
* The account for the external proxy user must be set up to be authenticated by the plugin. Use the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement to associate an account with an authentication plugin, or [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) to change its plugin.
* The account for the proxied user must exist and be granted the privileges to be assumed by the proxy user. Use the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements for this.
* Normally, the proxied user is configured so that it can be used only in proxying scenaries and not for direct logins.
* The proxy user account must have the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for the proxied account. Use the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement for this.
* For a client connecting to the proxy account to be treated as a proxy user, the authentication plugin must return a user name different from the client user name, to indicate the user name of the proxied account that defines the privileges to be assumed by the proxy user.

Alternatively, for plugins that are provided proxy mapping by the server, the proxied user is determined from the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege held by the proxy user.

The proxy mechanism permits mapping only the external client user name to the proxied user name. There is no provision for mapping host names:

* When a client connects to the server, the server determines the proper account based on the user name passed by the client program and the host from which the client connects.
* If that account is a proxy account, the server attempts to determine the appropriate proxied account by finding a match for a proxied account using the user name returned by the authentication plugin and the host name of the proxy account. The host name in the proxied account is ignored.

#### Simple Proxy User Example

Consider the following account definitions:

-- create proxy account

CREATE USER 'employee\_ext'@'localhost'

IDENTIFIED WITH my\_auth\_plugin

AS 'my\_auth\_string';

-- create proxied account and grant its privileges;

-- use mysql\_no\_login plugin to prevent direct login

CREATE USER 'employee'@'localhost'

IDENTIFIED WITH mysql\_no\_login;

GRANT ALL

ON employees.\*

TO 'employee'@'localhost';

-- grant to proxy account the

-- PROXY privilege for proxied account

GRANT PROXY

ON 'employee'@'localhost'

TO 'employee\_ext'@'localhost';

When a client connects as employee\_ext from the local host, MySQL uses the plugin named my\_auth\_plugin to perform authentication. Suppose that my\_auth\_plugin returns a user name of employee to the server, based on the content of '***my\_auth\_string***' and perhaps by consulting some external authentication system. The name employee differs from employee\_ext, so returning employee serves as a request to the server to treat the employee\_ext external user, for purposes of privilege checking, as the employee local user.

In this case, employee\_ext is the proxy user and employee is the proxied user.

The server verifies that proxy authentication for employee is possible for the employee\_ext user by checking whether employee\_ext (the proxy user) has the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for employee (the proxied user). If this privilege has not been granted, an error occurs. Otherwise, employee\_ext assumes the privileges of employee. The server checks statements executed during the client session by employee\_ext against the privileges granted to employee. In this case, employee\_ext can access tables in the employees database.

The proxied account, employee, uses the mysql\_no\_login authentication plugin to prevent clients from using the account to log in directly. (This assumes that the plugin is installed. For instructions, see [Section 6.4.1.8, “No-Login Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/no-login-pluggable-authentication.html).) For alternative methods of protecting proxied accounts against direct use, see [Preventing Direct Login to Proxied Accounts](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html#preventing-proxied-account-direct-login).

When proxying occurs, the [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) and [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) functions can be used to see the difference between the connecting user (the proxy user) and the account whose privileges apply during the current session (the proxied user). For the example just described, those functions return these values:

mysql> SELECT USER(), CURRENT\_USER();

+------------------------+--------------------+

| USER() | CURRENT\_USER() |

+------------------------+--------------------+

| employee\_ext@localhost | employee@localhost |

+------------------------+--------------------+

In the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement that creates the proxy user account, the IDENTIFIED WITH clause that names the proxy-supporting authentication plugin is optionally followed by an AS '***auth\_string***' clause specifying a string that the server passes to the plugin when the user connects. If present, the string provides information that helps the plugin determine how to map the proxy (external) client user name to a proxied user name. It is up to each plugin whether it requires the AS clause. If so, the format of the authentication string depends on how the plugin intends to use it. Consult the documentation for a given plugin for information about the authentication string values it accepts.

#### Preventing Direct Login to Proxied Accounts

Proxied accounts generally are intended to be used only by means of proxy accounts. That is, clients connect using a proxy account, then are mapped onto and assume the privileges of the appropriate proxied user.

There are multiple ways to ensure that a proxied account cannot be used directly:

* Associate the account with the mysql\_no\_login authentication plugin. In this case, the account cannot be used for direct logins under any circumstances. This assumes that the plugin is installed. For instructions, see [Section 6.4.1.8, “No-Login Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/no-login-pluggable-authentication.html).
* Include the ACCOUNT LOCK option when you create the account. See [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html). With this method, also include a password so that if the account is unlocked later, it cannot be accessed with no password. (If the validate\_password component is enabled, creating an account without a password is not permitted, even if the account is locked. See [Section 6.4.3, “The Password Validation Component”](https://dev.mysql.com/doc/refman/8.0/en/validate-password.html).)
* Create the account with a password but do not tell anyone else the password. If you do not let anyone know the password for the account, clients cannot use it to connect directly to the MySQL server.

#### Granting and Revoking the PROXY Privilege

The [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege is needed to enable an external user to connect as and have the privileges of another user. To grant this privilege, use the [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statement. For example:

GRANT PROXY ON 'proxied\_user' TO 'proxy\_user';

The statement creates a row in the mysql.proxies\_priv grant table.

At connect time, ***proxy\_user*** must represent a valid externally authenticated MySQL user, and ***proxied\_user*** must represent a valid locally authenticated user. Otherwise, the connection attempt fails.

The corresponding [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) syntax is:

REVOKE PROXY ON 'proxied\_user' FROM 'proxy\_user';

MySQL [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) and [REVOKE](https://dev.mysql.com/doc/refman/8.0/en/revoke.html) syntax extensions work as usual. Examples:

-- grant PROXY to multiple accounts

GRANT PROXY ON 'a' TO 'b', 'c', 'd';

-- revoke PROXY from multiple accounts

REVOKE PROXY ON 'a' FROM 'b', 'c', 'd';

-- grant PROXY to an account and enable the account to grant

-- PROXY to the proxied account

GRANT PROXY ON 'a' TO 'd' WITH GRANT OPTION;

-- grant PROXY to default proxy account

GRANT PROXY ON 'a' TO ''@'';

The [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege can be granted in these cases:

* By a user that has GRANT PROXY ... WITH GRANT OPTION for ***proxied\_user***.
* By ***proxied\_user*** for itself: The value of [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) must exactly match [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) and ***proxied\_user***, for both the user name and host name parts of the account name.

The initial root account created during MySQL installation has the [PROXY ... WITH GRANT OPTION](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for ''@'', that is, for all users and all hosts. This enables root to set up proxy users, as well as to delegate to other accounts the authority to set up proxy users. For example, root can do this:

CREATE USER 'admin'@'localhost'

IDENTIFIED BY 'admin\_password';

GRANT PROXY

ON ''@''

TO 'admin'@'localhost'

WITH GRANT OPTION;

Those statements create an admin user that can manage all GRANT PROXY mappings. For example, admin can do this:

GRANT PROXY ON sally TO joe;

#### Default Proxy Users

To specify that some or all users should connect using a given authentication plugin, create a “blank” MySQL account with an empty user name and host name (''@''), associate it with that plugin, and let the plugin return the real authenticated user name (if different from the blank user). Suppose that there exists a plugin named ldap\_auth that implements LDAP authentication and maps connecting users onto either a developer or manager account. To set up proxying of users onto these accounts, use the following statements:

-- create default proxy account

CREATE USER ''@''

IDENTIFIED WITH ldap\_auth

AS 'O=Oracle, OU=MySQL';

-- create proxied accounts; use

-- mysql\_no\_login plugin to prevent direct login

CREATE USER 'developer'@'localhost'

IDENTIFIED WITH mysql\_no\_login;

CREATE USER 'manager'@'localhost'

IDENTIFIED WITH mysql\_no\_login;

-- grant to default proxy account the

-- PROXY privilege for proxied accounts

GRANT PROXY

ON 'manager'@'localhost'

TO ''@'';

GRANT PROXY

ON 'developer'@'localhost'

TO ''@'';

Now assume that a client connects as follows:

shell> mysql --user=myuser --password ...

Enter password: myuser\_password

The server does not find myuser defined as a MySQL user, but because there is a blank user account (''@'') that matches the client user name and host name, the server authenticates the client against that account. The server invokes the ldap\_auth authentication plugin and passes myuser and ***myuser\_password*** to it as the user name and password.

If the ldap\_auth plugin finds in the LDAP directory that ***myuser\_password*** is not the correct password for myuser, authentication fails and the server rejects the connection.

If the password is correct and ldap\_auth finds that myuser is a developer, it returns the user name developer to the MySQL server, rather than myuser. Returning a user name different from the client user name of myuser signals to the server that it should treat myuser as a proxy. The server verifies that ''@'' can authenticate as developer (because ''@'' has the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege to do so) and accepts the connection. The session proceeds with myuser having the privileges of the developer proxied user. (These privileges should be set up by the DBA using [GRANT](https://dev.mysql.com/doc/refman/8.0/en/grant.html) statements, not shown.) The [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) and [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) functions return these values:

mysql> SELECT USER(), CURRENT\_USER();

+------------------+---------------------+

| USER() | CURRENT\_USER() |

+------------------+---------------------+

| myuser@localhost | developer@localhost |

+------------------+---------------------+

If the plugin instead finds in the LDAP directory that myuser is a manager, it returns manager as the user name and the session proceeds with myuser having the privileges of the manager proxied user.

mysql> SELECT USER(), CURRENT\_USER();

+------------------+-------------------+

| USER() | CURRENT\_USER() |

+------------------+-------------------+

| myuser@localhost | manager@localhost |

+------------------+-------------------+

For simplicity, external authentication cannot be multilevel: Neither the credentials for developer nor those for manager are taken into account in the preceding example. However, they are still used if a client tries to connect and authenticate directly as the developer or manager account, which is why those proxied accounts should be protected against direct login (see [Preventing Direct Login to Proxied Accounts](https://dev.mysql.com/doc/refman/8.0/en/proxy-users.html#preventing-proxied-account-direct-login)).

#### Default Proxy User and Anonymous User Conflicts

If you intend to create a default proxy user, check for other existing “match any user” accounts that take precedence over the default proxy user because they can prevent that user from working as intended.

In the preceding discussion, the default proxy user account has '' in the host part, which matches any host. If you set up a default proxy user, take care to also check whether nonproxy accounts exist with the same user part and '%' in the host part, because '%' also matches any host, but has precedence over '' by the rules that the server uses to sort account rows internally (see [Section 6.2.6, “Access Control, Stage 1: Connection Verification”](https://dev.mysql.com/doc/refman/8.0/en/connection-access.html)).

Suppose that a MySQL installation includes these two accounts:

-- create default proxy account

CREATE USER ''@''

IDENTIFIED WITH some\_plugin

AS 'some\_auth\_string';

-- create anonymous account

CREATE USER ''@'%'

IDENTIFIED BY 'anon\_user\_password';

The first account (''@'') is intended as the default proxy user, used to authenticate connections for users who do not otherwise match a more-specific account. The second account (''@'%') is an anonymous-user account, which might have been created, for example, to enable users without their own account to connect anonymously.

Both accounts have the same user part (''), which matches any user. And each account has a host part that matches any host. Nevertheless, there is a priority in account matching for connection attempts because the matching rules sort a host of '%' ahead of ''. For accounts that do not match any more-specific account, the server attempts to authenticate them against ''@'%' (the anonymous user) rather than ''@'' (the default proxy user). As a result, the default proxy account is never used.

To avoid this problem, use one of the following strategies:

* Remove the anonymous account so that it does not conflict with the default proxy user.
* Use a more-specific default proxy user that matches ahead of the anonymous user. For example, to permit only localhost proxy connections, use ''@'localhost':
* CREATE USER ''@'localhost'
* IDENTIFIED WITH some\_plugin

AS 'some\_auth\_string';

In addition, modify any GRANT PROXY statements to name ''@'localhost' rather than ''@'' as the proxy user.

Be aware that this strategy prevents anonymous-user connections from localhost.

* Use a named default account rather than an anonymous default account. For an example of this technique, consult the instructions for using the authentication\_windows plugin. See [Section 6.4.1.6, “Windows Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/windows-pluggable-authentication.html).
* Create multiple proxy users, one for local connections and one for “everything else” (remote connections). This can be useful particularly when local users should have different privileges from remote users.

Create the proxy users:

-- create proxy user for local connections

CREATE USER ''@'localhost'

IDENTIFIED WITH some\_plugin

AS 'some\_auth\_string';

-- create proxy user for remote connections

CREATE USER ''@'%'

IDENTIFIED WITH some\_plugin

AS 'some\_auth\_string';

Create the proxied users:

-- create proxied user for local connections

CREATE USER 'developer'@'localhost'

IDENTIFIED WITH mysql\_no\_login;

-- create proxied user for remote connections

CREATE USER 'developer'@'%'

IDENTIFIED WITH mysql\_no\_login;

Grant to each proxy account the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for the corresponding proxied account:

GRANT PROXY

ON 'developer'@'localhost'

TO ''@'localhost';

GRANT PROXY

ON 'developer'@'%'

TO ''@'%';

Finally, grant appropriate privileges to the local and remote proxied users (not shown).

Assume that the some\_plugin/'***some\_auth\_string***' combination causes some\_plugin to map the client user name to developer. Local connections match the ''@'localhost' proxy user, which maps to the 'developer'@'localhost' proxied user. Remote connections match the ''@'%' proxy user, which maps to the 'developer'@'%' proxied user.

#### Server Support for Proxy User Mapping

Some authentication plugins implement proxy user mapping for themselves (for example, the PAM and Windows authentication plugins). Other authentication plugins do not support proxy users by default. Of these, some can request that the MySQL server itself map proxy users according to granted proxy privileges: mysql\_native\_password, sha256\_password. If the [check\_proxy\_users](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_check_proxy_users) system variable is enabled, the server performs proxy user mapping for any authentication plugins that make such a request:

* By default, [check\_proxy\_users](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_check_proxy_users) is disabled, so the server performs no proxy user mapping even for authentication plugins that request server support for proxy users.
* If [check\_proxy\_users](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_check_proxy_users) is enabled, it may also be necessary to enable a plugin-specific system variable to take advantage of server proxy user mapping support:
  + For the mysql\_native\_password plugin, enable [mysql\_native\_password\_proxy\_users](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mysql_native_password_proxy_users).
  + For the sha256\_password plugin, enable [sha256\_password\_proxy\_users](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sha256_password_proxy_users).

For example, to enable all the preceding capabilities, start the server with these lines in the my.cnf file:

[mysqld]

check\_proxy\_users=ON

mysql\_native\_password\_proxy\_users=ON

sha256\_password\_proxy\_users=ON

Assuming that the relevant system variables have been enabled, create the proxy user as usual using [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), then grant it the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege to a single other account to be treated as the proxied user. When the server receives a successful connection request for the proxy user, it finds that the user has the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege and uses it to determine the proper proxied user.

-- create proxy account

CREATE USER 'proxy\_user'@'localhost'

IDENTIFIED WITH mysql\_native\_password

BY 'password';

-- create proxied account and grant its privileges;

-- use mysql\_no\_login plugin to prevent direct login

CREATE USER 'proxied\_user'@'localhost'

IDENTIFIED WITH mysql\_no\_login;

-- grant privileges to proxied account

GRANT ...

ON ...

TO 'proxied\_user'@'localhost';

-- grant to proxy account the

-- PROXY privilege for proxied account

GRANT PROXY

ON 'proxied\_user'@'localhost'

TO 'proxy\_user'@'localhost';

To use the proxy account, connect to the server using its name and password:

shell> mysql -u proxy\_user -p

Enter password: *(*enter proxy\_user password here*)*

Authentication succeeds, the server finds that proxy\_user has the [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege for proxied\_user, and the session proceeds with proxy\_user having the privileges of proxied\_user.

Proxy user mapping performed by the server is subject to these restrictions:

* The server does not proxy to or from an anonymous user, even if the associated [PROXY](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_proxy) privilege is granted.
* When a single account has been granted proxy privileges for more than one proxied account, server proxy user mapping is nondeterministic. Therefore, granting to a single account proxy privileges for multiple proxied accounts is discouraged.

#### Proxy User System Variables

Two system variables help trace the proxy login process:

* [proxy\_user](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_proxy_user): This value is NULL if proxying is not used. Otherwise, it indicates the proxy user account. For example, if a client authenticates through the ''@'' proxy account, this variable is set as follows:
* mysql> SELECT @@proxy\_user;
* +--------------+
* | @@proxy\_user |
* +--------------+
* | ''@'' |

+--------------+

* [external\_user](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_external_user): Sometimes the authentication plugin may use an external user to authenticate to the MySQL server. For example, when using Windows native authentication, a plugin that authenticates using the windows API does not need the login ID passed to it. However, it still uses a Windows user ID to authenticate. The plugin may return this external user ID (or the first 512 UTF-8 bytes of it) to the server using the external\_user read-only session variable. If the plugin does not set this variable, its value is NULL.

### Account Locking

MySQL supports locking and unlocking user accounts using the ACCOUNT LOCK and ACCOUNT UNLOCK clauses for the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) and [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statements:

* When used with [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html), these clauses specify the initial locking state for a new account. In the absence of either clause, the account is created in an unlocked state.

If the validate\_password component is enabled, creating an account without a password is not permitted, even if the account is locked. See [Section 6.4.3, “The Password Validation Component”](https://dev.mysql.com/doc/refman/8.0/en/validate-password.html).

* When used with [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html), these clauses specify the new locking state for an existing account. In the absence of either clause, the account locking state remains unchanged.

As of MySQL 8.0.19, [ALTER USER ... UNLOCK](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) unlocks any account named by the statement that is temporarily locked due to too many failed logins. See [Section 6.2.15, “Password Management”](https://dev.mysql.com/doc/refman/8.0/en/password-management.html).

Account locking state is recorded in the account\_locked column of the mysql.user system table. The output from [SHOW CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/show-create-user.html) indicates whether an account is locked or unlocked.

If a client attempts to connect to a locked account, the attempt fails. The server increments the [Locked\_connects](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Locked_connects) status variable that indicates the number of attempts to connect to a locked account, returns an [ER\_ACCOUNT\_HAS\_BEEN\_LOCKED](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_account_has_been_locked) error, and writes a message to the error log:

Access denied for user 'user\_name'@'host\_name'.

Account is locked.

Locking an account does not affect being able to connect using a proxy user that assumes the identity of the locked account. It also does not affect the ability to execute stored programs or views that have a DEFINER attribute naming the locked account. That is, the ability to use a proxied account or stored programs or views is not affected by locking the account.

The account-locking capability depends on the presence of the account\_locked column in the mysql.user system table. For upgrades from MySQL versions older than 5.7.6, perform the MySQL upgrade procedure to ensure that this column exists. See [Section 2.11, “Upgrading MySQL”](https://dev.mysql.com/doc/refman/8.0/en/upgrading.html). For nonupgraded installations that have no account\_locked column, the server treats all accounts as unlocked, and using the ACCOUNT LOCK or ACCOUNT UNLOCK clauses produces an error.

### Setting Account Resource Limits

One means of restricting client use of MySQL server resources is to set the global [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) system variable to a nonzero value. This limits the number of simultaneous connections that can be made by any given account, but places no limits on what a client can do once connected. In addition, setting [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) does not enable management of individual accounts. Both types of control are of interest to MySQL administrators.

To address such concerns, MySQL permits limits for individual accounts on use of these server resources:

* The number of queries an account can issue per hour
* The number of updates an account can issue per hour
* The number of times an account can connect to the server per hour
* The number of simultaneous connections to the server by an account

Any statement that a client can issue counts against the query limit. Only statements that modify databases or tables count against the update limit.

An “account” in this context corresponds to a row in the mysql.user system table. That is, a connection is assessed against the User and Host values in the user table row that applies to the connection. For example, an account 'usera'@'%.example.com' corresponds to a row in the user table that has User and Host values of usera and %.example.com, to permit usera to connect from any host in the example.com domain. In this case, the server applies resource limits in this row collectively to all connections by usera from any host in the example.com domain because all such connections use the same account.

Before MySQL 5.0, an “account” was assessed against the actual host from which a user connects. This older method of accounting may be selected by starting the server with the [--old-style-user-limits](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_old-style-user-limits) option. In this case, if usera connects simultaneously from host1.example.com and host2.example.com, the server applies the account resource limits separately to each connection. If usera connects again from host1.example.com, the server applies the limits for that connection together with the existing connection from that host.

To establish resource limits for an account at account-creation time, use the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement. To modify the limits for an existing account, use [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html). Provide a WITH clause that names each resource to be limited. The default value for each limit is zero (no limit). For example, to create a new account that can access the customer database, but only in a limited fashion, issue these statements:

mysql> CREATE USER 'francis'@'localhost' IDENTIFIED BY 'frank'

-> WITH MAX\_QUERIES\_PER\_HOUR 20

-> MAX\_UPDATES\_PER\_HOUR 10

-> MAX\_CONNECTIONS\_PER\_HOUR 5

-> MAX\_USER\_CONNECTIONS 2;

The limit types need not all be named in the WITH clause, but those named can be present in any order. The value for each per-hour limit should be an integer representing a count per hour. For MAX\_USER\_CONNECTIONS, the limit is an integer representing the maximum number of simultaneous connections by the account. If this limit is set to zero, the global [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) system variable value determines the number of simultaneous connections. If [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) is also zero, there is no limit for the account.

To modify limits for an existing account, use an [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement. The following statement changes the query limit for francis to 100:

mysql> ALTER USER 'francis'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 100;

The statement modifies only the limit value specified and leaves the account otherwise unchanged.

To remove a limit, set its value to zero. For example, to remove the limit on how many times per hour francis can connect, use this statement:

mysql> ALTER USER 'francis'@'localhost' WITH MAX\_CONNECTIONS\_PER\_HOUR 0;

As mentioned previously, the simultaneous-connection limit for an account is determined from the MAX\_USER\_CONNECTIONS limit and the [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) system variable. Suppose that the global [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) value is 10 and three accounts have individual resource limits specified as follows:

ALTER USER 'user1'@'localhost' WITH MAX\_USER\_CONNECTIONS 0;

ALTER USER 'user2'@'localhost' WITH MAX\_USER\_CONNECTIONS 5;

ALTER USER 'user3'@'localhost' WITH MAX\_USER\_CONNECTIONS 20;

user1 has a connection limit of 10 (the global [max\_user\_connections](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_user_connections) value) because it has a MAX\_USER\_CONNECTIONS limit of zero. user2 and user3 have connection limits of 5 and 20, respectively, because they have nonzero MAX\_USER\_CONNECTIONS limits.

The server stores resource limits for an account in the user table row corresponding to the account. The max\_questions, max\_updates, and max\_connections columns store the per-hour limits, and the max\_user\_connections column stores the MAX\_USER\_CONNECTIONS limit. (See [Section 6.2.3, “Grant Tables”](https://dev.mysql.com/doc/refman/8.0/en/grant-tables.html).)

Resource-use counting takes place when any account has a nonzero limit placed on its use of any of the resources.

As the server runs, it counts the number of times each account uses resources. If an account reaches its limit on number of connections within the last hour, the server rejects further connections for the account until that hour is up. Similarly, if the account reaches its limit on the number of queries or updates, the server rejects further queries or updates until the hour is up. In all such cases, the server issues appropriate error messages.

Resource counting occurs per account, not per client. For example, if your account has a query limit of 50, you cannot increase your limit to 100 by making two simultaneous client connections to the server. Queries issued on both connections are counted together.

The current per-hour resource-use counts can be reset globally for all accounts, or individually for a given account:

* To reset the current counts to zero for all accounts, issue a [FLUSH USER\_RESOURCES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-user-resources) statement. The counts also can be reset by reloading the grant tables (for example, with a [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement or a [**mysqladmin reload**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command).
* The counts for an individual account can be reset to zero by setting any of its limits again. Specify a limit value equal to the value currently assigned to the account.

Per-hour counter resets do not affect the MAX\_USER\_CONNECTIONS limit.

All counts begin at zero when the server starts. Counts do not carry over through server restarts.

For the MAX\_USER\_CONNECTIONS limit, an edge case can occur if the account currently has open the maximum number of connections permitted to it: A disconnect followed quickly by a connect can result in an error ([ER\_TOO\_MANY\_USER\_CONNECTIONS](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_too_many_user_connections) or [ER\_USER\_LIMIT\_REACHED](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_user_limit_reached)) if the server has not fully processed the disconnect by the time the connect occurs. When the server finishes disconnect processing, another connection is once more permitted.

### Troubleshooting Problems Connecting to MySQL

If you encounter problems when you try to connect to the MySQL server, the following items describe some courses of action you can take to correct the problem.

* Make sure that the server is running. If it is not, clients cannot connect to it. For example, if an attempt to connect to the server fails with a message such as one of those following, one cause might be that the server is not running:
* shell> mysql
* ERROR 2003: Can't connect to MySQL server on 'host\_name' (111)
* shell> mysql
* ERROR 2002: Can't connect to local MySQL server through socket

'/tmp/mysql.sock' (111)

* It might be that the server is running, but you are trying to connect using a TCP/IP port, named pipe, or Unix socket file different from the one on which the server is listening. To correct this when you invoke a client program, specify a [--port](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_port) option to indicate the proper port number, or a [--socket](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_socket) option to indicate the proper named pipe or Unix socket file. To find out where the socket file is, you can use this command:

shell> netstat -ln | grep mysql

* Make sure that the server has not been configured to ignore network connections or (if you are attempting to connect remotely) that it has not been configured to listen only locally on its network interfaces. If the server was started with the [skip\_networking](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_skip_networking) system variable enabled, no TCP/IP connections are accepted. If the server was started with the [bind\_address](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_bind_address) system variable set to 127.0.0.1, it listens for TCP/IP connections only locally on the loopback interface and does not accept remote connections.
* Check to make sure that there is no firewall blocking access to MySQL. Your firewall may be configured on the basis of the application being executed, or the port number used by MySQL for communication (3306 by default). Under Linux or Unix, check your IP tables (or similar) configuration to ensure that the port has not been blocked. Under Windows, applications such as ZoneAlarm or Windows Firewall may need to be configured not to block the MySQL port.
* The grant tables must be properly set up so that the server can use them for access control. For some distribution types (such as binary distributions on Windows, or RPM and DEB distributions on Linux), the installation process initializes the MySQL data directory, including the mysql system database containing the grant tables. For distributions that do not do this, you must initialize the data directory manually. For details, see [Section 2.10, “Postinstallation Setup and Testing”](https://dev.mysql.com/doc/refman/8.0/en/postinstallation.html).

To determine whether you need to initialize the grant tables, look for a mysql directory under the data directory. (The data directory normally is named data or var and is located under your MySQL installation directory.) Make sure that you have a file named user.MYD in the mysql database directory. If not, initialize the data directory. After doing so and starting the server, you should be able to connect to the server.

* After a fresh installation, if you try to log on to the server as root without using a password, you might get the following error message.
* shell> mysql -u root

ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

It means a root password has already been assigned during installation and it has to be supplied. See [Section 2.10.4, “Securing the Initial MySQL Account”](https://dev.mysql.com/doc/refman/8.0/en/default-privileges.html) on the different ways the password could have been assigned and, in some cases, how to find it. If you need to reset the root password, see instructions in [Section B.3.3.2, “How to Reset the Root Password”](https://dev.mysql.com/doc/refman/8.0/en/resetting-permissions.html). After you have found or reset your password, log on again as root using the [--password](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password) (or [-p](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password)) option:

shell> mysql -u root -p

Enter password:

However, the server is going to let you connect as root without using a password if you have initialized MySQL using [**mysqld --initialize-insecure**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) (see [Section 2.10.1, “Initializing the Data Directory”](https://dev.mysql.com/doc/refman/8.0/en/data-directory-initialization.html) for details). That is a security risk, so you should set a password for the root account; see [Section 2.10.4, “Securing the Initial MySQL Account”](https://dev.mysql.com/doc/refman/8.0/en/default-privileges.html) for instructions.

* If you have updated an existing MySQL installation to a newer version, did you perform the MySQL upgrade procedure? If not, do so. The structure of the grant tables changes occasionally when new capabilities are added, so after an upgrade you should always make sure that your tables have the current structure. For instructions, see [Section 2.11, “Upgrading MySQL”](https://dev.mysql.com/doc/refman/8.0/en/upgrading.html).
* If a client program receives the following error message when it tries to connect, it means that the server expects passwords in a newer format than the client is capable of generating:
* shell> mysql
* Client does not support authentication protocol requested

by server; consider upgrading MySQL client

* Remember that client programs use connection parameters specified in option files or environment variables. If a client program seems to be sending incorrect default connection parameters when you have not specified them on the command line, check any applicable option files and your environment. For example, if you get Access denied when you run a client without any options, make sure that you have not specified an old password in any of your option files!

You can suppress the use of option files by a client program by invoking it with the [--no-defaults](https://dev.mysql.com/doc/refman/8.0/en/option-file-options.html#option_general_no-defaults) option. For example:

shell> mysqladmin --no-defaults -u root version

The option files that clients use are listed in [Section 4.2.2.2, “Using Option Files”](https://dev.mysql.com/doc/refman/8.0/en/option-files.html). Environment variables are listed in [Section 4.9, “Environment Variables”](https://dev.mysql.com/doc/refman/8.0/en/environment-variables.html).

* If you get the following error, it means that you are using an incorrect root password:
* shell> mysqladmin -u root -pxxxx ver

Access denied for user 'root'@'localhost' (using password: YES)

If the preceding error occurs even when you have not specified a password, it means that you have an incorrect password listed in some option file. Try the [--no-defaults](https://dev.mysql.com/doc/refman/8.0/en/option-file-options.html#option_general_no-defaults) option as described in the previous item.

For information on changing passwords, see [Section 6.2.14, “Assigning Account Passwords”](https://dev.mysql.com/doc/refman/8.0/en/assigning-passwords.html).

If you have lost or forgotten the root password, see [Section B.3.3.2, “How to Reset the Root Password”](https://dev.mysql.com/doc/refman/8.0/en/resetting-permissions.html).

* localhost is a synonym for your local host name, and is also the default host to which clients try to connect if you specify no host explicitly.

You can use a [--host=127.0.0.1](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_host) option to name the server host explicitly. This causes a TCP/IP connection to the local [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) server. You can also use TCP/IP by specifying a [--host](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_host) option that uses the actual host name of the local host. In this case, the host name must be specified in a user table row on the server host, even though you are running the client program on the same host as the server.

* The Access denied error message tells you who you are trying to log in as, the client host from which you are trying to connect, and whether you were using a password. Normally, you should have one row in the user table that exactly matches the host name and user name that were given in the error message. For example, if you get an error message that contains using password: NO, it means that you tried to log in without a password.
* If you get an Access denied error when trying to connect to the database with mysql -u ***user\_name***, you may have a problem with the user table. Check this by executing mysql -u root mysql and issuing this SQL statement:

SELECT \* FROM user;

The result should include a row with the Host and User columns matching your client's host name and your MySQL user name.

* If the following error occurs when you try to connect from a host other than the one on which the MySQL server is running, it means that there is no row in the user table with a Host value that matches the client host:

Host ... is not allowed to connect to this MySQL server

You can fix this by setting up an account for the combination of client host name and user name that you are using when trying to connect.

If you do not know the IP address or host name of the machine from which you are connecting, you should put a row with '%' as the Host column value in the user table. After trying to connect from the client machine, use a SELECT USER() query to see how you really did connect. Then change the '%' in the user table row to the actual host name that shows up in the log. Otherwise, your system is left insecure because it permits connections from any host for the given user name.

On Linux, another reason that this error might occur is that you are using a binary MySQL version that is compiled with a different version of the glibc library than the one you are using. In this case, you should either upgrade your operating system or glibc, or download a source distribution of MySQL version and compile it yourself. A source RPM is normally trivial to compile and install, so this is not a big problem.

* If you specify a host name when trying to connect, but get an error message where the host name is not shown or is an IP address, it means that the MySQL server got an error when trying to resolve the IP address of the client host to a name:
* shell> mysqladmin -u root -pxxxx -h some\_hostname ver

Access denied for user 'root'@'' (using password: YES)

If you try to connect as root and get the following error, it means that you do not have a row in the user table with a User column value of 'root' and that [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) cannot resolve the host name for your client:

Access denied for user ''@'unknown'

These errors indicate a DNS problem. To fix it, execute [**mysqladmin flush-hosts**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) to reset the internal DNS host cache. See [Section 5.1.12.3, “DNS Lookups and the Host Cache”](https://dev.mysql.com/doc/refman/8.0/en/host-cache.html).

Some permanent solutions are:

* + Determine what is wrong with your DNS server and fix it.
  + Specify IP addresses rather than host names in the MySQL grant tables.
  + Put an entry for the client machine name in /etc/hosts on Unix or \windows\hosts on Windows.
  + Start [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) with the [skip\_name\_resolve](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_skip_name_resolve) system variable enabled.
  + Start [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) with the [--skip-host-cache](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-host-cache) option.
  + On Unix, if you are running the server and the client on the same machine, connect to localhost. For connections to localhost, MySQL programs attempt to connect to the local server by using a Unix socket file, unless there are connection parameters specified to ensure that the client makes a TCP/IP connection.
  + On Windows, if you are running the server and the client on the same machine and the server supports named pipe connections, connect to the host name . (period). Connections to . use a named pipe rather than TCP/IP.
* If mysql -u root works but mysql -h ***your\_hostname*** -u root results in Access denied (where ***your\_hostname*** is the actual host name of the local host), you may not have the correct name for your host in the user table. A common problem here is that the Host value in the user table row specifies an unqualified host name, but your system's name resolution routines return a fully qualified domain name (or vice versa). For example, if you have a row with host 'pluto' in the user table, but your DNS tells MySQL that your host name is 'pluto.example.com', the row does not work. Try adding a row to the user table that contains the IP address of your host as the Host column value. (Alternatively, you could add a row to the user table with a Host value that contains a wildcard (for example, 'pluto.%'). However, use of Host values ending with % is insecure and is not recommended!)
* If mysql -u ***user\_name*** works but mysql -u ***user\_name*** ***some\_db*** does not, you have not granted access to the given user for the database named ***some\_db***.
* If mysql -u ***user\_name*** works when executed on the server host, but mysql -h ***host\_name*** -u ***user\_name*** does not work when executed on a remote client host, you have not enabled access to the server for the given user name from the remote host.
* If you cannot figure out why you get Access denied, remove from the user table all rows that have Host values containing wildcards (rows that contain '%' or '\_' characters). A very common error is to insert a new row with Host='%' and User='***some\_user***', thinking that this enables you to specify localhost to connect from the same machine. The reason that this does not work is that the default privileges include a row with Host='localhost' and User=''. Because that row has a Host value 'localhost' that is more specific than '%', it is used in preference to the new row when connecting from localhost! The correct procedure is to insert a second row with Host='localhost' and User='***some\_user***', or to delete the row with Host='localhost' and User=''. After deleting the row, remember to issue a [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement to reload the grant tables. See also [Section  “Access Control, Stage 1: Connection Verification”](https://dev.mysql.com/doc/refman/8.0/en/connection-access.html).
* If you are able to connect to the MySQL server, but get an Access denied message whenever you issue a [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/8.0/en/select-into.html) or [LOAD DATA](https://dev.mysql.com/doc/refman/8.0/en/load-data.html) statement, your row in the user table does not have the [FILE](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_file) privilege enabled.
* If you change the grant tables directly (for example, by using [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html), [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), or [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html) statements) and your changes seem to be ignored, remember that you must execute a [FLUSH PRIVILEGES](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-privileges) statement or a [**mysqladmin flush-privileges**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) command to cause the server to reload the privilege tables. Otherwise, your changes have no effect until the next time the server is restarted. Remember that after you change the root password with an [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html) statement, you do not need to specify the new password until after you flush the privileges, because the server does not know until then that you have changed the password.
* If your privileges seem to have changed in the middle of a session, it may be that a MySQL administrator has changed them. Reloading the grant tables affects new client connections, but it also affects existing connections as indicated in [Section  “When Privilege Changes Take Effect”](https://dev.mysql.com/doc/refman/8.0/en/privilege-changes.html).
* If you have access problems with a Perl, PHP, Python, or ODBC program, try to connect to the server with mysql -u ***user\_name*** ***db\_name*** or mysql -u ***user\_name*** -p***password*** ***db\_name***. If you are able to connect using the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client, the problem lies with your program, not with the access privileges. (There is no space between -p and the password; you can also use the [--password=***password***](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password) syntax to specify the password. If you use the -p or [--password](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_password) option with no password value, MySQL prompts you for the password.)
* For testing purposes, start the [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) server with the [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) option. Then you can change the MySQL grant tables and use the [SHOW GRANTS](https://dev.mysql.com/doc/refman/8.0/en/show-grants.html) statement to check whether your modifications have the desired effect. When you are satisfied with your changes, execute [**mysqladmin flush-privileges**](https://dev.mysql.com/doc/refman/8.0/en/mysqladmin.html) to tell the [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) server to reload the privileges. This enables you to begin using the new grant table contents without stopping and restarting the server.
* If everything else fails, start the [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) server with a debugging option (for example, [--debug=d,general,query](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_debug)). This prints host and user information about attempted connections, as well as information about each command issued. See [Section 5.9.4, “The DBUG Package”](https://dev.mysql.com/doc/refman/8.0/en/dbug-package.html).
* If you have any other problems with the MySQL grant tables and ask on the [MySQL Community Slack](https://mysqlcommunity.slack.com/), always provide a dump of the MySQL grant tables. You can dump the tables with the [**mysqldump mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html) command. To file a bug report, see the instructions at [Section 1.6, “How to Report Bugs or Problems”](https://dev.mysql.com/doc/refman/8.0/en/bug-reports.html). In some cases, you may need to restart [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) with [--skip-grant-tables](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_skip-grant-tables) to run [**mysqldump**](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html).

### Based Account Activity Auditing

Applications can use the following guidelines to perform SQL-based auditing that ties database activity to MySQL accounts.

MySQL accounts correspond to rows in the mysql.user system table. When a client connects successfully, the server authenticates the client to a particular row in this table. The User and Host column values in this row uniquely identify the account and correspond to the '***user\_name***'@'***host\_name***' format in which account names are written in SQL statements.

The account used to authenticate a client determines which privileges the client has. Normally, the [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) function can be invoked to determine which account this is for the client user. Its value is constructed from the User and Host columns of the user table row for the account.

However, there are circumstances under which the [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) value corresponds not to the client user but to a different account. This occurs in contexts when privilege checking is not based the client's account:

* Stored routines (procedures and functions) defined with the SQL SECURITY DEFINER characteristic
* Views defined with the SQL SECURITY DEFINER characteristic
* Triggers and events

In those contexts, privilege checking is done against the DEFINER account and [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) refers to that account, not to the account for the client who invoked the stored routine or view or who caused the trigger to activate. To determine the invoking user, you can call the [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) function, which returns a value indicating the actual user name provided by the client and the host from which the client connected. However, this value does not necessarily correspond directly to an account in the user table, because the [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) value never contains wildcards, whereas account values (as returned by [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user)) may contain user name and host name wildcards.

For example, a blank user name matches any user, so an account of ''@'localhost' enables clients to connect as an anonymous user from the local host with any user name. In this case, if a client connects as user1 from the local host, [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) and [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) return different values:

mysql> SELECT USER(), CURRENT\_USER();

+-----------------+----------------+

| USER() | CURRENT\_USER() |

+-----------------+----------------+

| user1@localhost | @localhost |

+-----------------+----------------+

The host name part of an account can also contain wildcards. If the host name contains a '%' or '\_' pattern character or uses netmask notation, the account can be used for clients connecting from multiple hosts and the [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) value does not indicate which one. For example, the account 'user2'@'%.example.com' can be used by user2 to connect from any host in the example.com domain. If user2 connects from remote.example.com, [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) and [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) return different values:

mysql> SELECT USER(), CURRENT\_USER();

+--------------------------+---------------------+

| USER() | CURRENT\_USER() |

+--------------------------+---------------------+

| user2@remote.example.com | user2@%.example.com |

+--------------------------+---------------------+

If an application must invoke [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) for user auditing (for example, if it does auditing from within triggers) but must also be able to associate the [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) value with an account in the user table, it is necessary to avoid accounts that contain wildcards in the User or Host column. Specifically, do not permit User to be empty (which creates an anonymous-user account), and do not permit pattern characters or netmask notation in Host values. All accounts must have a nonempty User value and literal Host value.

With respect to the previous examples, the ''@'localhost' and 'user2'@'%.example.com' accounts should be changed not to use wildcards:

RENAME USER ''@'localhost' TO 'user1'@'localhost';

RENAME USER 'user2'@'%.example.com' TO 'user2'@'remote.example.com';

If user2 must be able to connect from several hosts in the example.com domain, there should be a separate account for each host.

To extract the user name or host name part from a [CURRENT\_USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_current-user) or [USER()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_user) value, use the [SUBSTRING\_INDEX()](https://dev.mysql.com/doc/refman/8.0/en/string-functions.html#function_substring-index) function:

mysql> SELECT SUBSTRING\_INDEX(CURRENT\_USER(),'@',1);

+---------------------------------------+

| SUBSTRING\_INDEX(CURRENT\_USER(),'@',1) |

+---------------------------------------+

| user1 |

+---------------------------------------+

mysql> SELECT SUBSTRING\_INDEX(CURRENT\_USER(),'@',-1);

+----------------------------------------+

| SUBSTRING\_INDEX(CURRENT\_USER(),'@',-1) |

+----------------------------------------+

| localhost |

+----------------------------------------+

## Using Encrypted Connections

With an unencrypted connection between the MySQL client and the server, someone with access to the network could watch all your traffic and inspect the data being sent or received between client and server.

When you must move information over a network in a secure fashion, an unencrypted connection is unacceptable. To make any kind of data unreadable, use encryption. Encryption algorithms must include security elements to resist many kinds of known attacks such as changing the order of encrypted messages or replaying data twice.

MySQL supports encrypted connections between clients and the server using the TLS (Transport Layer Security) protocol. TLS is sometimes referred to as SSL (Secure Sockets Layer) but MySQL does not actually use the SSL protocol for encrypted connections because its encryption is weak (see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html)).

TLS uses encryption algorithms to ensure that data received over a public network can be trusted. It has mechanisms to detect data change, loss, or replay. TLS also incorporates algorithms that provide identity verification using the X.509 standard.

X.509 makes it possible to identify someone on the Internet. In basic terms, there should be some entity called a “Certificate Authority” (or CA) that assigns electronic certificates to anyone who needs them. Certificates rely on asymmetric encryption algorithms that have two encryption keys (a public key and a secret key). A certificate owner can present the certificate to another party as proof of identity. A certificate consists of its owner's public key. Any data encrypted using this public key can be decrypted only using the corresponding secret key, which is held by the owner of the certificate.

Support for encrypted connections in MySQL is provided using OpenSSL. For information about the encryption protocols and ciphers that OpenSSL supports, see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html).

**Note**

From MySQL 8.0.11 to 8.0.17, it was possible to compile MySQL using wolfSSL as an alternative to OpenSSL. As of MySQL 8.0.18, support for wolfSSL is removed and all MySQL builds use OpenSSL.

By default, MySQL programs attempt to connect using encryption if the server supports encrypted connections, falling back to an unencrypted connection if an encrypted connection cannot be established. For information about options that affect use of encrypted connections, see [Section 6.3.1, “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html) and [Command Options for Encrypted Connections](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#encrypted-connection-options).

MySQL performs encryption on a per-connection basis, and use of encryption for a given user can be optional or mandatory. This enables you to choose an encrypted or unencrypted connection according to the requirements of individual applications. For information on how to require users to use encrypted connections, see the discussion of the REQUIRE clause of the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement in [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html). See also the description of the [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) system variable at [Section 5.1.8, “Server System Variables”](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html)

Encrypted connections can be used between source and replica servers. See [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html).

For information about using encrypted connections from the MySQL C API, see [C API Support for Encrypted Connections](https://dev.mysql.com/doc/c-api/8.0/en/c-api-encrypted-connections.html).

It is also possible to connect using encryption from within an SSH connection to the MySQL server host. For an example, see [Section 6.3.4, “Connecting to MySQL Remotely from Windows with SSH”](https://dev.mysql.com/doc/refman/8.0/en/windows-and-ssh.html).

### Configuring MySQL to Use Encrypted Connections

Several configuration parameters are available to indicate whether to use encrypted connections, and to specify the appropriate certificate and key files. This section provides general guidance about configuring the server and clients for encrypted connections:

Encrypted connections also can be used in these contexts:

* Between source and replica replication servers.
* Among Group Replication servers..
* By client programs that are based on the MySQL C API.

Instructions for creating any required certificate and key files are available in [Section , “Creating SSL and RSA Certificates and Keys”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files.html).

#### Server-Side Startup Configuration for Encrypted Connections

On the server side, the [--ssl](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_ssl) option specifies that the server permits but does not require encrypted connections. This option is enabled by default, so it need not be specified explicitly.

To require that clients connect using encrypted connections, enable the [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) system variable. See [Configuring Encrypted Connections as Mandatory](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html#mandatory-encrypted-connections).

These system variables on the server side specify the certificate and key files the server uses when permitting clients to establish encrypted connections:

* [ssl\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_ca): The path name of the Certificate Authority (CA) certificate file. ([ssl\_capath](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_capath) is similar but specifies the path name of a directory of CA certificate files.)
* [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert): The path name of the server public key certificate file. This certificate can be sent to the client and authenticated against the CA certificate that it has.
* [ssl\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_key): The path name of the server private key file.

For example, to enable the server for encrypted connections, start it with these lines in the my.cnf file, changing the file names as necessary:

[mysqld]

ssl\_ca=ca.pem

ssl\_cert=server-cert.pem

ssl\_key=server-key.pem

To specify in addition that clients are required to use encrypted connections, enable the [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) system variable:

[mysqld]

ssl\_ca=ca.pem

ssl\_cert=server-cert.pem

ssl\_key=server-key.pem

require\_secure\_transport=ON

Each certificate and key system variable names a file in PEM format. Should you need to create the required certificate and key files, see [Section 6.3.3, “Creating SSL and RSA Certificates and Keys”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files.html). MySQL servers compiled using OpenSSL can generate missing certificate and key files automatically at startup. See [Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html). Alternatively, if you have a MySQL source distribution, you can test your setup using the demonstration certificate and key files in its mysql-test/std\_data directory.

The server performs certificate and key file autodiscovery. If no explicit encrypted-connection options are given other than [--ssl](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_ssl) (possibly along with [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher)) to configure encrypted connections, the server attempts to enable encrypted-connection support automatically at startup:

* If the server discovers valid certificate and key files named ca.pem, server-cert.pem, and server-key.pem in the data directory, it enables support for encrypted connections by clients. (The files need not have been generated automatically; what matters is that they have those names and are valid.)
* If the server does not find valid certificate and key files in the data directory, it continues executing but without support for encrypted connections.

If the server automatically enables encrypted connection support, it writes a note to the error log. If the server discovers that the CA certificate is self-signed, it writes a warning to the error log. (The certificate is self-signed if created automatically by the server or manually using [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html).)

MySQL also provides these system variables for server-side encrypted-connection control:

* [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher): The list of permissible ciphers for connection encryption.
* [ssl\_crl](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_crl): The path name of the file containing certificate revocation lists. ([ssl\_crlpath](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_crlpath) is similar but specifies the path name of a directory of certificate revocation-list files.)
* [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version), [tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_ciphersuites): Which encryption protocols and ciphersuites the server permits for encrypted connections; see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html). For example, you can set [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) to prevent clients from using less-secure protocols.

If the server cannot create a valid TLS context from the system variables for server-side encrypted-connection control, the server does not support encrypted connections.

#### Server-Side Runtime Configuration and Monitoring for Encrypted Connections

Prior to MySQL 8.0.16, the tls\_***xxx*** and ssl\_***xxx*** system variables that configure encrypted-connection support can be set only at server startup. These system variables therefore determine the TLS context the server uses for all new connections.

As of MySQL 8.0.16, the tls\_***xxx*** and ssl\_***xxx*** system variables are dynamic and can be set at runtime, not just at startup. If changed with [SET GLOBAL](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html), the new values apply only until server restart. If changed with [SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html), the new values also carry over to subsequent server restarts. See [Section 13.7.6.1, “SET Syntax for Variable Assignment”](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html). However, runtime changes to these variables do not immediately affect the TLS context for new connections, as explained later in this section.

Along with the change in MySQL 8.0.16 that enables runtime changes to the TLS context-related system variables, the server enables runtime updates to the actual TLS context used for new connections. This capability may be useful, for example, to avoid restarting a MySQL server that has been running so long that its SSL certificate has expired.

To create the initial TLS context, the server uses the values that the context-related system variables have at startup. To expose the context values, the server also initializes a set of corresponding status variables. The following table shows the system variables that define the TLS context and the corresponding status variables that expose the currently active context values.

**Table 6.11 System and Status Variables for Server Main Connection Interface TLS Context**

| **System Variable Name** | **Corresponding Status Variable Name** |
| --- | --- |
| [ssl\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_ca) | [Current\_tls\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_ca) |
| [ssl\_capath](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_capath) | [Current\_tls\_capath](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_capath) |
| [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert) | [Current\_tls\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_cert) |
| [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher) | [Current\_tls\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_cipher) |
| [ssl\_crl](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_crl) | [Current\_tls\_crl](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_crl) |
| [ssl\_crlpath](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_crlpath) | [Current\_tls\_crlpath](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_crlpath) |
| [ssl\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_key) | [Current\_tls\_key](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_key) |
| [tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_ciphersuites) | [Current\_tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_ciphersuites) |
| [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) | [Current\_tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Current_tls_version) |

To reconfigure the TLS context at runtime, use this procedure:

1. For any TLS context-related system variables that should be changed, set them to their new values.
2. Execute [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls). This statement reconfigures the active TLS context from the current values of the TLS context-related system variables. It also sets the context-related status variables to reflect the new active context values. The statement requires the [CONNECTION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_connection-admin) privilege.
3. New connections established after execution of [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) use the new TLS context. Existing connections remain unaffected. If existing connections should be terminated, use the [KILL](https://dev.mysql.com/doc/refman/8.0/en/kill.html) statement.

The members of each pair of system and status variables may have different values temporarily due to the way the reconfiguration procedure works:

* Changes to the system variables prior to [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) do not change the TLS context. At this point, those changes have no effect on new connections, and corresponding context-related system and status variables may have different values. This enables you to make any changes required to individual system variables, then update the active TLS context atomically with [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) after all system variable changes have been made.
* After [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls), corresponding system and status variables have the same values. This remains true until the next change to the system variables.

In some cases, [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) by itself may suffice to reconfigure the TLS context, without changing any system variables. Suppose that the certificate in the file named by [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert) has expired. It is sufficient to replace the existing file contents with a nonexpired certificate and execute [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) to cause the new file contents to be read and used for new connections.

As of MySQL 8.0.21, the server implements independent connection-encryption configuration for the administrative connection interface. See [Administrative Interface Support for Encrypted Connections](https://dev.mysql.com/doc/refman/8.0/en/administrative-connection-interface.html#administrative-interface-encrypted-connections). In addition, [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) is extended with a FOR CHANNEL clause that enables specifying the channel (interface) for which to reload the TLS context. See [Section 13.1.5, “ALTER INSTANCE Statement”](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html). There are no status variables to expose the administrative interface TLS context, but the Performance Schema [tls\_channel\_status](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-tls-channel-status-table.html) table exposes TLS properties for both the main and administrative interfaces. See [Section 27.12.19.8, “The tls\_channel\_status Table”](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-tls-channel-status-table.html).

Updating the main interface TLS context has these effects:

* The update changes the TLS context used for new connections on the main connection interface.
* The update also changes the TLS context used for new connections on the administrative interface unless some nondefault TLS parameter value is configured for that interface.
* The update does not affect the TLS context used by other enabled server plugins or components such as Group Replication or X Plugin:
  + To apply the main interface reconfiguration to Group Replication's group communication connections, which take their settings from the server's TLS context-related system variables, you must execute [STOP GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/stop-group-replication.html) followed by [START GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html) to stop and restart Group Replication.
  + X Plugin initializes its TLS context at plugin initialization as described at [Section 20.5.3, “Using Encrypted Connections with X Plugin”](https://dev.mysql.com/doc/refman/8.0/en/x-plugin-encrypted-connections.html). This context does not change thereafter.

By default, the RELOAD TLS action rolls back with an error and has no effect if the configuration values do not permit creation of the new TLS context. The previous context values continue to be used for new connections. If the optional NO ROLLBACK ON ERROR clause is given and the new context cannot be created, rollback does not occur. Instead, a warning is generated and encryption is disabled for new connections on the interface to which the statement applies.

Options that enable or disable encrypted connections on a connection interface have an effect only at startup. For example, the [--ssl](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_ssl) and [--admin-ssl](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_admin-ssl) options affect only at startup whether the main and administrative interfaces support encrypted connections. Such options are ignored and have no effect on the operation of [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) at runtime. For example, you can use [--ssl=OFF](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_ssl) to start the server with encrypted connections disabled on the main interface, then reconfigure TLS and execute [ALTER INSTANCE RELOAD TLS](https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html#alter-instance-reload-tls) to enable encrypted connections at runtime.

#### Client-Side Configuration for Encrypted Connections

For a complete list of client options related to establishment of encrypted connections, see [Command Options for Encrypted Connections](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#encrypted-connection-options).

By default, MySQL client programs attempt to establish an encrypted connection if the server supports encrypted connections, with further control available through the [--ssl-mode](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option:

* In the absence of an [--ssl-mode](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option, clients attempt to connect using encryption, falling back to an unencrypted connection if an encrypted connection cannot be established. This is also the behavior with an explicit [--ssl-mode=PREFFERED](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option.
* With [--ssl-mode=REQUIRED](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode), clients require an encrypted connection and fail if one cannot be established.
* With [--ssl-mode=DISABLED](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode), clients use an unencrypted connection.
* With [--ssl-mode=VERIFY\_CA](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) or [--ssl-mode=VERIFY\_IDENTITY](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode), clients require an encrypted connection, and also perform verification against the server CA certificate and (with VERIFY\_IDENTITY) against the server host name in its certificate.

Attempts to establish an unencrypted connection fail if the [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) system variable is enabled on the server side to cause the server to require encrypted connections. See [Configuring Encrypted Connections as Mandatory](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html#mandatory-encrypted-connections).

The following options on the client side identify the certificate and key files clients use when establishing encrypted connections to the server. They are similar to the [ssl\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_ca), [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert), and [ssl\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_key) system variables used on the server side, but [--ssl-cert](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cert) and [--ssl-key](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-key) identify the client public and private key:

* [--ssl-ca](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-ca): The path name of the Certificate Authority (CA) certificate file. This option, if used, must specify the same certificate used by the server. (--ssl-capath is similar but specifies the path name of a directory of CA certificate files.)
* [--ssl-cert](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cert): The path name of the client public key certificate file.
* [--ssl-key](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-key): The path name of the client private key file.

For additional security relative to that provided by the default encryption, clients can supply a CA certificate matching the one used by the server and enable host name identity verification. In this way, the server and client place their trust in the same CA certificate and the client verifies that the host to which it connected is the one intended:

* To specify the CA certificate, use [--ssl-ca](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-ca) (or [--ssl-capath](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-capath)), and specify [--ssl-mode=VERIFY\_CA](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode).
* To enable host name identity verification as well, use [--ssl-mode=VERIFY\_IDENTITY](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) rather than [--ssl-mode=VERIFY\_CA](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode).

**Note**

Host name identity verification with VERIFY\_IDENTITY does not work with self-signed certificates that are created automatically by the server or manually using [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) (see [Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html)). Such self-signed certificates do not contain the server name as the Common Name value.

Host name identity verification also does not work with certificates that specify the Common Name using wildcards because that name is compared verbatim to the server name.

MySQL also provides these options for client-side SSL control:

* [--ssl-cipher](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cipher): The list of permissible ciphers for connection encryption.
* [--ssl-crl](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-crl): The path name of the file containing certificate revocation lists. (--ssl-crlpath is similar but specifies the path name of a directory of certificate revocation-list files.)
* [--tls-version](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-version), [--tls-ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-ciphersuites): The permitted encryption protocols and ciphersuites; see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html).

Depending on the encryption requirements of the MySQL account used by a client, the client may be required to specify certain options to connect using encryption to the MySQL server.

Suppose that you want to connect using an account that has no special encryption requirements or that was created using a [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement that included the REQUIRE SSL clause. Assuming that the server supports encrypted connections, a client can connect using encryption with no [--ssl-mode](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option or with an explicit [--ssl-mode=PREFFERED](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option:

mysql

Or:

mysql --ssl-mode=PREFERRED

For an account created with a REQUIRE SSL clause, the connection attempt fails if an encrypted connection cannot be established. For an account with no special encryption requirements, the attempt falls back to an unencrypted connection if an encrypted connection cannot be established. To prevent fallback and fail if an encrypted connection cannot be obtained, connect like this:

mysql --ssl-mode=REQUIRED

If the account has more stringent security requirements, other options must be specified to establish an encrypted connection:

* For accounts created with a REQUIRE X509 clause, clients must specify at least [--ssl-cert](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cert) and [--ssl-key](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-key). In addition, [--ssl-ca](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-ca) (or [--ssl-capath](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-capath)) is recommended so that the public certificate provided by the server can be verified. For example (enter the command on a single line):
* mysql --ssl-ca=ca.pem
* --ssl-cert=client-cert.pem

--ssl-key=client-key.pem

* For accounts created with a REQUIRE ISSUER or REQUIRE SUBJECT clause, the encryption requirements are the same as for REQUIRE X509, but the certificate must match the issue or subject, respectively, specified in the account definition.

For additional information about the REQUIRE clause, see [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html).

To prevent use of encryption and override other --ssl-***xxx*** options, invoke the client program with [--ssl-mode=DISABLED](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode):

mysql --ssl-mode=DISABLED

To determine whether the current connection with the server uses encryption, check the session value of the [Ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Ssl_cipher) status variable. If the value is empty, the connection is not encrypted. Otherwise, the connection is encrypted and the value indicates the encryption cipher. For example:

mysql> SHOW SESSION STATUS LIKE 'Ssl\_cipher';

+---------------+---------------------------+

| Variable\_name | Value |

+---------------+---------------------------+

| Ssl\_cipher | DHE-RSA-AES128-GCM-SHA256 |

+---------------+---------------------------+

For the [**mysql**](https://dev.mysql.com/doc/refman/8.0/en/mysql.html) client, an alternative is to use the STATUS or \s command and check the SSL line:

mysql> \s

...

SSL: Not in use

...

Or:

mysql> \s

...

SSL: Cipher in use is DHE-RSA-AES128-GCM-SHA256

...

#### Configuring Encrypted Connections as Mandatory

For some MySQL deployments it may be not only desirable but mandatory to use encrypted connections (for example, to satisfy regulatory requirements). This section discusses configuration settings that enable you to do this. These levels of control are available:

* You can configure the server to require that clients connect using encrypted connections.
* You can invoke individual client programs to require an encrypted connection, even if the server permits but does not require encryption.
* You can configure individual MySQL accounts to be usable only over encrypted connections.

To require that clients connect using encrypted connections, enable the [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) system variable. For example, put these lines in the server my.cnf file:

[mysqld]

require\_secure\_transport=ON

Alternatively, to set and persist the value at runtime, use this statement:

SET PERSIST require\_secure\_transport=ON;

[SET PERSIST](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) sets the value for the running MySQL instance. It also saves the value, causing it to be used for subsequent server restarts..

With [require\_secure\_transport](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_require_secure_transport) enabled, client connections to the server are required to use some form of secure transport, and the server permits only TCP/IP connections that use SSL, or connections that use a socket file (on Unix) or shared memory (on Windows). The server rejects nonsecure connection attempts, which fail with an [ER\_SECURE\_TRANSPORT\_REQUIRED](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_secure_transport_required) error.

To invoke a client program such that it requires an encrypted connection whether or not the server requires encryption, use an [--ssl-mode](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-mode) option value of REQUIRED, VERIFY\_CA, or VERIFY\_IDENTITY. For example:

mysql --ssl-mode=REQUIRED

mysqldump --ssl-mode=VERIFY\_CA

mysqladmin --ssl-mode=VERIFY\_IDENTITY

To configure a MySQL account to be usable only over encrypted connections, include a REQUIRE clause in the [CREATE USER](https://dev.mysql.com/doc/refman/8.0/en/create-user.html) statement that creates the account, specifying in that clause the encryption characteristics you require. For example, to require an encrypted connection and the use of a valid X.509 certificate, use REQUIRE X509:

CREATE USER 'jeffrey'@'localhost' REQUIRE X509;

For additional information about the REQUIRE clause, see [Section 13.7.1.3, “CREATE USER Statement”](https://dev.mysql.com/doc/refman/8.0/en/create-user.html).

To modify existing accounts that have no encryption requirements, use the [ALTER USER](https://dev.mysql.com/doc/refman/8.0/en/alter-user.html) statement.

### Encrypted Connection TLS Protocols and Ciphers

MySQL supports multiple TLS protocols and ciphers, and enables configuring which protocols and ciphers to permit for encrypted connections. It is also possible to determine which protocol and cipher the current session uses.

#### Supported Connection TLS Protocols

MySQL supports encrypted connections using the TLSv1, TLSv1.1, TLSv1.2, and TLSv1.3 protocols, listed in order from less secure to more secure. The set of protocols actually permitted for connections is subject to multiple factors:

* MySQL configuration. Permitted TLS protocols can be configured on both the server side and client side to include only a subset of the supported TLS protocols. The configuration on both sides must include at least one protocol in common or connection attempts cannot negotiate a protocol to use. For details, see [Connection TLS Protocol Negotiation](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html#encrypted-connection-protocol-negotiation).
* System-wide host configuration. The host system may permit only certain TLS protocols, which means that MySQL connections cannot use nonpermitted protocols even if MySQL itself permits them:
  + Suppose that MySQL configuration permits TLSv1, TLSv1.1, and TLSv1.2, but your host system configuration permits only connections that use TLSv1.2 or higher. In this case, you cannot establish MySQL connections that use TLSv1 or TLSv1.1, even though MySQL is configured to permit them, because the host system does not permit them.
  + If MySQL configuration permits TLSv1, TLSv1.1, and TLSv1.2, but your host system configuration permits only connections that use TLSv1.3 or higher, you cannot establish MySQL connections at all, because no protocol permitted by MySQL is permitted by the host system.

Workarounds for this issue include:

* + Change the system-wide host configuration to permit additional TLS protocols. Consult your operating system documentation for instructions. For example, your system may have an /etc/ssl/openssl.cnf file that contains these lines to restrict TLS protocols to TLSv1.2 or higher:
  + [system\_default\_sect]

MinProtocol = TLSv1.2

Changing the value to a lower protocol version or None makes the system more permissive. This workaround has the disadvantage that permitting lower (less secure) protocols may have adverse security consequences.

* + If you cannot or prefer not to change the host system TLS configuration, change MySQL applications to use higher (more secure) TLS protocols that are permitted by the host system. This may not be possible for older versions of MySQL that support only lower protocol versions. For example, TLSv1 is the only supported protocol prior to MySQL 5.6.46, so attempts to connect to a pre-5.6.46 server fail even if the client is from a newer MySQL version that supports higher protocol versions. In such cases, an upgrade to a version of MySQL that supports additional TLS versions may be required.
* The SSL library. If the SSL library does not support a particular protocol, neither does MySQL, and any parts of the following discussion that specify that protocol do not apply.

**Note**

Support for the TLSv1.3 protocol is available as of MySQL 8.0.16 (as of MySQL 8.0.18 for the Group Replication component). In addition, to use TLSv1.3, both the MySQL server and the client application must be compiled using OpenSSL 1.1.1 or higher.

#### Connection TLS Protocol Configuration

On the server side, the value of the [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) system variable determines which TLS protocols a MySQL server permits for encrypted connections. The [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) value applies to connections from clients, regular source/replica replication connections where this server instance is the source, Group Replication group communication connections, and Group Replication distributed recovery connections where this server instance is the donor. The variable value is a list of one or more comma-separated protocol versions from this list (not case-sensitive): TLSv1, TLSv1.1, TLSv1.2, and (if available) TLSV1.3. By default, this variable lists all protocols supported by the SSL library used to compile MySQL. To determine the value of [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) at runtime, use this statement:

mysql> SHOW GLOBAL VARIABLES LIKE 'tls\_version';

+---------------+-----------------------+

| Variable\_name | Value |

+---------------+-----------------------+

| tls\_version | TLSv1,TLSv1.1,TLSv1.2 |

+---------------+-----------------------+

To change the value of [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version), set it at server startup. For example, to permit connections that use the TLSv1.1 or TLSv1.2 protocol, but prohibit connections that use the less-secure TLSv1 protocol, use these lines in the server my.cnf file:

[mysqld]

tls\_version=TLSv1.1,TLSv1.2

To be even more restrictive and permit only TLSv1.2 connections, set [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) like this:

[mysqld]

tls\_version=TLSv1.2

As of MySQL 8.0.16, tls\_version can also be changed at runtime. See [Server-Side Runtime Configuration and Monitoring for Encrypted Connections](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html#using-encrypted-connections-server-side-runtime-configuration).

On the client side, the [--tls-version](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-version) option specifies which TLS protocols a client program permits for connections to the server. The format of the option value is the same as for the [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) system variable described previously (a list of one or more comma-separated protocol versions).

For source/replica replication connections where this server instance is the replica, the MASTER\_TLS\_VERSION option for the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement specifies which TLS protocols the replica permits for connections to the source. The format of the option value is the same as for the [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) system variable described previously. See [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html).

The protocols that can be specified for MASTER\_TLS\_VERSION depend on the SSL library. This option is independent of and not affected by the server [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) value. For example, a server that acts as a replica can be configured with [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) set to TLSv1.3 to permit only incoming connections that use TLSv1.3, but also configured with MASTER\_TLS\_VERSION set to TLSv1.2 to permit only TLSv1.2 for outgoing replica connections to the source.

For Group Replication distributed recovery connections where this server instance is the joining member that initiates distributed recovery (that is, the client), the [group\_replication\_recovery\_tls\_version](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_recovery_tls_version) system variable specifies which protocols are permitted by the client. This option is independent of and not affected by the server [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) value, which applies when this server instance is the donor. A Group Replication server generally participates in distributed recovery both as a donor and as a joining member over the course of its group membership, so both these system variables should be set. See [Section 18.5.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](https://dev.mysql.com/doc/refman/8.0/en/group-replication-secure-socket-layer-support-ssl.html).

TLS protocol configuration affects which protocol a given connection uses, as described in [Connection TLS Protocol Negotiation](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html#encrypted-connection-protocol-negotiation).

Permitted protocols should be chosen such as not to leave “holes” in the list. For example, these server configuration values do not have holes:

tls\_version=TLSv1,TLSv1.1,TLSv1.2,TLSv1.3

tls\_version=TLSv1.1,TLSv1.2,TLSv1.3

tls\_version=TLSv1.2,TLSv1.3

tls\_version=TLSv1.3

These values do have holes and should not be used:

tls\_version=TLSv1,TLSv1.2 *(*TLSv1*.*1 is missing*)*

tls\_version=TLSv1.1,TLSv1.3 *(*TLSv1*.*2 is missing*)*

The prohibition on holes also applies in other configuration contexts, such as for clients or replicas.

The list of permitted protocols should not be empty. If you set a TLS version parameter to the empty string, encrypted connections cannot be established:

* [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version): The server does not permit encrypted incoming connections.
* [--tls-version](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-version): The client does not permit encrypted outgoing connections to the server.
* MASTER\_TLS\_VERSION: The replica does not permit encrypted outgoing connections to the source.

#### Connection Cipher Configuration

A default set of ciphers applies to encrypted connections, which can be overridden by explicitly configuring the permitted ciphers. During connection establishment, both sides of a connection must permit some cipher in common or the connection fails. Of the permitted ciphers common to both sides, the SSL library chooses the one supported by the provided certificate that has the highest priority.

To specify a cipher or ciphers applicable for encrypted connections that use TLS protocols up through TLSv1.2:

* Set the [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher) system variable on the server side, and use the [--ssl-cipher](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cipher) option for client programs.
* For regular source/replica replication connections, where this server instance is the source, set the [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher) system variable. Where this server instance is the replica, use the MASTER\_SSL\_CIPHER option for the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement. See [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html).
* For a Group Replication group member, for Group Replication group communication connections and also for Group Replication distributed recovery connections where this server instance is the donor, set the [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher) system variable. For Group Replication distributed recovery connections where this server instance is the joining member, use the [group\_replication\_recovery\_ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_recovery_ssl_cipher) system variable..

For encrypted connections that use TLSv1.3, OpenSSL 1.1.1 and higher supports the following ciphersuites, the first three of which are enabled by default:

TLS\_AES\_128\_GCM\_SHA256

TLS\_AES\_256\_GCM\_SHA384

TLS\_CHACHA20\_POLY1305\_SHA256

TLS\_AES\_128\_CCM\_SHA256

TLS\_AES\_128\_CCM\_8\_SHA256

To configure the permitted TLSv1.3 ciphersuites explicitly, set the following parameters. In each case, the configuration value is a list of zero or more colon-separated ciphersuite names.

* On the server side, use the [tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_ciphersuites) system variable. If this variable is not set, its default value is NULL, which means that the server permits the default set of ciphersuites. If the variable is set to the empty string, no ciphersuites are enabled and encrypted connections cannot be established.
* On the client side, use the [--tls-ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-ciphersuites) option. If this option is not set, the client permits the default set of ciphersuites. If the option is set to the empty string, no ciphersuites are enabled and encrypted connections cannot be established.
* For regular source/replica replication connections, where this server instance is the source, use the [tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_ciphersuites) system variable. Where this server instance is the replica, use the MASTER\_TLS\_CIPHERSUITES option for the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement. See [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html).
* For a Group Replication group member, for Group Replication group communication connections and also for Group Replication distributed recovery connections where this server instance is the donor, use the [tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_ciphersuites) system variable. For Group Replication distributed recovery connections where this server instance is the joining member, use the [group\_replication\_recovery\_tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_recovery_tls_ciphersuites) system variable. See [Section 18.5.2, “Securing Group Communication Connections with Secure Socket Layer (SSL)”](https://dev.mysql.com/doc/refman/8.0/en/group-replication-secure-socket-layer-support-ssl.html).

**Note**

Ciphersuite support is available as of MySQL 8.0.16, but requires that both the MySQL server and the client application be compiled using OpenSSL 1.1.1 or higher.

In MySQL 8.0.16 through 8.0.18, the MASTER\_TLS\_CIPHERSUITES option for the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement and the [group\_replication\_recovery\_tls\_ciphersuites](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_recovery_tls_ciphersuites) system variable are not available. In these releases, if TLSv1.3 is used for source/replica replication connections, or in Group Replication for distributed recovery (supported from MySQL 8.0.18), the replication source or Group Replication donor servers must permit the use of at least one TLSv1.3 ciphersuite that is enabled by default. From MySQL 8.0.19, you can use the options to configure client support for any selection of ciphersuites, including only non-default ciphersuites if you want.

A given cipher may work only with particular TLS protocols, which affects the TLS protocol negotiation process. See [Connection TLS Protocol Negotiation](https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html#encrypted-connection-protocol-negotiation).

To determine which ciphers a given server supports, check the session value of the [Ssl\_cipher\_list](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Ssl_cipher_list) status variable:

SHOW SESSION STATUS LIKE 'Ssl\_cipher\_list';

The [Ssl\_cipher\_list](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Ssl_cipher_list) status variable lists the possible SSL ciphers (empty for non-SSL connections). If MySQL supports TLSv1.3, the value includes the possible TLSv1.3 ciphersuites.

For encrypted connections that use TLS.v1.3, MySQL uses the SSL library default ciphersuite list.

For encrypted connections that use TLS protocols up through TLSv1.2, MySQL passes the following default cipher list to the SSL library.

ECDHE-ECDSA-AES128-GCM-SHA256

ECDHE-ECDSA-AES256-GCM-SHA384

ECDHE-RSA-AES128-GCM-SHA256

ECDHE-RSA-AES256-GCM-SHA384

ECDHE-ECDSA-AES128-SHA256

ECDHE-RSA-AES128-SHA256

ECDHE-ECDSA-AES256-SHA384

ECDHE-RSA-AES256-SHA384

DHE-RSA-AES128-GCM-SHA256

DHE-DSS-AES128-GCM-SHA256

DHE-RSA-AES128-SHA256

DHE-DSS-AES128-SHA256

DHE-DSS-AES256-GCM-SHA384

DHE-RSA-AES256-SHA256

DHE-DSS-AES256-SHA256

ECDHE-RSA-AES128-SHA

ECDHE-ECDSA-AES128-SHA

ECDHE-RSA-AES256-SHA

ECDHE-ECDSA-AES256-SHA

DHE-DSS-AES128-SHA

DHE-RSA-AES128-SHA

TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA

DHE-RSA-AES256-SHA

AES128-GCM-SHA256

DH-DSS-AES128-GCM-SHA256

ECDH-ECDSA-AES128-GCM-SHA256

AES256-GCM-SHA384

DH-DSS-AES256-GCM-SHA384

ECDH-ECDSA-AES256-GCM-SHA384

AES128-SHA256

DH-DSS-AES128-SHA256

ECDH-ECDSA-AES128-SHA256

AES256-SHA256

DH-DSS-AES256-SHA256

ECDH-ECDSA-AES256-SHA384

AES128-SHA

DH-DSS-AES128-SHA

ECDH-ECDSA-AES128-SHA

AES256-SHA

DH-DSS-AES256-SHA

ECDH-ECDSA-AES256-SHA

DHE-RSA-AES256-GCM-SHA384

DH-RSA-AES128-GCM-SHA256

ECDH-RSA-AES128-GCM-SHA256

DH-RSA-AES256-GCM-SHA384

ECDH-RSA-AES256-GCM-SHA384

DH-RSA-AES128-SHA256

ECDH-RSA-AES128-SHA256

DH-RSA-AES256-SHA256

ECDH-RSA-AES256-SHA384

ECDHE-RSA-AES128-SHA

ECDHE-ECDSA-AES128-SHA

ECDHE-RSA-AES256-SHA

ECDHE-ECDSA-AES256-SHA

DHE-DSS-AES128-SHA

DHE-RSA-AES128-SHA

TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA

DHE-RSA-AES256-SHA

AES128-SHA

DH-DSS-AES128-SHA

ECDH-ECDSA-AES128-SHA

AES256-SHA

DH-DSS-AES256-SHA

ECDH-ECDSA-AES256-SHA

DH-RSA-AES128-SHA

ECDH-RSA-AES128-SHA

DH-RSA-AES256-SHA

ECDH-RSA-AES256-SHA

DES-CBC3-SHA

These cipher restrictions are in place:

* The following ciphers are permanently restricted:
* !DHE-DSS-DES-CBC3-SHA
* !DHE-RSA-DES-CBC3-SHA
* !ECDH-RSA-DES-CBC3-SHA
* !ECDH-ECDSA-DES-CBC3-SHA
* !ECDHE-RSA-DES-CBC3-SHA

!ECDHE-ECDSA-DES-CBC3-SHA

* The following categories of ciphers are permanently restricted:
* !aNULL
* !eNULL
* !EXPORT
* !LOW
* !MD5
* !DES
* !RC2
* !RC4
* !PSK

!SSLv3

If the server is started with the [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert) system variable set to a certificate that uses any of the preceding restricted ciphers or cipher categories, the server starts with support for encrypted connections disabled.

#### Connection TLS Protocol Negotiation

Connection attempts in MySQL negotiate use of the highest TLS protocol version available on both sides for which a protocol-compatible encryption cipher is available on both sides. The negotiation process depends on factors such as the SSL library used to compile the server and client, the TLS protocol and encryption cipher configuration, and which key size is used:

* For a connection attempt to succeed, the server and client TLS protocol configuration must permit some protocol in common.
* Similarly, the server and client encryption cipher configuration must permit some cipher in common. A given cipher may work only with particular TLS protocols, so a protocol available to the negotiation process is not chosen unless there is also a compatible cipher.
* If TLSv1.3 is available, it is used if possible. (This means that server and client configuration both must permit TLSv1.3, and both must also permit some TLSv1.3-compatible encryption cipher.) Otherwise, MySQL continues through the list of available protocols, using TLSv1.2 if possible, and so forth. Negotiation proceeds from more secure protocols to less secure. Negotiation order is independent of the order in which protocols are configured. For example, negotiation order is the same regardless of whether [tls\_version](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) has a value of TLSv1,TLSv1.1,TLSv1.2,TLSv1.3 or TLSv1.3,TLSv1.2,TLSv1.1,TLSv1.
* TLSv1.2 does not work with all ciphers that have a key size of 512 bits or less. To use this protocol with such a key, set the [ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cipher) system variable on the server side or use the [--ssl-cipher](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cipher) client option to specify the cipher name explicitly:
* AES128-SHA
* AES128-SHA256
* AES256-SHA
* AES256-SHA256
* CAMELLIA128-SHA
* CAMELLIA256-SHA
* DES-CBC3-SHA
* DHE-RSA-AES256-SHA
* RC4-MD5
* RC4-SHA

SEED-SHA

* For better security, use a certificate with an RSA key size of at least 2048 bits.

If the server and client do not have a permitted protocol in common, and a protocol-compatible cipher in common, the server terminates the connection request. Examples:

* If the server is configured with [tls\_version=TLSv1.1,TLSv1.2](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version):
  + Connection attempts fail for clients invoked with [--tls-version=TLSv1](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-version), and for older clients that support only TLSv1.
  + Similarly, connection attempts fail for replicas configured with MASTER\_TLS\_VERSION = 'TLSv1', and for older replicas that support only TLSv1.
* If the server is configured with [tls\_version=TLSv1](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tls_version) or is an older server that supports only TLSv1:
  + Connection attempts fail for clients invoked with [--tls-version=TLSv1.1,TLSv1.2](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_tls-version).
  + Similarly, connection attempts fail for replicas configured with MASTER\_TLS\_VERSION = 'TLSv1.1,TLSv1.2'.

MySQL permits specifying a list of protocols to support. This list is passed directly down to the underlying SSL library and is ultimately up to that library what protocols it actually enables from the supplied list. Please refer to the MySQL source code and the OpenSSL [SSL\_CTX\_new()](https://www.openssl.org/docs/man1.1.0/ssl/SSL_CTX_new.html) documentation for information about how the SSL library handles this.

#### Monitoring Current Client Session TLS Protocol and Cipher

To determine which encryption TLS protocol and cipher the current client session uses, check the session values of the [Ssl\_version](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Ssl_version) and [Ssl\_cipher](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html#statvar_Ssl_cipher) status variables:

mysql> SELECT \* FROM performance\_schema.session\_status

WHERE VARIABLE\_NAME IN ('Ssl\_version','Ssl\_cipher');

+---------------+---------------------------+

| VARIABLE\_NAME | VARIABLE\_VALUE |

+---------------+---------------------------+

| Ssl\_cipher | DHE-RSA-AES128-GCM-SHA256 |

| Ssl\_version | TLSv1.2 |

+---------------+---------------------------+

If the connection is not encrypted, both variables have an empty value.

### Creating SSL and RSA Certificates and Keys

The following discussion describes how to create the files required for SSL and RSA support in MySQL. File creation can be performed using facilities provided by MySQL itself, or by invoking the **openssl** command directly.

SSL certificate and key files enable MySQL to support encrypted connections using SSL. See [Section  “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html).

RSA key files enable MySQL to support secure password exchange over unencrypted connections for accounts authenticated by the sha256\_password or caching\_sha2\_password plugin. See [Section  “SHA-256 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/sha256-pluggable-authentication.html), and [Section , “Caching SHA-2 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/caching-sha2-pluggable-authentication.html).

#### Creating SSL and RSA Certificates and Keys using MySQL

MySQL provides these ways to create the SSL certificate and key files and RSA key-pair files required to support encrypted connections using SSL and secure password exchange using RSA over unencrypted connections, if those files are missing:

* The server can autogenerate these files at startup, for MySQL distributions.
* Users can invoke the [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) utility manually.
* For some distribution types, such as RPM and DEB packages, [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) invocation occurs during data directory initialization. In this case, the MySQL distribution need not have been compiled using OpenSSL as long as the **openssl** command is available.

**Important**

Server autogeneration and [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) help lower the barrier to using SSL by making it easier to generate the required files. However, certificates generated by these methods are self-signed, which may not be very secure. After you gain experience using such files, consider obtaining certificate/key material from a registered certificate authority.

##### Automatic SSL and RSA File Generation

For MySQL distributions compiled using OpenSSL, the MySQL server has the capability of automatically generating missing SSL and RSA files at startup. The [auto\_generate\_certs](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_auto_generate_certs), [sha256\_password\_auto\_generate\_rsa\_keys](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sha256_password_auto_generate_rsa_keys), and [caching\_sha2\_password\_auto\_generate\_rsa\_keys](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_caching_sha2_password_auto_generate_rsa_keys) system variables control automatic generation of these files. These variables are enabled by default. They can be enabled at startup and inspected but not set at runtime.

At startup, the server automatically generates server-side and client-side SSL certificate and key files in the data directory if the [auto\_generate\_certs](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_auto_generate_certs) system variable is enabled, no SSL options other than [--ssl](https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_ssl) are specified, and the server-side SSL files are missing from the data directory. These files enable encrypted client connections using SSL; see [Section  “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html).

1. The server checks the data directory for SSL files with the following names:
2. ca.pem
3. server-cert.pem

server-key.pem

1. If any of those files are present, the server creates no SSL files. Otherwise, it creates them, plus some additional files:
2. ca.pem Self-signed CA certificate
3. ca-key.pem CA private key
4. server-cert.pem Server certificate
5. server-key.pem Server private key
6. client-cert.pem Client certificate

client-key.pem Client private key

1. If the server autogenerates SSL files, it uses the names of the ca.pem, server-cert.pem, and server-key.pem files to set the corresponding system variables ([ssl\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_ca), [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert), [ssl\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_key)).

At startup, the server automatically generates RSA private/public key-pair files in the data directory if all of these conditions are true: The [sha256\_password\_auto\_generate\_rsa\_keys](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sha256_password_auto_generate_rsa_keys) or [caching\_sha2\_password\_auto\_generate\_rsa\_keys](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_caching_sha2_password_auto_generate_rsa_keys) system variable is enabled; no RSA options are specified; the RSA files are missing from the data directory. These key-pair files enable secure password exchange using RSA over unencrypted connections for accounts authenticated by the sha256\_password or caching\_sha2\_password plugin; see [Section  “SHA-256 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/sha256-pluggable-authentication.html), and [Section  “Caching SHA-2 Pluggable Authentication”](https://dev.mysql.com/doc/refman/8.0/en/caching-sha2-pluggable-authentication.html).

1. The server checks the data directory for RSA files with the following names:
2. private\_key.pem Private member of private/public key pair

public\_key.pem Public member of private/public key pair

1. If any of these files are present, the server creates no RSA files. Otherwise, it creates them.
2. If the server autogenerates the RSA files, it uses their names to set the corresponding system variables ([sha256\_password\_private\_key\_path](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sha256_password_private_key_path) and [sha256\_password\_public\_key\_path](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sha256_password_public_key_path); [caching\_sha2\_password\_private\_key\_path](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_caching_sha2_password_private_key_path) and [caching\_sha2\_password\_public\_key\_path](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_caching_sha2_password_public_key_path)).

##### Manual SSL and RSA File Generation Using mysql\_ssl\_rsa\_setup

MySQL distributions include a [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) utility that can be invoked manually to generate SSL and RSA files. This utility is included with all MySQL distributions, but it does require that the **openssl** command be available. For usage instructions, see [Section 4.4.3, “**mysql\_ssl\_rsa\_setup** — Create SSL/RSA Files”](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html).

##### SSL and RSA File Characteristics

SSL and RSA files created automatically by the server or by invoking [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) have these characteristics:

* SSL and RSA keys are have a size of 2048 bits.
* The SSL CA certificate is self signed.
* The SSL server and client certificates are signed with the CA certificate and key, using the sha256WithRSAEncryption signature algorithm.
* SSL certificates use these Common Name (CN) values, with the appropriate certificate type (CA, Server, Client):
* ca.pem: MySQL\_Server\_suffix\_Auto\_Generated\_CA\_Certificate
* server-cert.pm: MySQL\_Server\_suffix\_Auto\_Generated\_Server\_Certificate

client-cert.pm: MySQL\_Server\_suffix\_Auto\_Generated\_Client\_Certificate

The ***suffix*** value is based on the MySQL version number. For files generated by [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html), the suffix can be specified explicitly using the [--suffix](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html#option_mysql_ssl_rsa_setup_suffix) option.

For files generated by the server, if the resulting CN values exceed 64 characters, the \_***suffix*** portion of the name is omitted.

* SSL files have blank values for Country (C), State or Province (ST), Organization (O), Organization Unit Name (OU) and email address.
* SSL files created by the server or by [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) are valid for ten years from the time of generation.
* RSA files do not expire.
* SSL files have different serial numbers for each certificate/key pair (1 for CA, 2 for Server, 3 for Client).
* Files created automatically by the server are owned by the account that runs the server. Files created using [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) are owned by the user who invoked that program. This can be changed on systems that support the chown() system call if the program is invoked by root and the [--uid](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html#option_mysql_ssl_rsa_setup_uid) option is given to specify the user who should own the files.
* On Unix and Unix-like systems, the file access mode is 644 for certificate files (that is, world readable) and 600 for key files (that is, accessible only by the account that runs the server).

To see the contents of an SSL certificate (for example, to check the range of dates over which it is valid), invoke **openssl** directly:

openssl x509 -text -in ca.pem

openssl x509 -text -in server-cert.pem

openssl x509 -text -in client-cert.pem

It is also possible to check SSL certificate expiration information using this SQL statement:

mysql> SHOW STATUS LIKE 'Ssl\_server\_not%';

+-----------------------+--------------------------+

| Variable\_name | Value |

+-----------------------+--------------------------+

| Ssl\_server\_not\_after | Apr 28 14:16:39 2027 GMT |

| Ssl\_server\_not\_before | May 1 14:16:39 2017 GMT |

+-----------------------+--------------------------+

#### Creating SSL Certificates and Keys Using openssl

This section describes how to use the **openssl** command to set up SSL certificate and key files for use by MySQL servers and clients. The first example shows a simplified procedure such as you might use from the command line. The second shows a script that contains more detail. The first two examples are intended for use on Unix and both use the **openssl** command that is part of OpenSSL. The third example describes how to set up SSL files on Windows.

**Note**

There are easier alternatives to generating the files required for SSL than the procedure described here: Let the server autogenerate them or use the [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) program. See [Section , “Creating SSL and RSA Certificates and Keys using MySQL”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html).

**Important**

Whatever method you use to generate the certificate and key files, the Common Name value used for the server and client certificates/keys must each differ from the Common Name value used for the CA certificate. Otherwise, the certificate and key files do not work for servers compiled using OpenSSL. A typical error in this case is:

ERROR 2026 (HY000): SSL connection error:

error:00000001:lib(0):func(0):reason(1)

##### Example 1: Creating SSL Files from the Command Line on Unix

The following example shows a set of commands to create MySQL server and client certificate and key files. You must respond to several prompts by the **openssl** commands. To generate test files, you can press Enter to all prompts. To generate files for production use, you should provide nonempty responses.

# Create clean environment

rm -rf newcerts

mkdir newcerts && cd newcerts

# Create CA certificate

openssl genrsa 2048 > ca-key.pem

openssl req -new -x509 -nodes -days 3600 \

-key ca-key.pem -out ca.pem

# Create server certificate, remove passphrase, and sign it

# server-cert.pem = public key, server-key.pem = private key

openssl req -newkey rsa:2048 -days 3600 \

-nodes -keyout server-key.pem -out server-req.pem

openssl rsa -in server-key.pem -out server-key.pem

openssl x509 -req -in server-req.pem -days 3600 \

-CA ca.pem -CAkey ca-key.pem -set\_serial 01 -out server-cert.pem

# Create client certificate, remove passphrase, and sign it

# client-cert.pem = public key, client-key.pem = private key

openssl req -newkey rsa:2048 -days 3600 \

-nodes -keyout client-key.pem -out client-req.pem

openssl rsa -in client-key.pem -out client-key.pem

openssl x509 -req -in client-req.pem -days 3600 \

-CA ca.pem -CAkey ca-key.pem -set\_serial 01 -out client-cert.pem

After generating the certificates, verify them:

openssl verify -CAfile ca.pem server-cert.pem client-cert.pem

You should see a response like this:

server-cert.pem: OK

client-cert.pem: OK

To see the contents of a certificate (for example, to check the range of dates over which a certificate is valid), invoke **openssl** like this:

openssl x509 -text -in ca.pem

openssl x509 -text -in server-cert.pem

openssl x509 -text -in client-cert.pem

Now you have a set of files that can be used as follows:

* ca.pem: Use this to set the [ssl\_ca](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_ca) system variable on the server side and the [--ssl-ca](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-ca) option on the client side. (The CA certificate, if used, must be the same on both sides.)
* server-cert.pem, server-key.pem: Use these to set the [ssl\_cert](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_cert) and [ssl\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_ssl_key) system variables on the server side.
* client-cert.pem, client-key.pem: Use these as the arguments to the [--ssl-cert](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-cert) and [--ssl-key](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_ssl-key) options on the client side.

For additional usage instructions, see [Section  “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html).

##### Example 2: Creating SSL Files Using a Script on Unix

Here is an example script that shows how to set up SSL certificate and key files for MySQL. After executing the script, use the files for SSL connections as described in [Section  “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html).

DIR=`pwd`/openssl

PRIV=$DIR/private

mkdir $DIR $PRIV $DIR/newcerts

cp /usr/share/ssl/openssl.cnf $DIR

replace ./demoCA $DIR -- $DIR/openssl.cnf

# Create necessary files: $database, $serial and $new\_certs\_dir

# directory (optional)

touch $DIR/index.txt

echo "01" > $DIR/serial

#

# Generation of Certificate Authority(CA)

#

openssl req -new -x509 -keyout $PRIV/cakey.pem -out $DIR/ca.pem \

-days 3600 -config $DIR/openssl.cnf

# Sample output:

# Using configuration from /home/jones/openssl/openssl.cnf

# Generating a 1024 bit RSA private key

# ................++++++

# .........++++++

# writing new private key to '/home/jones/openssl/private/cakey.pem'

# Enter PEM pass phrase:

# Verifying password - Enter PEM pass phrase:

# -----

# You are about to be asked to enter information to be

# incorporated into your certificate request.

# What you are about to enter is what is called a Distinguished Name

# or a DN.

# There are quite a few fields but you can leave some blank

# For some fields there will be a default value,

# If you enter '.', the field will be left blank.

# -----

# Country Name (2 letter code) [AU]:FI

# State or Province Name (full name) [Some-State]:.

# Locality Name (eg, city) []:

# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB

# Organizational Unit Name (eg, section) []:

# Common Name (eg, YOUR name) []:MySQL admin

# Email Address []:

#

# Create server request and key

#

openssl req -new -keyout $DIR/server-key.pem -out \

$DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf

# Sample output:

# Using configuration from /home/jones/openssl/openssl.cnf

# Generating a 1024 bit RSA private key

# ..++++++

# ..........++++++

# writing new private key to '/home/jones/openssl/server-key.pem'

# Enter PEM pass phrase:

# Verifying password - Enter PEM pass phrase:

# -----

# You are about to be asked to enter information that will be

# incorporated into your certificate request.

# What you are about to enter is what is called a Distinguished Name

# or a DN.

# There are quite a few fields but you can leave some blank

# For some fields there will be a default value,

# If you enter '.', the field will be left blank.

# -----

# Country Name (2 letter code) [AU]:FI

# State or Province Name (full name) [Some-State]:.

# Locality Name (eg, city) []:

# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB

# Organizational Unit Name (eg, section) []:

# Common Name (eg, YOUR name) []:MySQL server

# Email Address []:

#

# Please enter the following 'extra' attributes

# to be sent with your certificate request

# A challenge password []:

# An optional company name []:

#

# Remove the passphrase from the key

#

openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem

#

# Sign server cert

#

openssl ca -cert $DIR/ca.pem -policy policy\_anything \

-out $DIR/server-cert.pem -config $DIR/openssl.cnf \

-infiles $DIR/server-req.pem

# Sample output:

# Using configuration from /home/jones/openssl/openssl.cnf

# Enter PEM pass phrase:

# Check that the request matches the signature

# Signature ok

# The Subjects Distinguished Name is as follows

# countryName :PRINTABLE:'FI'

# organizationName :PRINTABLE:'MySQL AB'

# commonName :PRINTABLE:'MySQL admin'

# Certificate is to be certified until Sep 13 14:22:46 2003 GMT

# (365 days)

# Sign the certificate? [y/n]:y

#

#

# 1 out of 1 certificate requests certified, commit? [y/n]y

# Write out database with 1 new entries

# Data Base Updated

#

# Create client request and key

#

openssl req -new -keyout $DIR/client-key.pem -out \

$DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf

# Sample output:

# Using configuration from /home/jones/openssl/openssl.cnf

# Generating a 1024 bit RSA private key

# .....................................++++++

# .............................................++++++

# writing new private key to '/home/jones/openssl/client-key.pem'

# Enter PEM pass phrase:

# Verifying password - Enter PEM pass phrase:

# -----

# You are about to be asked to enter information that will be

# incorporated into your certificate request.

# What you are about to enter is what is called a Distinguished Name

# or a DN.

# There are quite a few fields but you can leave some blank

# For some fields there will be a default value,

# If you enter '.', the field will be left blank.

# -----

# Country Name (2 letter code) [AU]:FI

# State or Province Name (full name) [Some-State]:.

# Locality Name (eg, city) []:

# Organization Name (eg, company) [Internet Widgits Pty Ltd]:MySQL AB

# Organizational Unit Name (eg, section) []:

# Common Name (eg, YOUR name) []:MySQL user

# Email Address []:

#

# Please enter the following 'extra' attributes

# to be sent with your certificate request

# A challenge password []:

# An optional company name []:

#

# Remove the passphrase from the key

#

openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem

#

# Sign client cert

#

openssl ca -cert $DIR/ca.pem -policy policy\_anything \

-out $DIR/client-cert.pem -config $DIR/openssl.cnf \

-infiles $DIR/client-req.pem

# Sample output:

# Using configuration from /home/jones/openssl/openssl.cnf

# Enter PEM pass phrase:

# Check that the request matches the signature

# Signature ok

# The Subjects Distinguished Name is as follows

# countryName :PRINTABLE:'FI'

# organizationName :PRINTABLE:'MySQL AB'

# commonName :PRINTABLE:'MySQL user'

# Certificate is to be certified until Sep 13 16:45:17 2003 GMT

# (365 days)

# Sign the certificate? [y/n]:y

#

#

# 1 out of 1 certificate requests certified, commit? [y/n]y

# Write out database with 1 new entries

# Data Base Updated

#

# Create a my.cnf file that you can use to test the certificates

#

cat <<EOF > $DIR/my.cnf

[client]

ssl-ca=$DIR/ca.pem

ssl-cert=$DIR/client-cert.pem

ssl-key=$DIR/client-key.pem

[mysqld]

ssl\_ca=$DIR/ca.pem

ssl\_cert=$DIR/server-cert.pem

ssl\_key=$DIR/server-key.pem

EOF

##### Example 3: Creating SSL Files on Windows

Download OpenSSL for Windows if it is not installed on your system. An overview of available packages can be seen here:

http://www.slproweb.com/products/Win32OpenSSL.html

Choose the Win32 OpenSSL Light or Win64 OpenSSL Light package, depending on your architecture (32-bit or 64-bit). The default installation location is C:\OpenSSL-Win32 or C:\OpenSSL-Win64, depending on which package you downloaded. The following instructions assume a default location of C:\OpenSSL-Win32. Modify this as necessary if you are using the 64-bit package.

If a message occurs during setup indicating '...critical component is missing: Microsoft Visual C++ 2008 Redistributables', cancel the setup and download one of the following packages as well, again depending on your architecture (32-bit or 64-bit):

* Visual C++ 2008 Redistributables (x86), available at:

http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF

* Visual C++ 2008 Redistributables (x64), available at:

http://www.microsoft.com/downloads/details.aspx?familyid=bd2a6171-e2d6-4230-b809-9a8d7548c1b6

After installing the additional package, restart the OpenSSL setup procedure.

During installation, leave the default C:\OpenSSL-Win32 as the install path, and also leave the default option 'Copy OpenSSL DLL files to the Windows system directory' selected.

When the installation has finished, add C:\OpenSSL-Win32\bin to the Windows System Path variable of your server (depending on your version of Windows, the following path-setting instructions might differ slightly):

1. On the Windows desktop, right-click the My Computer icon, and select **Properties**.
2. Select the **Advanced** tab from the **System Properties** menu that appears, and click the Environment Variables button.
3. Under **System Variables**, select **Path**, then click the Edit button. The **Edit System Variable** dialogue should appear.
4. Add ';C:\OpenSSL-Win32\bin' to the end (notice the semicolon).
5. Press OK 3 times.
6. Check that OpenSSL was correctly integrated into the Path variable by opening a new command console (**Start>Run>cmd.exe**) and verifying that OpenSSL is available:
7. Microsoft Windows [Version ...]
8. Copyright (c) 2006 Microsoft Corporation. All rights reserved.
9. C:\Windows\system32>cd \
10. C:\>openssl
11. OpenSSL> exit <<< If you see the OpenSSL prompt, installation was successful.

C:\>

After OpenSSL has been installed, use instructions similar to those from Example 1 (shown earlier in this section), with the following changes:

* Change the following Unix commands:
* # Create clean environment
* rm -rf newcerts

mkdir newcerts && cd newcerts

On Windows, use these commands instead:

# Create clean environment

md c:\newcerts

cd c:\newcerts

* When a '\' character is shown at the end of a command line, this '\' character must be removed and the command lines entered all on a single line.

After generating the certificate and key files, to use them for SSL connections, see [Section 6.3.1, “Configuring MySQL to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html).

#### Creating RSA Keys Using openssl

This section describes how to use the **openssl** command to set up the RSA key files that enable MySQL to support secure password exchange over unencrypted connections for accounts authenticated by the sha256\_password and caching\_sha2\_password plugins.

**Note**

There are easier alternatives to generating the files required for RSA than the procedure described here: Let the server autogenerate them or use the [**mysql\_ssl\_rsa\_setup**](https://dev.mysql.com/doc/refman/8.0/en/mysql-ssl-rsa-setup.html) program. See [Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”](https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html).

To create the RSA private and public key-pair files, run these commands while logged into the system account used to run the MySQL server so that the files are owned by that account:

openssl genrsa -out private\_key.pem 2048

openssl rsa -in private\_key.pem -pubout -out public\_key.pem

Those commands create 2,048-bit keys. To create stronger keys, use a larger value.

Then set the access modes for the key files. The private key should be readable only by the server, whereas the public key can be freely distributed to client users:

chmod 400 private\_key.pem

chmod 444 public\_key.pem

### Connecting to MySQL Remotely from Windows with SSH

This section describes how to get an encrypted connection to a remote MySQL server with SSH. The information was provided by David Carlson <[dcarlson@mplcomm.com](mailto:dcarlson@mplcomm.com)>.

1. Install an SSH client on your Windows machine. For a comparison of SSH clients, see <http://en.wikipedia.org/wiki/Comparison_of_SSH_clients>.
2. Start your Windows SSH client. Set Host\_Name = ***yourmysqlserver\_URL\_or\_IP***. Set userid=***your\_userid*** to log in to your server. This userid value might not be the same as the user name of your MySQL account.
3. Set up port forwarding. Either do a remote forward (Set local\_port: 3306, remote\_host: ***yourmysqlservername\_or\_ip***, remote\_port: 3306 ) or a local forward (Set port: 3306, host: localhost, remote port: 3306).
4. Save everything, otherwise you must redo it the next time.
5. Log in to your server with the SSH session you just created.
6. On your Windows machine, start some ODBC application (such as Access).
7. Create a new file in Windows and link to MySQL using the ODBC driver the same way you normally do, except type in localhost for the MySQL host server, not ***yourmysqlservername***.

At this point, you should have an ODBC connection to MySQL, encrypted using SSH.