



[Login](#) | [Register](#)

- [Developer Zone](#)
- [Downloads](#)
- [Documentation](#)
- [MySQL Server](#)
- [MySQL Enterprise](#)
- [MySQL Workbench](#)
- [MySQL Cluster](#)
- [Topic Guides](#)
- [Expert Guides](#)
- [Other Docs](#)
- [Archives](#)
- [About](#)

[MySQL 5.1 Reference Manual](#) :: [5 MySQL Server Administration](#) :: [5.5 MySQL User Account Management](#) :: [5.5.2 Adding User Accounts](#)

« [5.5.1 User Names and Passwords](#)  
[5.5.3 Removing User Accounts](#) »

## 5.5.2. Adding User Accounts

You can create MySQL accounts in two ways:

- By using statements intended for creating accounts, such as [CREATE USER](#) or [GRANT](#). These statements cause the server to make appropriate modifications to the grant tables.
- By manipulating the MySQL grant tables directly with statements such as [INSERT](#), [UPDATE](#), or [DELETE](#).

The preferred method is to use account-creation statements because they are more concise and less error-prone than manipulating the grant tables directly. [CREATE USER](#) and [GRANT](#) are described in [Section 12.7.1, “Account Management Statements”](#).

Another option for creating accounts is to use one of several available third-party programs that offer capabilities for MySQL account administration. [phpMyAdmin](#) is one such program.

The following examples show how to use the [mysql](#) client program to set up new accounts. These examples assume that privileges have been set up according to the defaults described in [Section 2.12.2, “Securing the Initial MySQL Accounts”](#). This means that to make changes, you must connect to the MySQL server as the MySQL `root` user, and the `root` account must have the [INSERT](#) privilege for the `mysql` database and the [RELOAD](#) administrative privilege.

As noted in the examples where appropriate, some of the statements will fail if the server's SQL mode has

**Section Navigation** [\[Toggle\]](#)

- [5.5 MySQL User Account Management](#)
- [5.5.1 User Names and Passwords](#)
- [5.5.2 Adding User Accounts](#)
- [5.5.3 Removing User Accounts](#)
- [5.5.4 Setting Account Resource Limits](#)
- [5.5.5 Assigning Account Passwords](#)
- [5.5.6 Using SSL for Secure Connections](#)
- [5.5.7 Connecting to MySQL Remotely from Windows with SSH](#)
- [5.5.8 Auditing MySQL Account Activity](#)

been set to enable certain restrictions. In particular, strict mode ([STRICT\\_TRANS\\_TABLES](#), [STRICT\\_ALL\\_TABLES](#)) and [NO\\_AUTO\\_CREATE\\_USER](#) will prevent the server from accepting some of the statements. Workarounds are indicated for these cases. For more information about SQL modes and their effect on grant table manipulation, see [Section 5.1.6, “Server SQL Modes”](#), and [Section 12.7.1.3, “GRANT Syntax”](#).

First, use the `mysql` program to connect to the server as the MySQL `root` user:

```
shell> mysql --user=root mysql
```

If you have assigned a password to the `root` account, you will also need to supply a `--password` or `-p` option, both for this `mysql` command and for those later in this section.

After connecting to the server as `root`, you can add new accounts. The following statements use [GRANT](#) to set up four new accounts:

```
mysql> CREATE USER 'monty'@'localhost' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'localhost'
-> WITH GRANT OPTION;
mysql> CREATE USER 'monty'@'%' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'monty'@'%'
-> WITH GRANT OPTION;
mysql> CREATE USER 'admin'@'localhost';
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
mysql> CREATE USER 'dummy'@'localhost';
```

The accounts created by these statements have the following properties:

- Two of the accounts have a user name of `monty` and a password of `some_pass`. Both accounts are superuser accounts with full privileges to do anything. The `'monty'@'localhost'` account can be used only when connecting from the local host. The `'monty'@'%'` account uses the `'%'` wildcard for the host part, so it can be used to connect from any host.

It is necessary to have both accounts for `monty` to be able to connect from anywhere as `monty`. Without the `localhost` account, the anonymous-user account for `localhost` that is created by [mysql\\_install\\_db](#) would take precedence when `monty` connects from the local host. As a result, `monty` would be treated as an anonymous user. The reason for this is that the anonymous-user account has a more specific `Host` column value than the `'monty'@'%'` account and thus comes earlier in the `user` table sort order. (`user` table sorting is discussed in [Section 5.4.4, “Access Control, Stage 1: Connection Verification”](#).)

- The `'admin'@'localhost'` account has no password. This account can be used only by `admin` to connect from the local host. It is granted the [RELOAD](#) and [PROCESS](#) administrative privileges. These privileges enable the `admin` user to execute the [mysqladmin reload](#), [mysqladmin refresh](#), and [mysqladmin flush-xxx](#) commands, as well as [mysqladmin processlist](#). No privileges are granted for accessing any databases. You could add such privileges later by issuing other [GRANT](#) statements.
- The `'dummy'@'localhost'` account has no password. This account can be used only to connect from the local host. No privileges are granted. It is assumed that you will grant specific privileges to the account later.

The statements that create accounts with no password will fail if the [NO\\_AUTO\\_CREATE\\_USER](#) SQL mode is

enabled. To deal with this, use an [IDENTIFIED BY](#) clause that specifies a nonempty password.

To check the privileges for an account, use [SHOW GRANTS](#):

```
mysql> SHOW GRANTS FOR 'admin'@'localhost';
+-----+
| Grants for admin@localhost |
+-----+
| GRANT RELOAD, PROCESS ON *.* TO 'admin'@'localhost' |
+-----+
```

As an alternative to [CREATE USER](#) and [GRANT](#), you can create the same accounts directly by issuing [INSERT](#) statements and then telling the server to reload the grant tables using [FLUSH PRIVILEGES](#):

```
shell> mysql --user=root mysql
mysql> INSERT INTO user
->     VALUES ('localhost', 'monty', PASSWORD('some_pass'),
->     'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');
mysql> INSERT INTO user
->     VALUES ('%', 'monty', PASSWORD('some_pass'),
->     'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
->     'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
->     '', '', '', '', 0, 0, 0, 0);
mysql> INSERT INTO user SET Host='localhost', User='admin',
->     Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host, User, Password)
->     VALUES ('localhost', 'dummy', '');
mysql> FLUSH PRIVILEGES;
```

When you create accounts with [INSERT](#), it is necessary to use [FLUSH PRIVILEGES](#) to tell the server to reload the grant tables. Otherwise, the changes go unnoticed until you restart the server. With [CREATE USER](#), [FLUSH PRIVILEGES](#) is unnecessary.

The reason for using the [PASSWORD\(\)](#) function with [INSERT](#) is to encrypt the password. The [CREATE USER](#) statement encrypts the password for you, so [PASSWORD\(\)](#) is unnecessary.

The `'Y'` values enable privileges for the accounts. Depending on your MySQL version, you may have to use a different number of `'Y'` values in the first two [INSERT](#) statements. The [INSERT](#) statement for the `admin` account employs the more readable extended [INSERT](#) syntax using [SET](#).

In the [INSERT](#) statement for the `dummy` account, only the `Host`, `User`, and `Password` columns in the `user` table row are assigned values. None of the privilege columns are set explicitly, so MySQL assigns them all the default value of `'N'`. This is equivalent to what [CREATE USER](#) does.

If strict SQL mode is enabled, all columns that have no default value must have a value specified. In this case, [INSERT](#) statements must explicitly specify values for the `ssl_cipher`, `x509_issuer`, and `x509_subject` columns.

To set up a superuser account, it is necessary only to insert a `user` table row with all privilege columns set to `'Y'`. The `user` table privileges are global, so no entries in any of the other grant tables are needed.

The next examples create three accounts and give them access to specific databases. Each of them has a

user name of `custom` and password of `obscure`.

To create the accounts with [CREATE USER](#) and [GRANT](#), use the following statements:

```
shell> mysql --user=root mysql
mysql> CREATE USER 'custom'@'localhost' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
->     ON bankaccount.*
->     TO 'custom'@'localhost';
mysql> CREATE USER 'custom'@'host47.example.com' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
->     ON expenses.*
->     TO 'custom'@'host47.example.com';
mysql> CREATE USER 'custom'@'server.domain' IDENTIFIED BY 'obscure';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
->     ON customer.*
->     TO 'custom'@'server.domain';
```

The three accounts can be used as follows:

- The first account can access the `bankaccount` database, but only from the local host.
- The second account can access the `expenses` database, but only from the host `host47.example.com`.
- The third account can access the `customer` database, but only from the host `server.domain`.

To set up the `custom` accounts without [GRANT](#), use [INSERT](#) statements as follows to modify the grant tables directly:

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
->     VALUES ('localhost','custom',PASSWORD('obscure'));
mysql> INSERT INTO user (Host,User,Password)
->     VALUES ('host47.example.com','custom',PASSWORD('obscure'));
mysql> INSERT INTO user (Host,User,Password)
->     VALUES ('server.domain','custom',PASSWORD('obscure'));
mysql> INSERT INTO db
->     (Host,Db,User,Select_priv,Insert_priv,
->     Update_priv>Delete_priv>Create_priv,Drop_priv)
->     VALUES ('localhost','bankaccount','custom',
->     'Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
->     (Host,Db,User,Select_priv,Insert_priv,
->     Update_priv>Delete_priv>Create_priv,Drop_priv)
->     VALUES ('host47.example.com','expenses','custom',
->     'Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
->     (Host,Db,User,Select_priv,Insert_priv,
```

```

-> Update_priv,Delete_priv,Create_priv,Drop_priv)
-> VALUES ('server.domain','customer','custom',
-> 'Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;

```

The first three [INSERT](#) statements add `user` table entries that permit the user `custom` to connect from the various hosts with the given password, but grant no global privileges (all privileges are set to the default value of `'N'`). The next three [INSERT](#) statements add `db` table entries that grant privileges to `custom` for the `bankaccount`, `expenses`, and `customer` databases, but only when accessed from the proper hosts. As usual when you modify the grant tables directly, you must tell the server to reload them with [FLUSH PRIVILEGES](#) so that the privilege changes take effect.

To create a user who has access from all machines in a given domain (for example, `mydomain.com`), you can use the `"%"` wildcard character in the host part of the account name:

```
mysql> CREATE USER 'myname'@'%.mydomain.com' IDENTIFIED BY 'mypass';
```

To do the same thing by modifying the grant tables directly, do this:

```

mysql> INSERT INTO user (Host,User,Password,...)
-> VALUES ('%.mydomain.com','myname',PASSWORD('mypass'),...);
mysql> FLUSH PRIVILEGES;

```

Copyright © 1997, 2012, Oracle and/or its affiliates. All rights reserved. [Legal Notices](#)

[Previous](#) / [Next](#) / [Up](#) / [Table of Contents](#)

## User Comments

Posted by Chinmay Jain on November 11 2011 5:33am

[\[Delete\]](#) [\[Edit\]](#)

```

DROP PROCEDURE IF EXISTS sa_rootdb.CreateUser;
CREATE PROCEDURE sa_rootdb.`CreateUser`(sUser Char(16),sPassword Char(41),sSchema Char(20))
Begin

```

```

Set @Query = CONCAT_WS("','create user ',char(39) , sUser , char(39),'@',char(39) , 'localhost',CHAR(39), '
identified by ', char(39) , sPassword , Char(39));

```

```

PREPARE stmt1 FROM @Query;
Execute stmt1 ;

```

```

Set @Query = CONCAT_WS("','grant SELECT,INSERT,DELETE,UPDATE on ', sSchema , '.* to ',char(39)
,sUser,CHAR(39), '@', char(39) , 'localhost' , Char(39));

```

```


PREPARE stmt1 FROM @Query;
Execute stmt1 ;

```

End;

[Add your own comment.](#)

[Top](#) / [Previous](#) / [Next](#) / [Up](#) / [Table of Contents](#)

 © 2012, Oracle Corporation and/or its affiliates