Short List of MySQL Commands

Conventions used here:

- MySQL key words are shown in CAPS
- User-specified names are in small letters
- Optional items are enclosed in square brackets [ ]
- Items in parentheses must appear in the command, along with the parentheses
- Items that can be repeated as often as desired are indicated by an ellipsis ...

Quoting in MySQL statments

- Don't quote database, table, or column names
- Don't quote column types or modifiers
- Don't quote numerical values
- Quote (single or double) non-numeric values
- Quote file names and passwords
- User names are NOT quoted in GRANT or REVOKE statements, but they are quoted in other statements.

General Commands

USE database_name

Change to this database. You need to change to some database when you first connect to MySQL.

SHOW DATABASES

Lists all MySQL databases on the system.

SHOW TABLES [FROM database_name]

Lists all tables from the current database or from the database given in the command.

DESCRIBE table_name

SHOW FIELDS FROM table_name

SHOW COLUMNS FROM table_name

These commands all give a list of all columns (fields) from the given table, along with column type and other info.

SHOW INDEX FROM table_name

Lists all indexes from this tables.

SET PASSWORD=PASSWORD('new_password')

Allows the user to set his/her own password.

Table Commands

CREATE TABLE table_name (create_clause1, create_clause2, ...)

Creates a table with columns as indicated in the create clauses.

create_clause

column name followed by column type, followed optionally by modifiers. For example, "gene_id INT AUTO_INCREMENT PRIMARY KEY" (without the quotes) creates a column of type integer with the modifiers described below.

create_clause modifiers

- AUTO_INCREMENT : each data record is assigned the next sequential number when it is given a NULL value.
- PRIMARY KEY : Items in this column have unique names, and the table is indexed automatically based on this column. One column must be the PRIMARY KEY, and only one column may be the PRIMARY KEY. This column should also be NOT NULL.
- NOT NULL : No NULL values are allowed in this column: a NULL generates an error message as the data is inserted into the table.
- DEFAULT value : If a NULL value is used in the data for this column, the default value is entered instead.

DROP TABLE table_name

Removes the table from the database. Permanently! So be careful with this command!

ALTER TABLE table_name ADD (create_clause1, create_clause2, ...)

Adds the listed columns to the table.

ALTER TABLE table_name DROP column_name

Drops the listed columns from the table.

ALTER TABLE table_name MODIFY create_clause

Changes the type or modifiers to a column. Using MODIFY means that the column keeps the same name even though its type is altered. MySQL attempts to convert the data to match the new type: this can cause problems.

ALTER TABLE table_name CHANGE column_name create_clause

Changes the name and type or modifiers of a column. Using CHANGE (instead of MODIFY) implies that the column is getting a new name.

ALTER TABLE table_name ADD INDEX [index_name] (column_name1, column_name2, ...)

CREATE INDEX index_name ON table_name (column_name1, column_name2, ...)

Adds an index to this table, based on the listed columns. Note that the order of the columns is important, because additional indexes are created from all subsets of the listed columns reading from left to write. The index name is optional if you use ALTER TABLE, but it is necesary if you use CREATE INDEX. Rarely is the name of an index useful (in my experience).

Data Commands

INSERT [INTO] table_name VALUES (value1, value2, ...)

Insert a complete row of data, giving a value (or NULL) for every column in the proper order.

INSERT [INTO] table_name (column_name1, column_name2, ...) VALUES (value1, value2, ...)

INSERT [INTO] table_name SET column_name1=value1, column_name2=value2, ...

Insert data into the listed columns only. Alternate forms, with the SET form showing column assignments more explicitly.

INSERT [INTO] table_name (column_name1, column_name2, ...) SELECT list_of_fields_from_another_table FROM other_table_name WHERE where_clause

Inserts the data resulting from a SELECT statement into the listed columns. Be sure the number of items taken from the old table match the number of columns they are put into!

DELETE FROM table_name WHERE where_clause

Delete rows that meet the conditions of the where_clause. If the WHERE statement is omitted, the table is emptied, although its structure remains intact.

UPDATE table_name SET column_name1=value1, column_name2=value2, ... [WHERE where_clause]

Alters the data within a column based on the conditions in the where_clause.

LOAD DATA LOCAL INFILE 'path to external file' INTO TABLE table_name

Loads data from the listed file into the table. The default assumption is that fields in the file are separated by tabs, and each data record is separated from the others by a newline. It also

assumes that nothing is quoted: quote marks are considered to be part of the data. Also, it assumes that the number of data fields matches the number of table columns. Columns that are AUTO_INCREMENT should have NULL as their value in the file.

LOAD DATA LOCAL INFILE 'path to external file' [FIELDS TERMINATED BY 'termination_character'] [FIELDS ENCLOSED BY 'quoting character'] [LINES TERMINATED BY 'line termination character'] FROM table_name

Loads data from the listed file into the table, using the field termination character listed (default is tab \t), and/or the listed quoting character (default is nothing), and/or the listed line termination chacracter (default is a newline \n).

SELECT column_name1, column_name2, ... INTO OUTFILE 'path to external file' [FIELDS TERMINATED BY 'termination_character'] [FIELDS ENCLOSED BY 'quoting character'] [LINES TERMINATED BY 'line termination character'] FROM table_name [WHERE where_clause]

Allows you to move data from a table into an external file. The field and line termination clauses are the same as for LOAD above. Several tricky features:

1. Note the positions of the table_name and where_clause, after the external file is given.
2. You must use a complete path, not just a file name. Otherwise MySQL attempts to write to the directory where the database is stored, where you don't have permission to write.
3. The user who is writing the file is 'mysql', not you! This means that user 'mysql' needs permission to write to the directory you specify. The best way to do that is to creat a new directory under your home directory, then change the directory's permission to 777, then write to it. For example: **mkdir mysql_output**, **chmod 777 mysql_output**.

Privilege Commands

Most of the commands below require MySQL root access

GRANT USAGE ON *.* TO user_name@localhost [IDENTIFIED BY 'password']

Creates a new user on MySQL, with no rights to do anything. The IDENTIFED BY clause creates or changes the MySQL password, which is not necessarily the same as the user's system password. The @localhost after the user name allows usage on the local system, which is usually what we do; leaving this off allows the user to access the database from another system. User name NOT in quotes.

GRANT SELECT ON *.* TO user_name@localhost

In general, unless data is supposed to be kept private, all users should be able to view it. A

debatable point, and most databases will only grant SELECT privileges on particular databases. There is no way to grant privileges on all databses EXCEPT specifically enumerated ones.

GRANT ALL ON database_name.* TO user_name@localhost

Grants permissions on all tables for a specific database (database_name.*) to a user. Permissions are for: ALTER, CREATE, DELETE, DROP, INDEX, INSERT, SELECT, UPDATE.

FLUSH PRIVILEGES

Needed to get updated privileges to work immediately. You need RELOAD privileges to get this to work.

SET PASSWORD=PASSWORD('new_password')

Allows the user to set his/her own password.

REVOKE ALL ON [database_name.]* FROM user_name@localhost

Revokes all permissions for the user, but leaves the user in the MySQL database. This can be done for all databases using "ON *", or for all tables within a specific databse, using "ON database_name.*".

DELETE FROM mysql.user WHERE user='user_name@localhost'

Removes the user from the database, which revokes all privileges. Note that the user name is in quotes here.

UPDATE mysql.user SET password=PASSWORD('my_password') WHERE user='user_name'

Sets the user's password. The PASSWORD function encrypts it; otherwise it will be in plain text.

SELECT user, host, password, select_priv, insert_priv, shutdown_priv, grant_priv FROM mysql.user

A good view of all users and their approximate privileges. If there is a password, it will by an encrytped string; if not, this field is blank. Select is a very general privlege; insert allows table manipulation within a database; shutdown allows major system changes, and should only be usable by root; the ability to grant permissions is separate from the others.

SELECT user, host, db, select_priv, insert_priv, grant_priv FROM mysql.db

View permissions for individual databases.