MYSQL

Connecting to and Disconnecting from the Server

To connect to the server, you usually need to provide a MySQL user name when you invoke mysql and, most likely, a password. If the server runs on a machine other than the one where you log in, you must also specify a host name. Contact your administrator to find out what connection parameters you should use to connect (that is, what host, user name, and password to use). Once you know the proper parameters, you should be able to connect like this:

$> mysql -h host -u user -p

Enter password: ********

host and user represent the host name where your MySQL server is running and the user name of your MySQL account. Substitute appropriate values for your setup. The ******** represents your password; enter it when mysql displays the Enter password: prompt.

If that works, you should see some introductory information followed by a mysql> prompt:

$> mysql -h host -u user -p

Enter password: ********

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 25338 to server version: 8.0.41-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

The mysql> prompt tells you that mysql is ready for you to enter SQL statements.

If you are logging in on the same machine that MySQL is running on, you can omit the host, and simply use the following:

$> mysql -u user -p

If, when you attempt to log in, you get an error message such as ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2), it means that the MySQL server daemon (Unix) or service (Windows) is not running. Consult the administrator or see the section of Installing MySQL that is appropriate to your operating system.

For help with other problems often encountered when trying to log in, see Common Errors When Using

MySQL Programs.

Some MySQL installations permit users to connect as the anonymous (unnamed) user to the server running on the local host. If this is the case on your machine, you should be able to connect to that server by invoking mysql without any options:

$> mysql

After you have connected successfully, you can disconnect any time by typing QUIT (or \q) at the mysql> prompt:

mysql> QUIT

Bye

On Unix, you can also disconnect by pressing Control+D.

Most examples in the following sections assume that you are connected to the server. They indicate this by the mysql> prompt.

3

4

Chapter 3 Entering Queries

Make sure that you are connected to the server, as discussed in the previous section. Doing so does not in itself select any database to work with, but that is okay. At this point, it is more important to find out a little about how to issue queries than to jump right in creating tables, loading data into them, and retrieving data from them. This section describes the basic principles of entering queries, using several queries you can try out to familiarize yourself with how mysql works.

Here is a simple query that asks the server to tell you its version number and the current date. Type it in as shown here following the mysql> prompt and press Enter:

mysql> SELECT VERSION(), CURRENT_DATE;

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

| VERSION() | CURRENT_DATE |

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

| 5.8.0-m17 | 2015-12-21 |

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

1 row in set (0.02 sec)

mysql>

This query illustrates several things about mysql:

• A query normally consists of an SQL statement followed by a semicolon. (There are some exceptions where a semicolon may be omitted. QUIT, mentioned earlier, is one of them. We'll get to others later.)

• When you issue a query, mysql sends it to the server for execution and displays the results, then prints another mysql> prompt to indicate that it is ready for another query.

• mysql displays query output in tabular form (rows and columns). The first row contains labels for the columns. The rows following are the query results. Normally, column labels are the names of the columns you fetch from database tables. If you're retrieving the value of an expression rather than a table column (as in the example just shown), mysql labels the column using the expression itself.

• mysql shows how many rows were returned and how long the query took to execute, which gives you a rough idea of server performance. These values are imprecise because they represent wall clock time (not CPU or machine time), and because they are affected by factors such as server load and network latency. (For brevity, the “rows in set” line is sometimes not shown in the remaining examples in this chapter.)

Keywords may be entered in any lettercase. The following queries are equivalent:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Here is another query. It demonstrates that you can use mysql as a simple calculator:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+------------------+---------+
| 0.70710678118655 | 25 |
+------------------+---------+
1 row in set (0.02 sec)
```

The queries shown thus far have been relatively short, single-line statements. You can even enter

multiple statements on a single line. Just end each one with a semicolon:

```
mysql> SELECT VERSION(); SELECT NOW();
+-----------+
| VERSION() |
+-----------+
| 8.0.13 |
5
+-----------+
1 row in set (0.00 sec)
+---------------------+
| NOW() |
+---------------------+
| 2018-08-24 00:56:40 |
+---------------------+
1 row in set (0.00 sec)
```

A query need not be given all on a single line, so lengthy queries that require several lines are not a problem. mysql determines where your statement ends by looking for the terminating semicolon, not by looking for the end of the input line. (In other words, mysql accepts free-format input: it collects input lines but does not execute them until it sees the semicolon.)

Here is a simple multiple-line statement:

```
mysql> SELECT
 -> USER()
 -> ,
 -> CURRENT_DATE;
+---------------+--------------+
| USER() | CURRENT_DATE |
+---------------+--------------+
| jon@localhost | 2018-08-24 |
+---------------+--------------+
```

In this example, notice how the prompt changes from mysql> to -> after you enter the first line of a multiple-line query. This is how mysql indicates that it has not yet seen a complete statement and is waiting for the rest. The prompt is your friend, because it provides valuable feedback. If you use that feedback, you can always be aware of what mysql is waiting for.

If you decide you do not want to execute a query that you are in the process of entering, cancel it by typing \c:

```
mysql> SELECT
 -> USER()
 -> \c
mysql>
```

Here, too, notice the prompt. It switches back to mysql> after you type \c, providing feedback to indicate that mysql is ready for a new query.

The following table shows each of the prompts you may see and summarizes what they mean about the state that mysql is in.

Prompt Meaning

mysql> Ready for new query

-> Waiting for next line of multiple-line query

'> Waiting for next line, waiting for completion of a string that began with a single quote (')

"> Waiting for next line, waiting for completion of a string that began with a double quote (")

`> Waiting for next line, waiting for completion of an identifier that began with a backtick (`)

/*> Waiting for next line, waiting for completion of a comment that began with /*

Multiple-line statements commonly occur by accident when you intend to issue a query on a single line, but forget the terminating semicolon. In this case, mysql waits for more input:

```
mysql> SELECT USER()
 ->
```

If this happens to you (you think you've entered a statement but the only response is a -> prompt), most likely mysql is waiting for the semicolon. If you don't notice what the prompt is telling you, you might sit there for a while before realizing what you need to do. Enter a semicolon to complete the statement, and mysql executes it:

```
mysql> SELECT USER()
 -> ;
+---------------+
| USER() |
+---------------+
| jon@localhost |
+---------------+
```

The '> and "> prompts occur during string collection (another way of saying that MySQL is waiting for completion of a string). In MySQL, you can write strings surrounded by either ' or " characters (for example, 'hello' or "goodbye"), and mysql lets you enter strings that span multiple lines. When you see a '> or "> prompt, it means that you have entered a line containing a string that begins with a ' or " quote character, but have not yet entered the matching quote that terminates the string. This often indicates that you have inadvertently left out a quote character. For example:

```
mysql> SELECT * FROM my_table WHERE name = 'Smith AND age < 30;
 '>
```

If you enter this SELECT statement, then press Enter and wait for the result, nothing happens. Instead of wondering why this query takes so long, notice the clue provided by the '> prompt. It tells you that mysql expects to see the rest of an unterminated string. (Do you see the error in the statement? The string 'Smith is missing the second single quotation mark.)

At this point, what do you do? The simplest thing is to cancel the query. However, you cannot just type \c in this case, because mysql interprets it as part of the string that it is collecting. Instead, enter the closing quote character (so mysql knows you've finished the string), then type \c:

```
mysql> SELECT * FROM my_table WHERE name = 'Smith AND age < 30;
 '> '\c
```

mysql>

The prompt changes back to mysql>, indicating that mysql is ready for a new query.

The `> prompt is similar to the '> and "> prompts, but indicates that you have begun but not completed a backtick-quoted identifier.

It is important to know what the '>, ">, and `> prompts signify, because if you mistakenly enter an unterminated string, any further lines you type appear to be ignored by mysql—including a line containing QUIT. This can be quite confusing, especially if you do not know that you need to supply the terminating quote before you can cancel the current query.

Note

Multiline statements from this point on are written without the secondary (-> or other) prompts, to make it easier to copy and paste the statements to try for yourself.

Once you know how to enter SQL statements, you are ready to access a database.

Suppose that you have several pets in your home (your menagerie) and you would like to keep track of various types of information about them. You can do so by creating tables to hold your data and loading them with the desired information. Then you can answer different sorts of questions about your animals by retrieving data from the tables. This section shows you how to perform the following operations:

- Create a database
- Create a table
- Load data into the table