MySQL

Essentials

MySQL 5 Essentials

MySQL 5 Essentials – Edition 1.0

Table of Contents

Chapter 1. About MySQL Essentials

Before the arrival of MySQL, implementing a database was typically a complex and expensive task involving the purchase, installation and maintenance of a proprietary database management system from a vendor such as Oracle or IBM. In contrast, MySQL provides a free, open source database management system that is easy to install, implement and maintain. In addition, MySQL is fast, extremely reliable and widely deployed by many companies and organizations throughout the world.

It is not an exaggeration to say that MySQL has brought the power of a fully featured, scalable relational database management system into the reach of anyone with a computer and the desire to build a data driven application or web site. With this goal in mind, MySQL Essentials is designed to provide a step by step path to proficiency with MySQL.

Beginning with the basics of relational databases, this book goes on to cover everything needed to successfully build, maintain and secure MySQL based databases.

Topics covered include the MySQL database architecture, use of the command line tool and graphical MySQL Workbench, database creation and management, the SQL language, regular expressions and working with views, dates, calculations. In addition, details are provided on monitoring databases and managing user access and privileges.

It is intended that upon completion of this book, the reader will have a firm understanding of both MySQL and the SQL language, and be able to confidently create manage and maintain MySQL based databases.

Chapter 2. Database Basics

2.1 What is a Database?

The chances are that if you have ever logged into a web site or purchased an item on the internet you have interacted with a database in some way. Anything that involves the retrieval or storage of information on a computer system is most likely to involve a database. In fact, databases are the core of just about every application that relies on data of some form to complete a task.

The first step in learning MySQL is to understand the difference between a *database* and a *database management system (DBMS)*. The term *database* refers to the entity that stores the actual data (such as ID numbers, names and addresses for example) in a structured way. A *database management system (DBMS)* on the other hand, refers to the software used to store, access and manipulate the data stored in the *database*. All interactions with the database are always performed via the *DBMS*.

Modern databases and database management systems are not restricted to storing just text. Today, databases are used to store such items as images, videos and software objects.

2.2 Understanding Database Tables

Database *Tables* provide the most basic level of data structure in a database. Each database can contain multiple tables and each table is designed to hold information of a specific type. For example, a database may contain a *customer* table which contains the name, address and telephone number for all the customers of a particular business. The same database may also include a *products* table used to store the product descriptions with associated product codes for the items sold by the business.

Each table in a database is assigned a name which must be unique within that particular database. A table name, once assigned to a table in one database, may only be re-used within the context of a different database.

2.3 Introducing Database Schema

Database Schema define the characteristics of the data stored in a database table. For example, the table schema for a customer database table might define that the customer name is a string of no more than 20 characters in length, and that the customer phone number is a numerical data field of a certain format.

Schema are also used to define the structure of entire databases and the relationship between the various tables contained in each database.

2.4 Columns and Datatypes

It is helpful at this stage to begin to view a database table as being similar to a spreadsheet where data is stored in rows and columns.

Each column represents a data field in the corresponding table. For example, the name, address and telephone data fields of a table are all *columns*.

Each column, in turn, is defined to contain a certain *datatype* which dictates the type of data the column can contain. A column designed to store numbers would, therefore, be defined as a numerical datatype.

2.5 Database Rows

Each new record that is saved to a table is stored in a row. Each row, in turn, consists of the columns of data associated with the saved record.

Once again, consider the spreadsheet analogy described earlier in this chapter. Each entry in a customer table is equivalent to a row in a spreadsheet and each column contains the data for each customer (name, address, telephone etc). When a new customer is added to the table, a new row is created and the data for that customer stored in the corresponding columns of the new row.

Rows are also sometimes referred to as *records* and these terms can generally be used interchangeably.

2.6 Introducing Primary Keys

Each database table must contain one or more columns that can be used to uniquely identify each row in the table. This is known in database terminology as the *Primary Key*. For example, a table may use a bank account number column as the primary key. Alternatively, a customer table may use the customer's social security number of the primary key.

Primary keys allow the database management system to uniquely identify a specific row in a table. Without a primary key it would not be possible to retrieve or delete a specific row in a table because there can be no certainty that the correct row has been selected. For example, suppose a table existed where the customer's last name had been defined as the primary key. Imagine then the problem that might arise if more than one customer called "Smith" was recorded in the database. Without some guaranteed way to uniquely identify a specific row it would be impossible to ensure the correct data was being accessed at any given time.

Primary keys can comprise a single column or multiple columns in a table. To qualify as a single column primary key, no two rows can contain matching primary key values. When using multiple columns to construct a primary key, individual column values do not need to be unique, but all the columns combined *must be unique*.

Finally, whilst primary keys are not mandatory in database tables their use is strongly recommended.

2.7 What is SQL?

As discussed previously, a *database management system* (DBMS) provides the means to access the data stored in a database. One of the key methods for achieving this is via a language called the Structured Query Language. This is usually abbreviated to SQL and pronounced *Sequel*.

SQL is essentially a very simple and easy to use language designed specifically to enable the reading and writing of database data. Because SQL contains a small set of keywords it can be learned quickly. In addition, SQL syntax is more or less identical between most DBMS implementations, so having learned SQL for one system, it is likely that your skills will transfer to other database management systems.

Throughout the remainder of this book particular attention will be paid to explaining the key SQL commands so that the reader will be proficient in using SQL to read and write database data.

Chapter 3. MySQL Database Architecture

There are two flavors of Database Management System (DBMS) known as shared-file and client-server. A shared file based DBMS consists of a database access application which interacts directly with the underlying database files. These types of database are typically designed for less demanding data storage needs and are used almost exclusively on desktop computers. Microsoft Access is a typical example of this category of DBMS. Such database systems are never used in distributed or enterprise level environments.

MySQL falls into the client-server DBMS category. A client-server DBMS is split into two components. The server component typically resides on the same physical computer as the database files and is responsible for all interactions with the database. The second component is the client. The client sends all database requests to the server which in turn processes the request and returns the results of the request back to the client.

There are a couple of key advantages to the client-server architecture DBMS. Firstly, there is no need for the client to be running on the same computer system as the server. Instead, requests can be sent by the client over a network or internet connection to the server on a remote host. The fact that the server resides on a remote computer is invisible to the client user. This makes the database available to greater numbers of users than a shared-file DBMS offers. In large scale enterprise environments this also allows high levels of fault tolerance and load balancing to be implemented.

Secondly, separating the client from the server allows a wider range of client types to be used to access the database. Valid clients can be the MySQL tools, applications written in other programming languages such as C, C++ or Java, or web based applications developed using languages such as PHP or JSP.

Whilst the advantages of the client-server architecture are clear, it is important to appreciate that there is nothing to prevent both the client and server from running on the same physical computer. The majority of this book will assume that the reader is running both the MySQL client and server components on the same computer system.

Chapter 4. The mysql Command-Line Tool

In MySQL Database Architecture we covered the fact that MySQL is a client-server based database management system (DBMS). In this chapter we will begin by looking at the *mysql* client tool, arguably the most valuable of the *client tools* provided with MySQL.

4.1 The mysql Command-line Utility

The *mysql* tool is probably the most useful utility and is the tool that you will likely use the most as you learn and continue to use MySQL. *mysql* is a command-line client tool that is installed as standard with the MySQL package. From the *mysql* command-prompt it is possible to issue a wide range of commands to the database server such as creating and deleting databases and tables, searching for data, adding new rows and much more. Throughout this book the capabilities of the *mysql* tool will be covered in great detail.

Assuming MySQL has been installed the *mysql* tool may be loaded at the operating system command-prompt as follows:

```
mysql
```

The above command will display the following output:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 234343 to server version: 5.0.27

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

mysql>
```

If you see an error message, it may be that your database system is configured to require login credentials, or that the server is running on a different system. For example, if your database server requires a username and password to gain access the -u and -p command-line options may be used respectively:

```
mysql -u john -p
```

The above command will prompt for the password for user *john* before allowing access to the database management system.

If the *mysql* client is running a different system to the MySQL server, the -h flag may be used to specify the name of the remote host together with -P to specify the port:

```
mysql -h myDBServer -p 5678
```

A list of command-line options can be obtained running *mysql --help* at the command-prompt (listed below).

Once *mysql* is running, commands are entered at the *mysql>* prompt. Typing *help* at this prompt will display a list of the commands supported by the tool (listed below).

Commands are terminated by a semi-colon (;). If a command is not terminated by a semi-colon, pressing enter simply continues the current command on the following line.

To exit from *mysql* type *quit* at the *mysql>* command-prompt.

4.2 mysql Command-Line Arguments

The *mysql* tool supports the following command-line options:

```
Usage: mysql [OPTIONS] [database]
  -?, --help          Display this help and exit.
  -I, --help          Synonym for -?
  --auto-rehash       Enable automatic rehashing. One doesn't need to use
                      'rehash' to get table and field completion, but startup
                      and reconnecting may take a longer time. Disable with
                      --disable-auto-rehash.
  -A, --no-auto-rehash
                      No automatic rehashing. One has to use 'rehash' to get
                      table and field completion. This gives a quicker start
of
                      mysql and disables rehashing on reconnect. WARNING:
                      options deprecated; use --disable-auto-rehash instead.
  -B, --batch         Don't use history file. Disable interactive behavior.
                      (Enables --silent)
  --character-sets-dir=name
                      Directory where character sets are.
  --default-character-set=name
                      Set the default character set.
  -C, --compress      Use compression in server/client protocol.
  -#, --debug[=#]     This is a non-debug version. Catch this and exit
  -D, --database=name Database to use.
  --delimiter=name    Delimiter to be used.
  -e, --execute=name  Execute command and quit. (Disables --force and history
                      file)
  -E, --vertical      Print the output of a query (rows) vertically.
  -f, --force         Continue even if we get an sql error.
  -G, --named-commands
                      Enable named commands. Named commands mean this
program's
                      internal commands; see mysql> help . When enabled, the
                      named commands can be used from any line of the query,
                      otherwise only from the first line, before an enter.
                      Disable with --disable-named-commands. This option is
                      disabled by default.
  -g, --no-named-commands
                      Named commands are disabled. Use \* form only, or use
                      named commands only in the beginning of a line ending
                      with a semicolon (;) Since version 10.9 the client now
                      starts with this option ENABLED by default! Disable
with
                      '-G'. Long format commands still work from the first
                      line. WARNING: option deprecated; use
                      --disable-named-commands instead.
  -i, --ignore-spaces Ignore space after function names.
  --local-infile      Enable/disable LOAD DATA LOCAL INFILE.
  -b, --no-beep       Turn off beep on error.
  -h, --host=name     Connect to host.
  -H, --html          Produce HTML output.
  -X, --xml           Produce XML output
  --line-numbers      Write line numbers for errors.
  -L, --skip-line-numbers
                      Don't write line number for errors. WARNING: -L is
                      deprecated, use long version of this option instead.
  -n, --unbuffered    Flush buffer after each query.
  --column-names      Write column names in results.
  -N, --skip-column-names
                      Don't write column names in results. WARNING: -N is
                      deprecated, use long version of this options instead.
  -O, --set-variable=name
                      Change the value of a variable. Please note that this
                      option is deprecated; you can set variables directly
with
                      --variable-name=value.
  --sigint-ignore     Ignore SIGINT (CTRL-C)
  -o, --one-database  Only update the default database. This is useful for
                      skipping updates to other database in the update log.
  --pager[=name]      Pager to use to display results. If you don't supply an
                      option the default pager is taken from your ENV
variable
                      PAGER. Valid pagers are less, more, cat [> filename],
                      etc. See interactive help (\h) also. This option does
not
                      work in batch mode. Disable with --disable-pager. This
                      option is disabled by default.
  --no-pager          Disable pager and print to stdout. See interactive help
                      (\h) also. WARNING: option deprecated; use
                      --disable-pager instead.
  -p, --password[=name]
                      Password to use when connecting to server. If password
is
                      not given it's asked from the tty.
  -P, --port=#        Port number to use for connection.
  --prompt=name       Set the mysql prompt to this value.
  --protocol=name     The protocol of connection (tcp,socket,pipe,memory).
  -q, --quick         Don't cache result, print it row by row. This may slow
                      down the server if the output is suspended. Doesn't use
                      history file.
  -r, --raw           Write fields without conversion. Used with --batch.
  --reconnect         Reconnect if the connection is lost. Disable with
                      --disable-reconnect. This option is enabled by default.
  -s, --silent        Be more silent. Print results with a tab as separator,
                      each row on new line.
  -S, --socket=name   Socket file to use for connection.
  --ssl               Enable SSL for connection (automatically enabled with
                      other flags). Disable with --skip-ssl.
  --ssl-ca=name       CA file in PEM format (check OpenSSL docs, implies
                      --ssl).
  --ssl-capath=name   CA directory (check OpenSSL docs, implies --ssl).
  --ssl-cert=name     X509 cert in PEM format (implies --ssl).
  --ssl-cipher=name   SSL cipher to use (implies --ssl).
  --ssl-key=name      X509 key in PEM format (implies --ssl).
  --ssl-verify-server-cert
                      Verify server's "Common Name" in its cert against
                      hostname used when connecting. This option is disabled
by
                      default.
  -t, --table         Output in table format.
  -T, --debug-info    Print some debug info at exit.
  --tee=name          Append everything into outfile. See interactive help
(\h)
                      also. Does not work in batch mode. Disable with
                      --disable-tee. This option is disabled by default.
  --no-tee            Disable outfile. See interactive help (\h) also.
WARNING:
                      option deprecated; use --disable-tee instead
  -u, --user=name     User for login if not current user.
  -U, --safe-updates  Only allow UPDATE and DELETE that uses keys.
  -U, --i-am-a-dummy  Synonym for option --safe-updates, -U.
  -v, --verbose       Write more. (-v -v -v gives the table output format).
  -V, --version       Output version information and exit.
  -w, --wait          Wait and retry if connection is down.
  --connect_timeout=# Number of seconds before connection timeout.
  --max_allowed_packet=#
                      Max packet length to send to, or receive from server
  --net_buffer_length=#
                      Buffer for TCP/IP and socket communication
  --select_limit=#    Automatic limit for SELECT when using --safe-updates
  --max_join_size=#   Automatic limit for rows in a join when using
                      --safe-updates
  --secure-auth       Refuse client connecting to server if it uses old
                      (pre-4.1.1) protocol
  --show-warnings     Show warnings after every statement.

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf /etc/my.cnf
The following groups are read: mysql client
The following options may be given as the first argument:
--print-defaults        Print the program argument list and exit
--no-defaults           Don't read default options from any options file
--defaults-file=#       Only read default options from the given file #
--defaults-extra-file=# Read this file after the global files are read

Variables (--variable-name=value)
and boolean options {FALSE|TRUE}  Value (after reading options)
--------------------------------- -----------------------------
auto-rehash                       TRUE
character-sets-dir                (No default value)
default-character-set             latin1
compress                          FALSE
database                          (No default value)
delimiter                         ;
vertical                          FALSE
force                             FALSE
named-commands                    FALSE
local-infile                      FALSE
no-beep                           FALSE
host                              (No default value)
html                              FALSE
xml                               FALSE
line-numbers                      TRUE
unbuffered                        FALSE
column-names                      TRUE
sigint-ignore                     FALSE
port                              0
prompt                            mysql>
```

```
quick                             FALSE
raw                               FALSE
reconnect                         TRUE
socket                            (No default value)
ssl                               FALSE
ssl-ca                            (No default value)
ssl-capath                        (No default value)
ssl-cert                          (No default value)
ssl-cipher                        (No default value)
ssl-key                           (No default value)
ssl-verify-server-cert            FALSE
table                             FALSE
debug-info                        FALSE
user                              (No default value)
safe-updates                      FALSE
i-am-a-dummy                      FALSE
connect_timeout                   0
max_allowed_packet                16777216
net_buffer_length                 16384
select_limit                      1000
max_join_size                     1000000
secure-auth                       FALSE
show-warnings                     FALSE
```

4.3 mysql Commands

The following commands are supported at the *mysql>* prompt:

```
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?          (\?) Synonym for `help'.
clear      (\c) Clear command.
connect    (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter. NOTE: Takes the rest of the line as
new delimiter.
edit       (\e) Edit command with $EDITOR.
```

```
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given
outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing
binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
```

4.4 Summary

This chapter has covered the basics of the mysql command line tool.

Chapter 5. Creating Databases and Tables Using SQL Commands

In this chapter we will learn how to create new databases and tables by issuing SQL commands using the *mysql* client.

This chapter assumes that the *mysql* tool is running and connected to the MySQL database server. If this is not the case and you are unsure as to how to achieve this refer to *The mysql Command-Line Tool*. Alternatively, the SQL commands outlined in that chapter may be executed in the MySQL Workbench as outlined in Using the MySQL Workbench SQL Editor.

5.1 Creating a New MySQL Database

A new database is created using the *CREATE DATABASE* SQL statement followed by the name of the database to be created. The *CREATE SCHEMA* statement may also be used for this purpose. For example, to create a new database called *MySampleDB* the following statement would be entered at the *mysql>* prompt:

```
CREATE DATABASE MySampleDB;
```

If successful, the command will generate output similar to the following:

```
Query OK, 1 row affected (0.00 sec)
```

If the database name specified conflicts with an existing database MySQL will display and error message reporting this fact:

```
ERROR 1007 (HY000): Can't create database 'MySampleDB'; database exists
```

In this situation, a different database name should be selected, or the *IF NOT EXISTS* option should be used. This option only creates the database if it does not already exist:

```
CREATE DATABASE IF NOT EXISTS MySampleDB;
```

5.2 Creating Tables with SQL

New tables are added to an existing database using the SQL *CREATE TABLE* statement. The *CREATE TABLE* statement is followed by the name of the table to be created followed by a comma separated list of the names and definitions of each table column:

```
CREATE TABLE table_name ( column_name definitions, table_name definitions
..., PRIMARY KEY=(column_name) ) ENGINE=engine_type;
```

The definitions field for each column defines information such as the data type of the column, whether the column can be NULL and whether the column value auto increments. The CREATE TABLE statement also allows a column (or group of columns) to be specified as the primary key (see *Database Basics* for a description of primary keys).

Before a table can be created a database must first be selected so that MySQL knows where to create the table. This is achieved using the *USE* SQL statement:

```
USE MySampleDB;
```

Having selected a database, the following example creates a table consisting of three columns named *customer_id*, *customer_name* and *customer_address*. The *customer_id* and *customer_name* columns must contain values (i.e. NOT NULL). The *customer_id* holds an integer value which will auto increment as new rows are added, and the others hold character strings up to 20 characters in length. The primary key is defined to be the *customer_id*

```
CREATE TABLE customer
(
customer_id int NOT NULL AUTO_INCREMENT,
customer_name char(20) NOT NULL,
customer_address char(20) NULL,
PRIMARY KEY (customer_id)
) ENGINE=InnoDB;
```

5.3 Understanding NULL and NOT NULL Values

When a column is specified to be NULL then a row can be added to a database when there is no value assigned to that column. Conversely, if a column is defined as NOT NULL then it must have a value assigned to it before the row can be added to table.

5.4 Primary Keys

As covered in *Database Basics* a primary key is a column used to identify individual records in a table. The value of a primary key column must be unique within the context of the table in which it exists, or if multiple columns are combined to constitute a primary key, the combination of key values must be unique to each row.

The primary key is defined using the *PRIMARY KEY* statement during table creation. If multiple columns are being used they are comma separated:

```
PRIMARY KEY (column_name, column_name ... )
```

In the following example, a table is created using two columns as the primary key:

```
CREATE TABLE product
(
  prod_code INT NOT NULL AUTO_INCREMENT,
  prod_name char(30) NOT NULL,
  prod_desc char(60) NULL,
  PRIMARY KEY (prod_code, prod_name)
) ENGINE=InnoDB;
```

5.5 AUTO_INCREMENT

AUTO_INCREMENT is one of the simplest, yet most useful column definitions in the SQL language. Essentially, when a column is defined using AUTO_INCREMENT the value of the column is increased automatically each time a new row is added to a table. This is especially useful when using a column as a primary key. By using AUTO_INCREMENT it is not necessary to write SQL statements to calculate a new unique id for each row. This is all handled by the MySQL server when the row is added.

There are two rules that must be obeyed when using AUTO_INCREMENT. Firstly, only one column per table may be assigned AUTO_INCREMENT status. Secondly, the AUTO_INCREMENT column must be indexed (e.g. by declaring it as the primary key).

It is possible to override the AUTO_INCREMENT value of a column simply by specifying a value when executing an INSERT statement. As long as the specified value is unique the value provided will be used in the new row and subsequent increments will start at the newly inserted value.