MySQL User Guide

MySQL User Guide

Abstract

This is the MySQL User Guide version 1.0.

Document generated on: 2008-03-15 (revision: 10248)

Table of Contents

List of Figures

List of Tables

Chapter 1. About This Book

1.1. This is *Not* the MySQL Reference Manual

On the MySQL documentation web page (http://dev.mysql.com/doc/) you'll find the various versions of the MySQL Reference manual. At the time of writing there are four different versions, each one running to about 2,000 pages in PDF format. These manuals are meant to be the definitive reference books for each specific server version; they should provide the answer to any question you have about any version of MySQL.

The manuals are an excellent resource but they can be very intimidating for users who are new to MySQL, especially for those who have no previous experience with a Relational Database Management System (RDMS). Even for users familiar with other RDMSs, the wealth of information contained in the manuals makes it easy to lose sight of the forest for the trees. For this reason the focus of the MySQL User Guide is strictly circumscribed; as far as possible, this book aims to be OS-neutral and MySQL version-specific.

This book deals with MySQL version 5.0 [5.1 if it is GA] only. Concentrating on one server version makes for greater simplicity and using version 5.0 means we can take advantage of the improved feature set of the newest production version.

MySQL's popularity is partly due to the fact that it is supported on numerous operating systems (OSs). However, this also adds a level of complexity to the manual — exactly the kind of thing we aim to reduce. For this reason, as far as is possible, this book ignores any OS-specific features of MySQL. For instance, `mysqlhotcopy` is a very useful utility but it is not supported under Windows or Mac OS X so will not be discussed here. Concentrating on features common to all OSs removes one of the distractions inherent in reading the MySQL manual.

In no way is this book meant to be a definitive treatment of MySQL but therein lies its strength. It should speed up the process of getting you up and running with MySQL. For those questions it doesn't answer, see the manual.

MySQL is a flexible RDMS useful in many circumstances. At one end of the spectrum it is suitable for enterprise applications and at the other it can also be used for simple desktop applications. Regardless of how you plan to use MySQL, the MySQL User Guide should prove to be a good introduction to MySQL.

1.2. Target Audience

When describing the nature of the MySQL User Guide in Section 1.1, “This is *Not* the MySQL Reference Manual” we hinted at the kinds of readers that this book should appeal to; users new to databases in general and users new to MySQL in particular.

It is also probably a fair assumption that many of our readers, regardless of the OS they are using, will be more familiar working within a GUI environment than from the command line. There's no getting around the fact that being capable from the command line is an advantage when using MySQL. The primary tool for interacting with a MySQL server, `mysql`, is command-line based.

Mastering MySQL from the command line will allow you to operate in environments where there is no GUI, the majority of web servers for example. Besides, some command-line tools are unquestionably superior. There's no quicker way of creating a database than issuing the command, **`mysqladmin create db_name`**.

However, whenever possible we'll make use of MySQL Administrator and MySQL Query Browser, open source MySQL GUI Tools. Creating database objects is made especially easy using the Table Editor, a feature of the Query Browser also common to other GUI Tools. By pointing and clicking you can quickly build a table without knowing anything about data definition language (DDL). Not only will the table editor help you work more quickly, but it's a good way to learn MySQL's implementation of SQL. Any alterations made to a table using the graphical interface are shown as SQL statements, making it easy to learn the appropriate SQL command.

1.3. We Want Your Help

The idea for a MySQL User Guide came about as a response to the expressed needs of MySQL users and especially community comments about the manuals. This is your chance to help create an important piece of MySQL documentation — The MySQL User Guide.

1.3.1. Contribution Checklist

If you do plan to contribute to the MySQL User Guide, pay special attention to the following sections:

- Section 1.1, “This is *Not* the MySQL Reference Manual”
- Section 1.2, “Target Audience”

Familiarity with these sections is essential to determining an appropriate topic at an appropriate level of complexity.

The MySQL User Guide is licensed under the GNU Free Documentation License (GFDL). For a copy of this license see Appendix D, *GNU Free Documentation License*.

1.3.2. Content of Contributions

The table of contents (TOC) can serve as a guide to the topics we would like to see covered in the MySQL User Guide. Parts I through III deal with introductory topics and later sections deal with more advanced topics. Chapters or sections should integrate closely with related subjects. Some sections will be shorter or longer depending upon the nature of the topic.

We've tried to structure the MySQL User Guide in a format that will encourage a variety of submissions of different lengths at differing levels of difficulty. Some sections have already been completed by MySQL staff and can serve as examples of the kinds of submissions we are seeking. For example, Section 24.2, "Using `mysqldump`" could serve as a template for a section on `mysqlimport` and Chapter 30, *Migrating a Spreadsheet to MySQL* could serve as a guide to Chapter 31, *Migrating an Access Database to MySQL*.

Depending upon your time and expertise you may wish to submit a short, narrowly focused section on a specific introductory topic or a complete chapter on something more advanced.

The TOC is meant as a guideline for possible submissions but you needn't feel constrained by the chapters or sections listed there. If you feel that a topic warrants inclusion but is not included in the TOC, please let us know.

Part I. Getting Started

Table of Contents

Chapter 2. Introduction

2.1. What is MySQL?

2.1.1. Client and Server

2.2. MySQL Applications

2.2.1. MySQL On the Web

2.2.2. MySQL Enterprise Applications

2.2.3. MySQL Desktop Applications

2.2.4. Main Features

Chapter 3. Installing MySQL

3.1. Introduction

Unless you have been provided with a working copy of MySQL from your ISP or employer, the first step in using MySQL is to install MySQL on your local machine or server.

MySQL is available pre-compiled and packaged for a wide variety of platforms including `Microsoft Windows`, `Linux`, `Solaris`, `FreeBSD`, and `Mac OS X` to name a few.

The installation process varies by platform but generally involves downloading an installer or compressed archive, extracting or executing the downloaded file, and then configuring and starting the MySQL server. Once the MySQL server is successfully installed and configured, you can download and install the MySQL GUI tools to manage and query your new server.

In the sections that follow we will explain the process described above with regards to the more popular platforms used with MySQL, namely Microsoft Windows, Linux, and Mac OS X.

3.2. Downloading MySQL

All binary and source versions of MySQL are available at http://dev.mysql.com/downloads/. We strongly recommend you download the latest stable version of MySQL that is available for your platform (currently MySQL 5.0.X, which this userguide is based on).

This chapter will deal with installation from pre-compiled binaries only and it is recommended that you download the same. If you want information on compiling and installing MySQL from source code, please refer to the MySQL Installation Using a Source Distribution section of the MySQL Reference Manual.

The binaries you download will depend on the platform you intend to use. For specific information please refer to the appropriate section that follows.

3.3. Installation

3.3.1. Installing MySQL on <operating system>

Chapter 4. MySQL Client Programs

4.1. What are the Client Programs?

A number of client programs are packaged with the MySQL RDMS. There are administrative programs, utilities to assist in backing up data, utilities for repairing tables, and numerous other tools.

4.2. The MySQL Client, `mysql`

MySQL Contributor. This section was contributed by MySQL staff. For more information see http://mysql.com.

The most important client tool is the `mysql` program, usually referred to as the MySQL client. This is a command-line program for interacting with a MySQL server. Use `mysql` to:

- create a database or database objects
- query a database
- perform administration tasks

Most often databases are integrated into applications and most of the interaction with the database server happens through the program's user interface. However, this is usually not a convenient way to create databases or database objects. Nor is it a convenient way of creating users or changing their privileges. Likewise, it is often helpful to test any SQL statements that are issued by a program. An easy way to perform all these tasks is to use the MySQL client.

However, we won't be discussing SQL commands here. The purpose of this section is to explain the options and commands most commonly used with the MySQL client. Using these options and commands is an essential part of mastering MySQL. Options passed to the `mysql` command make connecting to a MySQL server possible and also change the way that the `mysql` client program behaves. Some of these options are absolutely essential, some are nice to know, and others are used infrequently. We will deal with the essential and nice-to-know options.

4.2.1. Essential Options

The essential options are as follows:

- `--host=hostname, -h hostname` – the host to connect to
- `--password=[user_password], -p[user_password]` – password for connecting to the server
- `--port=port_number, -P port_number` – the TCP/IP port number
- `--user=user_name, -u user_name` – the MySQL username to use when connecting to the server

> **Note**
>
> The commonly used options of any MySQL program typically have a long and a short form. The long form is always a full word preceded by two dashes and followed by an equals sign and a value, if a value is required. The short form is a single letter, upper or lower case, preceded by one dash and followed by a space and a value if necessary. (The short form of the password option is the only exception to this rule and will be dealt with shortly.) When an option is first introduced, using the long form is helpful for reasons of clarity. Afterwards, the short form is preferred for the sake of brevity.

A MySQL server runs on a specific host and listens on a specific port. The MySQL client can connect to the server using TCP/IP and for this reason you must provide `--host` and `--port` options.

Since the `mysql` client program gives access to a specific MySQL server, you must have credentials on that server; you must provide values for the `--user` and `--password` options.

Any of the utilities that require a connection to the MySQL server, will have `host, port, user`, and `password` options. The syntax for using these options is the same for all the various MySQL programs so you must be familiar with these options.

To open a `mysql` console window and communicate with a MySQL server, type the following:

```
shell> mysql --user=user_name --password=user_password --host=localhost --port=3306
```

> **Note**
>
> If you have only just installed MySQL and have not yet defined any MySQL users then specify the default, `root`, as the user name. There is no password for this default user. For information on creating additional users see <xref>.

The same effect can be achieved using the short forms of the above options. Starting `mysql` using short forms is done as follows:

```
shell> mysql -u user_name -puser_password -h localhost -P 3306
```

Specifying a password immediately after the `-p` option is not a requirement but if you do so no space is permitted between the option `-p` and the password. Omitting the password value following the password option is considered more secure. If you do this, you are prompted for a password and asterisks replace any letters typed.

Fortunately, both the `host` and `port` options have default values so you need not supply them every time you connect to a MySQL server. The default value for the port is `3306`, and for the host, `localhost`. Most MySQL servers listen on port `3306` and typically you will connect to a server running on the same machine as the MySQL client.

If the server you wish to connect to is running on port `3306` on the same machine as the MySQL client then you need not specify either the port or the hostname. The `mysql` program will also check for the environment variable `USER`, if no user name is provided at the command line. To check the value of this variable under Windows go to the command line and type:

```
shell> echo %USER%
```

Under Linux or Mac OS X type:

```
shell> echo $USER
```

> **Warning**
>
> On any operating system (OS) the value of the variable `USER` is typically the name of the current OS user — there is no requirement that there also be a MySQL user with the same name, though this may often in fact be the case.

If this user name is a valid user name for the MySQL server then you need not specify the `--user` option in order to connect. Connecting to a MySQL server can be as simple as:

```
shell> mysql -p
```

With the use of a configuration file, even this option need not be specified at the command line. If you typically start up `mysql` using a number of options, then storing these options in a configuration file is a good way to simplify things. Configuration files are discussed in detail in <xref>.

4.2.2. Other Options

The following list of options are useful to know and can appreciably improve your efficiency when using `mysql`.

- `--help, -?` – show the available options and their default values and close the MySQL client
- `--auto-rehash` – enable table name and column name completion
- `--database-name=db_name, -D db_name, db_name` – the database to use on start up
- `--execute=statement, -e statement` – execute the specified statement and close the MySQL client
- `--html, -H` – output in HTML format
- `--no-tee` – do not copy output to file
- `--prompt=opt_string` – configure the prompt

- `--tee=outfile, -T outfile` – copy output to the specified file
- `--xml, -X` – output in XML format

The `--help` option is especially useful should you forget what options are available. Execute the `mysql` command with this option in order to display all available options and their default values. The interactive MySQL shell does not open when you use this option. Most of the MySQL programs have `--help` as an option.

The `--auto-rehash` option is on by default. It enables automatic completion of table and column names, in the way that most Unix command shells complete file names. Unfortunately, this option only works on Unix operating systems. For performance purposes you can turn this option off by specifying `--skip-auto-rehash`.

If you wish to start the MySQL client using a specific database, use the `--database=dbname` option. In addition to using the short form, `-D dbname`, you can also start the MySQL client using a specific database simply by specifying the database name at the command line. This option is equivalent to opening the MySQL client and then issuing a `use dbname` command.

To execute a single SQL statement and then exit the client shell, use the `--execute=statement` option. For example, the following command shows all the records in a specific table:

```
shell> mysql -u user_name -p --execute="SELECT * FROM dbname.table_name;"
```

The "`;`" terminating the `SELECT` statement is optional.

The `--html` and `--xml` options format all output as HTML or XML. This can be especially useful and time-saving if you need to dump the contents of a table in HTML or XML format. To get maximum benefit from these options you need to be familiar with the `--tee=file_name` option — an option that copies all the output of `mysql` to a text file. To create an HTML file of all statements issue the following command:

```
shell> mysql -u user_name -p --tee=outfile.html --html
```

> **Note**
>
> There is no short form for the `--tee=file_name` option.

The `--tee=file_name` option is also especially useful if you want to keep a record of the SQL statements that you have issued. This is also an excellent way to begin creating a script file. Script files are dealt with in detail in <xref>.

The `--prompt` option allows you to customized the prompt that the MySQL console displays. The prompt can be configured in a variety of ways; to show the current date and time, to display the default database, and the current server version, for example. This topic will be dealt with in detail in <xref>.

As noted earlier, only selected `mysql` options are discussed here. For a complete list see Section C.1, "`mysql` Options" or, at the command line simply type **`mysql -?`**.

4.2.3. The `mysql` Commands

The `mysql` client is an interactive shell. Once you have opened it you can use the `mysql` commands. A shortlist of the most useful commands follows:

- `help [argument], \? [argument], ? [argument]` – display the available commands, provide assistance with SQL
- `exit, quit, \q` – exit the MySQL shell
- `ego, \G` – send the statement to the server and display the result vertically.
- `notee, \t` – stop capturing output to file
- `source, \. file_name` – run a script file
- `tee file_name, \T file_name` – copy output to the specified file
- `use dbname, \u dbname` –

The long forms of commands can be issued by typing the command name with or without a ";". For example, the commands `help;` and `help` produce the same output.

To quit the MySQL shell use one of the forms of the `quit` command.

The `ego` command can be especially useful when issuing a select statement that returns one record with numerous columns. For example issuing the select statement, `SELECT * FROM mysql.user WHERE User=`root`\G`, produces much more readable output than the same statement terminated by a ";".

To execute a script file use the `source` command. You can also execute script commands by redirecting a file to the `mysql` command. This is done on all operating systems by using the redirection operator like so; **`mysql -u user_name -p < script.sql`**. Using script files is especially useful when you have repetitive tasks to perform. Script files are discussed in detail in <xref>.

An alternative to the `source` command is to redirect a script file to the `mysql` command from the command prompt. You can do this in the following way: ...

To avoid having to fully qualify a table name by preceding it with the database name, use the `use dbname` command. This command makes the specified database the default database.

Some of the commands are identical to the options shown in Section 4.2.2, "Other Options". For example issuing the `tee outfile.txt` command is identical to using the start-up option `--tee outfile.txt`. Being able to redirect output to a file after starting up the MySQL client can be very convenient when you only want to capture some and not all output to file. When you no longer wish to capture output, issue the `no-tee` command.

On the other hand, the `help` command, though it shares the same name as the `--help` option, when issued without an argument, outputs a list of all the commands but no options. For a complete list of all the available commands see Section C.2, "`mysql` Commands", or from the `mysql` shell, issue the command **`?`**.

However, the `help` command does a lot more than list available commands. For this reason the next section is devoted entirely to this command.

4.2.4. Using the `help` Command

It is very easy to overlook the usefulness of the `help` command. Firstly, it is easily confused with the `--help` *option*. Secondly, you may mistakenly believe that it only displays a list of the `mysql` commands.

To see just how helpful this command can be, from the `mysql` shell, issue the command `help` followed by the argument, `contents`. Doing this results in the following display:

```
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the following
categories:
Account Management
Administration
Data Definition
Data Manipulation
Data Types
Functions
Functions and Modifiers for Use with GROUP BY
Geographic Features
Language Structure
Storage Engines
Stored Routines
Table Maintenance
Transactions
Triggers
```

Now try issuing the **`? Functions`** command. You should see a listing of all the function categories. To see the date and time functions type **`? Date and Time Functions`**. This displays the names of all functions in this category.

To drill down even further, specify a function name in the following way; **`? DATE_FORMAT`**. Issuing this command displays the function prototype and gives examples of how this function is used. This is very useful given the many and various format specifiers that can be used with this function.

The `help topic` command only works if the `help_*` tables in have been installed in the `mysql` database. Most recent binary releases come with these tables installed but if you find that they are missing, go to http://dev.mysql.com/doc/ and locate the `MySQL Help Tables` section. Find the help file for the MySQL server version 5.0 and download it. Decompress it and install it in the following way:

```
shell> mysql -u root -p mysql < file_name
```

The `help` command is especially useful if you are new to MySQL but even experienced users will find it helpful on many occasions.

4.3. The `mysqladmin` Client Program

MySQL Contributor. This section was contributed by MySQL staff. For more information see http://mysql.com.

The `mysqladmin` program is a client utility for administering a MySQL server. As is the case with most MySQL utilities, there are often alternatives to using `mysqladmin`. The GUI Tool, MySQL Administrator, for example, can do most of the tasks performed by `mysqladmin` and is a very useful administrative tool especially if you are new to MySQL. This application is examined in detail in <xref>. Likewise, many of the capabilities of `mysqladmin` are also available when using the MySQL client program.

However, the MySQL server may be running on a machine that does not have a GUI; for instance, most web servers would fall into this category. `mysqladmin` offers a convenient alternative that allows you to perform common tasks quickly from the command line without starting up a GUI application or the MySQL client. This chapter shows how to use `mysqladmin` for these kinds of tasks. It is not meant as a definite treatment of `mysqladmin`; for complete coverage of this utility see http://dev.mysql.com/doc/refman5.0/en/mysqladmin.html. The more advanced commands and options of `mysqladmin` will be discussed in more detail in <xref>.

4.3.1. Options and Commands

Unlike other utilities, `mysqladmin` supports both commands and options. This section identifies the most commonly used options and commands and briefly describes each one. Examples of using these options and commands are given in subsequent sections.

The essential commands are as follows:

- `create` *`db_name`* – create a database
- `drop` *`db_name`* – drop a database
- `ping` – check that the MySQL server is running
- `shutdown` – bring down the MySQL server

Since the `mysqladmin` utility gives access to a specific MySQL server, you must have credentials on that server; you must explicitly or implicitly provide a `--user` and `--password`. Likewise you must provide `--host` and `--port` options. In this respect, `mysqladmin` does not differ from the MySQL client program, `mysql`.

Essential commands are shown above; the essential `mysqladmin` *options* are as follows:

- `--help, -?` – display the help message and exit
- `--force, -f` – do not ask for confirmation when dropping a database

Find a complete list of all the options see Section C.3, “`mysqladmin` Options”.

4.3.2. Using `mysqladmin`

This section is concerned with the `mysqladmin` commands and options listed in the previous section, discussing them in more detail with the exception of the `--help` option since this option functions in exactly the same way for all the MySQL programs.

4.3.2.1. Shutting Down the Server

The most common use for `mysqladmin` is to shut down the MySQL server. This is done in the following way:

```
shell> mysqladmin shutdown -u user_name -p
```

Note

This is a command rather than an option so do not precede `shutdown` with "`--`". The reasons for shutting down a MySQL server are various:

- you wish to start the server with different options
- you have changed the configuration file
- you wish to back up data

This command is used extensively in <xref>. In that section we examine the configuration file (`my.ini` under Windows, `my.cnf` under Unix and Mac OS X). In order for configuration changes to have effect the server must be stopped and re-started.

4.3.2.2. Creating and Dropping Databases

Instead of issuing an SQL command to create a database, you can create one using `mysqladmin` in the following way:

```
shell> mysqladmin create db_name -u user_name -p
```

You can just as easily remove a database using the `drop db_name` command. To avoid confirming your action use this command with the `-f` option.

4.3.2.3. Checking that the Server is Running

The `ping` command is an easy way to determine if your MySQL server is running. To test this command, do the following:

```
shell> mysqladmin ping -u user_name -p
```

This should result in the message, `mysqld is alive`, confirming that your MySQL server is running.

If you are working in a development environment, shut down the server using `mysqladmin` and the `shutdown` command. Try connecting to the server using the MySQL client `mysql`. On Windows you should see the error message:

```
ERROR 2003 (HY000) Can't connect to MySQL server on 'localhost' (10061)
```

The message on Unix systems is slightly different:

```
ERROR 2002 (HY000): Can't connect to local MySQL server through socket
'/var/lib/mysql/mysql.sock' (2)
```

Familiarize yourself with these error messages because you will see them again — but probably by accident and not, as in this case, by design. In most situations, these errors indicate that the MySQL server is not running.

Execute `mysqladmin` once more with the `ping` command, to see the message displayed when the server is down. You should see:

```
mysqladmin: connect to server at 'localhost' failed
error: Can't connect to MySQL server  ...
Check that mysqld is running
```

To restart the server on Unix [and Mac] systems issue the command, `mysqld_safe` and on Windows, `mysqld` (?). For more information ...

Chapter 5. Basic Administration

5.1. Introduction

5.2. Using MySQL Administrator

5.3. Starting and Stopping the MySQL Server

5.4. Administering Users

Chapter 6. MySQL Server Programs

6.1. The MySQL Server

6.1.1. Essential Server Options

The essential options and system variables are as follows:

- `--bind-address=IP_address`
- `--date_format=format_string`
- `--datetime_format=format_string`
- `--default-storage-engine=engine_type`
- `--local_infile=[0|1]`
- `--sql_mode=mode`
- `--skip_networking` – allow local connections only
- `--skip_show_databases`

To see the default format that MySQL uses for dates issue the statement **`SELECT NOW();`** from the MySQL client. You should see output like the following:

```
+---------------------+
| NOW()               |
+---------------------+
| 2007-06-27 16:19:37 |
+---------------------+
```

If you don't find this date/time format suitable, you can change it by using the `--date-format` and `--datetime-format` options. For example, to change the date format to a two digit month followed by a two digit day of the month and a four digit year separated by a forward slash use the following option: `--date-format=`%m/%d/%Y``. For all the legal formats see Appendix A, *Date Format Specifiers Table*.

The `--skip-networking` option allows only local (non-TCP) connections. On Unix, local connections use a Unix socket file and on Windows, they use a named pipe or shared memory. For a development server you will most likely not want to use this option. However, some distributions come with this option activated so it is useful to know about it so that you can disable it.

Likewise with the `--bind-address`. This option binds the MySQL server to a specific IP address, typically `127.0.0.1` effectively disabling access from a remote location. It is mentioned here so that you know how to disable it.

6.2. The `my.cnf/my.ini` File

In the chapter on MySQL client programs we hinted that there was a better way to make use of command options than specifying them at the command line. Entering options from the command line can be both tedious and error-prone. The better way is to store options in a configuration file so that the need not be specified each time you use a specific program.

Under Windows this file is usually found in the `C:\Program Files\MySQL\MySQL Server 5.0` directory and on Linux ...

Options in `/etc/my.cnf` and `$MYSQL_HOME/my.cnf` are processed before command-line options, so it is recommended that you put a `--user` option in the configuration file and specify a value other than root. The option in the configuration file is found before any other `--user` options, which ensures that the server runs as a user other than root, and that a warning results if any other `--user` option is found.

Changing the configuration file requires stopping the server. The easiest way to start and stop the MySQL server under Windows is from the Microsoft Management Console Services window. To open this window, go to the `Control Panel` and find `Administrative Tools`. Double click this icon and then choose `Services`. Find the MySQL entry, select it, and stop the service. Also

menu item ...

To stop the MySQL server under Linux or Mac OS X use `mysqladmin` with the `shutdown` command. You may also need to specify other options such as `--user` and `--password`.

You are now ready to make changes to the configuration file.

6.3. Changing Your Configuration File

Part II. Using MySQL

Table of Contents

Chapter 7. Introduction to Using MySQL

7.1. Variations Between Operating Systems

MySQL functions slightly differently on different operating systems. For the most part these differences can be ignored but you should be aware of the following items:

- sockets
- auto-completion
- file separators
- etc

7.2. Working From the Command Line

7.3. Using Query Browser

7.4. Sample Schema

Chapter 8. Populating a Schema

8.1. Populating a Schema with a SQL File

Chapter 9. Querying Data

9.1. Using SELECT

9.1.1. The SELECT Clause

9.2. Functions in Queries

9.3. Aggregate Functions in Queries

9.4. Using User Variables

9.5. Subqueries

Chapter 10. Data Types

10.1. <Data Type>

Chapter 11. Operators

11.1. <Operator Type>

Chapter 12. Functions

12.1. <Function Type>

Chapter 13. Modifying Data

13.1. Adding Data with the INSERT Statement

Part III. Advanced MySQL Usage

Table of Contents

Chapter 14. Creating Schemas and Tables

14.1. Designing a Schema

Chapter 15. Indexing Data

15.1. Introduction to Indexing

Note: Experienced users may wish to skip this section.

An index in a relational database serves much the same purpose as an index in a book: books are not (typically) organized alphabetically by subject, but are instead organized into logical chapters and parts. This works well until you need to find all references to a specific subject.

When you find yourself in such a situation you look at the index. In the index, the contents of the book are listed in alphabetical order, with a page number displayed for each subject.

Rather than go through the book page by page, you can look a subject up in the index, then move to the page number in question and scan the page for the subject you are interested in.

In a relational database, rows are not stored in a table in any particular order. An index creates a sorted version of a column (or set of columns) from a table that can be searched more efficiently than the table itself.

Each entry in the index stores either the row itself (in the case of InnoDB), or a pointer to the row's location in the data file (similar to a page number in a book index).

Effective indexing can greatly improve the performance of `SELECT` queries. When a specific column value is searched for, and an appropriate index is *not* available, the MySQL server must read every row of the table looking for matching rows. When an appropriate index is available, the MySQL server can look in the index for the appropriate values. If you are querying only columns that appear in the index, the MySQL server skips reading the table itself and return data directly from the index.

This chapter covers the following topics:

- Types of Indexes
  - INDEX
  - PRIMARY KEY
  - UNIQUE INDEX
  - SPATIAL INDEX
  - FULLTEXT INDEX
- Choosing Columns to Index
- Displaying Index Information
  - Using `SHOW`
  - Using the `INFORMATION_SCHEMA`
  - Using MySQL Administrator
- Creating Indexes
  - Using `CREATE TABLE`
  - Using `CREATE INDEX`
  - Using `ALTER TABLE`
- Composite Indexes
- Dropping Indexes
- Using `FULLTEXT`

- Optimizing Indexes

15.2. Types of Indexes

There are five different index types available for use in MySQL:

- `INDEX`: The standard table index. This type of index speeds up queries, but does not enforce uniqueness or uniquely identify the row.
- `PRIMARY`: This index is placed on the `PRIMARY KEY` of the table. Each entry in the index must be unique and the column(s) of this index are considered to be the unique identifier of the row within the table.
- `UNIQUE`: This index enforces that the column(s) being indexed are unique within the table, but is not considered to be the unique row identifier for the table.
- `FULLTEXT`: This index is used to increase the efficiency of querying natural language information in `CHAR`, `VARCHAR`, and `TEXT` columns. The `FULLTEXT` index is available for the `MyISAM` storage engine only.
- `SPATIAL`: This index is used with `GIS` applications to improve the performance of functions such as `MBRContains()` or `MBRWithin()`. The `SPATIAL` index is available for the `MyISAM` storage engine only.

Your choice of index usually depends on whether a given column should contain unique values: if a column contains unique values and those values are considered to uniquely identify a row (such as `ISBN`, `UPC`, or `SSN` values), choose a `PRIMARY` index. If your values are unique, but are not considered to uniquely identify the row, choose a `UNIQUE` index. If you wish to improve the performance of queries that match values in a column, use a regular `INDEX`. Only one `PRIMARY` index is allowed per table, all other index types can occur multiple times in a single table.

In special cases you may need to use `FULLTEXT` and `SPATIAL` indexes. If you are searching columns that contain natural language, you should use a `FULLTEXT` index. An example of natural language would be the titles and bodies of a collection of articles, as opposed to simple textual data such as person or place names, which should be indexed using a regular `INDEX`. The `SPATIAL` index is for use with `GIS` data only.

15.3. Choosing Columns to Index

To use indexes effectively, it is necessary to identify those queries that are executing slowly and are not using indexes, or columns which contain unique values. Slow queries can be identified by directly reviewing the queries you use or by using the MySQL Slow Query Log.

15.3.1. Reviewing Queries for Index Usage

To identify queries that make good candidates for indexing, look at the `WHERE`, `GROUP BY`, and `ORDER BY` clauses of your queries.

For example, look at the following query performed against the `sakila` sample database:

```
SELECT last_name, first_name FROM actor WHERE last_name = 'Walken'
```

In this case the WHERE clause contains a reference to the `last_name` column of the `actor` table.

Here is another example:

```
SELECT film.title, COUNT(inventory.inventory_id) AS Stock
FROM film, inventory
WHERE film.film_id = inventory.film_id
  AND inventory.store_id = 1
GROUP BY film.film_id
```

In this case the `film_id` and `store_id` columns are candidates for indexing.

Once you have identified candidate columns, you need to evaluate how often the column is involved in a query: the most often a column is referenced in your various queries, the stronger a candidate it becomes for indexing.

Each index you add to a table will have a negative effect on the performance of `INSERT`, `UPDATE` and `DELETE` queries because not only does the row data need to be changed, the index information must also be updated for each affected index. Over-indexing can lead to performance loss.

Once you have identified your candidate columns you can check whether they are already indexed by displaying the existing indexes on a table.

15.3.2. Identifying Slow Queries with the Slow Query Log

If you need to identify slow queries on a production MySQL server you may benefit from using the MySQL Slow Query Log. When the MySQL server is started with the Slow Query Log enabled, it writes all queries that take longer than a configurable number of seconds to a log file. The queries in the Slow Query Log can be further examined and optimized. For more information see Section 25.3, “The Slow Query Log”

15.4. Displaying Table Indexes

Before creating a new index, it is important to know what indexes currently exist for a given table. Index information can be retrieved using either the `SHOW` syntax or through use of the `INFORMATION_SCHEMA`. `INFORMATION_SCHEMA` is a standard method for accessing information, while `SHOW` is an extension of the SQL standard. Index information can also be browsed using the MySQL GUI tools.

15.4.1. Displaying Table Indexes Using the SHOW Command

The `SHOW` command can be used within the active schema to display the index information for a table:

```
mysql> USE sakila;
Database changed

mysql> SHOW INDEX FROM film\G
*************************** 1. row ***************************
       Table: film
  Non_unique: 0
    Key_name: PRIMARY
Seq_in_index: 1
 Column_name: film_id
   Collation: A
 Cardinality: 2
    Sub_part: NULL
      Packed: NULL
        Null:
  Index_type: BTREE
     Comment:
*************************** 2. row ***************************
       Table: film
  Non_unique: 1
    Key_name: Title_Description_Fulltext
Seq_in_index: 1
 Column_name: title
   Collation: NULL
 Cardinality: NULL
    Sub_part: NULL
      Packed: NULL
        Null:
  Index_type: FULLTEXT
     Comment:
*************************** 3. row ***************************
       Table: film
  Non_unique: 1
    Key_name: Title_Description_Fulltext
Seq_in_index: 2
 Column_name: description
   Collation: NULL
 Cardinality: NULL
    Sub_part: NULL
      Packed: NULL
        Null: YES
  Index_type: FULLTEXT
     Comment:
3 rows in set (0.00 sec)
```

The output of the `SHOW` command shows that there are two indexes on the `film` table: `PRIMARY` and `Title_Description_Fulltext` (as seen from the `Key_name` value).

The columns being indexed are listed in the `Column_name` field. In this case there are indexes on the `film_id`, `title`, and `description` columns.

The `title` and `description` columns form two parts of the `Title_Description_Fulltext` index, with the `title` column appearing before the `description` column in the index, according to the `Seq_in_index` field.

You can determine whether or not an index enforces uniqueness by the `Non_unique` field: `0` indicates that the index enforces uniqueness, `1` indicates that the index does not enforce uniqueness.

15.4.2. Displaying Table Indexes Using the INFORMATION_SCHEMA

As an alternative to the `SHOW` command, users can query the `STATISTICS` table of the `INFORMATION_SCHEMA`.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.STATISTICS
    -> WHERE TABLE_SCHEMA = 'sakila'
    ->   AND table_name = 'film'\G
*************************** 1. row ***************************
TABLE_CATALOG: NULL
 TABLE_SCHEMA: sakila
   TABLE_NAME: film
   NON_UNIQUE: 0
 INDEX_SCHEMA: sakila
   INDEX_NAME: PRIMARY
 SEQ_IN_INDEX: 1
  COLUMN_NAME: film_id
    COLLATION: A
  CARDINALITY: 2
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT:
*************************** 2. row ***************************
TABLE_CATALOG: NULL
 TABLE_SCHEMA: sakila
   TABLE_NAME: film
   NON_UNIQUE: 1
 INDEX_SCHEMA: sakila
   INDEX_NAME: Title_Description_Fulltext
 SEQ_IN_INDEX: 1
  COLUMN_NAME: title
    COLLATION: NULL
  CARDINALITY: NULL
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: FULLTEXT
      COMMENT:
*************************** 3. row ***************************
TABLE_CATALOG: NULL
 TABLE_SCHEMA: sakila
   TABLE_NAME: film
   NON_UNIQUE: 1
 INDEX_SCHEMA: sakila
   INDEX_NAME: Title_Description_Fulltext
 SEQ_IN_INDEX: 2
  COLUMN_NAME: description
    COLLATION: NULL
  CARDINALITY: NULL
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE: YES
   INDEX_TYPE: FULLTEXT
      COMMENT:
3 rows in set (0.00 sec)
```

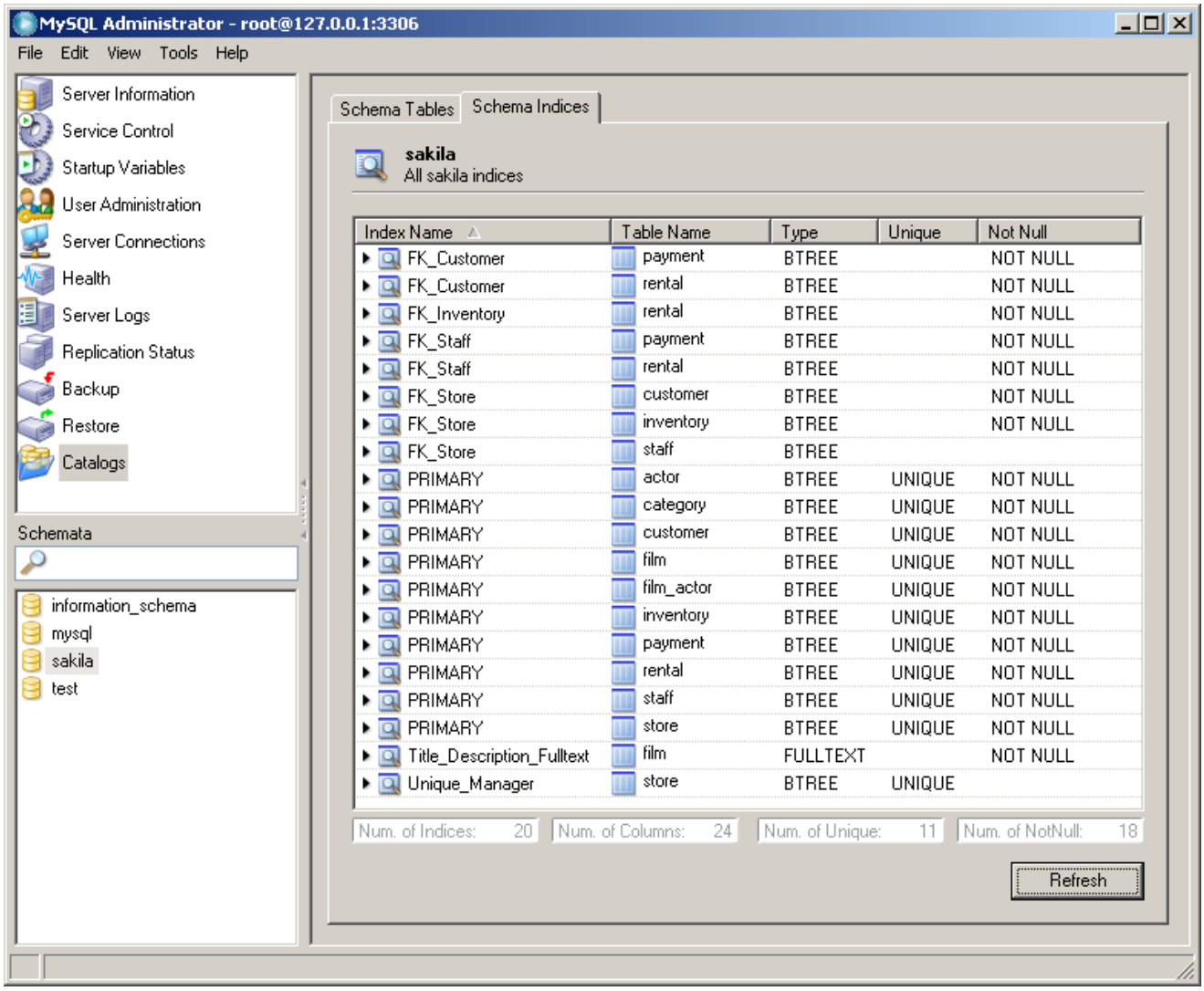
The output of a query to the `INFORMATION_SCHEMA` closely matches that of the `SHOW` statement. For information on interpreting the output of a query on the `INFORMATION_SCHEMA`, please see Section 15.4.1, “Displaying Table Indexes Using the SHOW Command”.

One advantage of the `INFORMATION_SCHEMA` is that you can view the index information of more than one table at a time by modifying the `TABLE_NAME` portion of the `WHERE` clause of your query.

15.4.3. Displaying Index Information Using MySQL Administrator

Users can also view index information using MySQL Administrator. To view index information, select the desired schema in the `Catalogs` screen, then select the `Schema Indices` tab:

Figure 15.1. Displaying index information with MySQL Administrator



The `Catalog` screen displays the index information for all tables in the selected schema. To view the columns that make up a given index, click the arrow icon to the left of the index name.

15.5. Creating Indexes

Indexes can be created using either a `CREATE TABLE`, `CREATE INDEX` or `ALTER TABLE` syntax. Before creating an index, you must know which table the index will be added to, which column(s) the index will apply to, the type of index you will create, and whether the index will enforce uniqueness.

15.5.1. Creating Indexes with the CREATE TABLE Statement

Indexes can be created during table creation by specifying the index information as part of the `CREATE TABLE` statement, specifying the index information either as part of a column information or at the end of the column definitions.

At minimum, the `PRIMARY KEY` and `UNIQUE` indexes should be specified at table creation to prevent duplicate key issues from occurring when the indexes added later (if you try to add such an index after your table is populated, you may have to remove duplicate rows manually before the indexes can be created).

For example, this is a simplified version of the CREATE TABLE statement for the `inventory` table:

```
CREATE TABLE inventory (
```

```
  inventory_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  film_id INT UNSIGNED NOT NULL,
  store_id INT UNSIGNED NOT NULL,
  INDEX store_id_index (store_id)
)
```

The primary key was created as part of the column creation line, while the index on the `store_id` column was created at the end of the `CREATE TABLE` statement.

It is necessary to specify index information at the end of the `CREATE TABLE` statement if the index contains more than one column. For example, in the `film_actor` table a single actor cannot appear more than once in the same film, so the primary key is a combination of the `film_id` and `actor_id` columns:

```
CREATE TABLE film_actor (
  actor_id INT UNSIGNED NOT NULL,
  film_id INT UNSIGNED NOT NULL,
  PRIMARY KEY  (film_id, actor_id)
)
```

15.5.2. Using the CREATE INDEX Syntax

To create a table with the `CREATE INDEX` syntax, you specify the type of index, a name for the index, the table to create the index on, and a list of the columns that form the index. `CREATE INDEX` is a non-standard alternative syntax to `ALTER TABLE`, described in the section that follows.

For example, to create an index on the `last_name` and `first_name` columns of the actor table, you would execute the following statement:

```
 CREATE INDEX last_first_name ON actor (last_name, first_name)
```

This next example creates an index that enforces uniqueness on the `manager` column of the `store`, ensuring no employee is so overworked that they have to manage two locations:

```
CREATE UNIQUE INDEX unique_manager ON store (manager)
```

The index types can be `INDEX`, `UNIQUE INDEX`, `SPATIAL INDEX`, and `FULLTEXT INDEX`.

15.5.3. Using ALTER TABLE to Create Indexes

You can use the `ALTER TABLE` statement to create indexes on existing tables. The benefit of `ALTER TABLE` is that it can be used to create multiple indexes in a single statement, which can speed index creation when multiple indexes are needed.

For example, to create an index on `last_name` and `first_name` columns of the actor table, you would execute the following statement:

```
ALTER TABLE actor ADD INDEX last_first_name (last_name, first_name)
```

This example creates an index that enforces uniqueness on the `manager` column of the `store`:

```
ALTER TABLE actor ADD UNIQUE unique_manager (manager)
```

Here is an example of creating a primary key on the `staff` table:

```
ALTER TABLE staff ADD PRIMARY KEY (staff_id)
```

Primary keys can only be created with `CREATE TABLE` and `ALTER TABLE` statements, there is no `CREATE INDEX` equivalent for primary keys.

Multiple indexes can be created with a single `ALTER TABLE` statement by separating the `ADD` statements with commas:

```
ALTER TABLE actor ADD UNIQUE unique_manager (manager), ADD PRIMARY KEY (staff_id)
```

```
```

Other index types you can add include `ADD SPATIAL` and `ADD FULLTEXT`.

15.5.4. Indexing the Prefix of a Column

It is possible to index only a prefix of a `VARCHAR`, `CHAR` or `TEXT` column by placing the size of the prefix (in characters) within brackets after the column name:

```
CREATE INDEX lname ON actor (last_name(5))
```

Indexing the prefix of a column decreases the size of the index on disk compared to indexing the entire column, which in turn increases the performance of the index. An index can prefix up to 1000 bytes of a column in `MyISAM`, 767 bytes in `InnoDB`, and 255 bytes for all other storage engines.

One way to find the proper prefix size for a column is to perform the following `SELECT` query:

```
SELECT COUNT(DISTINCT column_name) AS distinct_rows,
COUNT(DISTINCT(LEFT(column_name, N))) AS prefix_distinct
FROM table_name
```

Start with a *N* value of `3` and increase the size of *N* until the value of `prefix_distinct` nears that of `distinct_rows`.

15.5.4.1. Limitations of Index Prefixing

When indexing a `TEXT` or a `BLOB` column you must specify a prefix size, so the use of prefixes is optional for `CHAR`, `VARCHAR`, `BINARY`, and `VARBINARY` columns only.

A prefixed index `lname` as described in Section 15.5.4, “Indexing the Prefix of a Column” decreases the size of the index file and performs reasonably well when doing lookups. For example:

```
SELECT * FROM actor WHERE last_name = 'Depp';
```

However, prefixed indexes aren't very useful even for looking up rows if values have low cardinality in the prefixes. For example, if you prefix the first three characters of `abcd`, `abce`, `abcf`, and `abcg`, the prefix values are identical and do not distiguish rows.

Further, prefixed indexes are sometimes ignored when used for the following operations:

- `GROUP BY`
- `DISTINCT`
- `ORDER BY`

Additionally, if two tables are joined on columns that use prefix indexing, the index may be ignored and result in a full table scan. The optimizer is more often able to make use of full keys than prefix key values, so, in general it is safer to user a full key, especially if you are not sure exactly how your tables will be queried. If optimization is a major concern use an `EXPLAIN` statement to determine whether prefixed indexes are being used.

15.6. Creating and Using Composite Indexes

When executing a `SELECT` query, the MySQL server typically uses only one index per table involved in the query. If the `WHERE` clause of the query references more than one column, a single-column index may be less than optimal. For example, say you were executing the following query:

```
SELECT actor_id
FROM actor
WHERE last_name = 'Johnson'
  AND first_name = 'Robert'
```

If the table had an index on the `last_name` column, the index could be used to narrow the table down to all actors with the last name `Johnson`, but MySQL would still have to scan all the matched rows to find actors with the first name `Robert`.

By using a composite index, or an index on multiple columns, the preceding query could be fully optimized. Here is an example of a composite index on the `actor` table:

```
CREATE INDEX last_first_name ON actor (last_name, first_name)
```

With such a composite index, MySQL can first find the last name `Johnson` in the table, then search for the first name `Robert` in the matching index entries.

Composite indexes can also be partially used when the columns in the `WHERE` clause of a query appear in the left-most part of the composite index. For instance, the following query would make use of the composite index we have created:

```
SELECT first_name
FROM actor
WHERE last_name = 'Johnson'
```

However, the following query would not make use of our composite index:

```
SELECT last_name
FROM actor
WHERE first_name = 'Robert'
```

The second example does not make use of the composite index because the `first_name` column is not the left-most part of the index columns.

This rule applies no matter how many parts a composite index has; if you have an index on (`columnA`, `columnB`, `columnC`, `columnD`), the index will be used on queries that contain the following columns in the `WHERE` clause: (`columnA`), (`columnA`, `columnB`),(`columnA`, `columnB`, `columnC`), and (`columnA`, `columnB`, `columnC`, `columnD`). You would not be able to create any queries without `columnA` and expect the composite index to be used.

For a more detailed description on when an index will be used, see the section titled How MySQL Uses Indexes in the MySQL Reference Manual.

15.7. Dropping Indexes

Existing indexes can be dropped using either a `DROP INDEX` or `ALTER TABLE` syntax:

```
DROP INDEX index_name ON table_name
ALTER TABLE table_name DROP PRIMARY KEY
ALTER TABLE table_name DROP INDEX index_name
```

You can drop multiple indexes in a single `ALTER TABLE` statement by separating them with commas:

```
 ALTER TABLE actor DROP PRIMARY KEY, DROP INDEX last_first_name
```

15.8. Using FULLTEXT Indexes

While regular indexes are effective for many purposes, they are not effective for columns that contain natural language. An index can be used for single word `CHAR` and `VARCHAR` columns, but `FULLTEXT` indexes are designed for finding strings within larger natural language fields.

A `FULLTEXT` search takes a string and column list and searches the specified columns for the string, returning results ranked by relevancy.

The `FULLTEXT` index is available for the `MyISAM` storage engine only.

The syntax for creating a `FULLTEXT` index is listed in Section 15.5, “Creating Indexes”. Once the index is created, the `MATCH ... AGAINST` syntax can be used to perform `FULLTEXT` queries.

The MATCH clause indicates which columns are to be searched. The list of columns in the MATCH clause must be identical to the list of columns in the FULLTEXT index.

The AGAINST clause contains the string being searched for. The string in the AGAINST clause must be a constant string: you cannot use a user variable or search result in the AGAINST clause.

Here is an example of a basic FULLTEXT query that searches for movies in the film table that contain the word army in the title or description columns:

```
 SELECT title, description FROM film WHERE MATCH (title, description) AGAINST ('army')

*************************** 1. row ***************************
          title: ARMY FLINTSTONES
    description: A Boring Saga of a Database Administrator And a Womanizer who
must Battle a Waitress in Nigeria
1 row in set (0.00 sec)
```

Results from a query with MATCH ... AGAINST in the WHERE clause will always return in descending order based on relevancy.

Here is the same query performed with a LIKE clause instead:

```
 SELECT title, description FROM film WHERE title LIKE '%army%' OR description LIKE '%army%'

*************************** 1. row ***************************
          title: ARMY FLINTSTONES
    description: A Boring Saga of a Database Administrator And a Womanizer who
must Battle a Waitress in Nigeria
1 row in set (0.20 sec)
```

Note the performance improvement provided by the FULLTEXT index.

The MATCH ... AGAINST syntax can also provide relevancy ranking information:

```
 SELECT title, description, MATCH (title, description) AGAINST ('army') AS rank
 FROM film
 WHERE MATCH (title, description) AGAINST ('army')

*************************** 1. row ***************************
     title: ARMY FLINTSTONES
description: A Boring Saga of a Database Administrator And a Womanizer who
must Battle a Waitress in Nigeria
      rank: 6.1943987015493
1 row in set (0.00 sec)
```

The relevancy scores are based on the weighting of words within the individual rows. Words that occur rarely in the table are ranked higher than words that appear in a large percentage of the rows.

For more information on the FULLTEXT search engine, see the Fulltext Search section of the MySQL Reference Manual.

15.9. Using EXPLAIN to Optimize Indexing

Sometimes it is not easy to identify which columns of a table to index, even when you have identified the slow queries in your application. The EXPLAIN statement is designed to assist in the query optimization process by providing insight into how the MySQL optimizer handles a specific query.

To analyze a query, precede the query with the EXPLAIN keyword:

```
 EXPLAIN SELECT film.title FROM actor, film, film_actor
 WHERE actor.actor_id = film_actor.actor_id
   AND film.film_id = film_actor.film_id
   AND actor.last_name = 'Walken'\G

*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: actor
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 200
```

```
       Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: film_actor
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: sakila.actor.actor_id
         rows: 26
        Extra: Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: film
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: sakila.film_actor.film_id
         rows: 1
        Extra:
3 rows in set (0.00 sec)
```

`EXPLAIN` returns a row of information for each table used in the `SELECT` statement. The tables are listed in the output in the order that MySQL would read them while processing the query.

The main things to look out for are rows where the `key` column is `NULL`, where the `type` column is `range`, `index`, or `ALL`, or where the `Extra` column contains `Using filesort` or `Using temporary`. Such queries should be closely examined for proper index usage as they generally indicate that no index is being used.

For additional information on using the `EXPLAIN` statement, see the `EXPLAIN` section of the MySQL Reference Manual.

Chapter 16. MySQL Views

16.1. What is a Database View?

Chapter 17. Triggers

17.1. What Are Triggers?

Chapter 18. MySQL Stored Procedures

18.1. What is a Stored Procedure?

Chapter 19. MySQL Storage Engines

19.1. <type> Engine

Chapter 20. Optimization

20.1. Using EXPLAIN

20.2. Optimizing SELECT Statements

20.3. Optimizing Indexes

Part IV. Advanced MySQL Administration

Table of Contents

Chapter 21. Configuring MySQL

21.1. MySQL Option FIles

Chapter 22. Upgrading MySQL

22.1. Upgrading on <operating system>

Chapter 23. MySQL Security

23.1. Security Basics

23.2. Grants

23.3. Securing Default User Accounts

23.4. Advanced Utilities

- myisamchk – checking and repairing MyISAM tables

23.5. Advanced Features of Previously Mentioned Utilities

23.6. Best Practices

Chapter 24. Backing Up Data

24.1. Introduction

Say something about disaster recovery.

24.2. Using `mysqldump`

MySQL Contributor. This section was contributed by MySQL staff. For more information see http://mysql.com.

The `mysqldump` utility is a database back-up program capable of copying everything on a specific MySQL server — both the database objects and the data. It can also be used to copy a number of databases, one particular database, one or more tables from a specific database, or just specific records from one table. Any kind of data can be saved using this utility — even images stored as binary data.

The `mysqldump` utility creates a script file of SQL statements that recreate the database objects selected and it also creates `INSERT` statements to restore data. There are various other ways to back up MySQL databases or tables; using the `mysql` client program and SQL statements, copying the MySQL data directory, using binary logs, using MySQL Administrator, and also the Unix-specific utility, `mysqlhotcopy`. However, `mysqldump` is the most versatile and accessible tool for backing up tables and databases and it is available for all operating systems.

The reasons for creating back-up files vary:

- As replacements for existing files in the event of database. corruption
- To transfer files from a development server to a production. server
- To migrate to another file format.
- For reverse engineering. For using a file created by `mysqldump` to see reverse engineering a database.

This chapter shows how to use `mysqldump` for each of these tasks. This is not meant as a definite treatment of `mysqldump`; for complete coverage of this utility see http://dev.mysql.com/doc/refman5.0/en/mysqldump.html.

24.2.1. Options

This section identifies the most commonly used options and briefly describes each one. Examples of using these options are given in subsequent sections.

Since the `mysqldump` utility gives access to a specific MySQL server, you must have credentials on that server; you must explicitly or implicitly provide a `--user` and `--password`. Likewise you must provide `--host` and `--port` options. In this respect, `mysqldump` does not differ from the MySQL client program, `mysql`, or from other utilities such as `mysqladmin`.

Other common options are:

- `--all-databases, -A` – Dump all tables in all databases.
- `--databases, -B` – Specify this option and `mysqldump` regards all name arguments as database names
- `--fields-terminated-by` – Used in conjunction with the `--tab` option to specify a field terminator.
- `--no-create-db` – Used in conjunction with the `--all-databases` or the `--databases` option to suppress the `CREATE DATABASE` statement
- `--no-data` – Save database objects but not data.
- `--opt` – This option is shorthand for a group of options. See Section 24.2.1.1, “The `--opt` Group of Options”.
- `--skip-opt` – Turn off the `--opt` group of options.
- `--tab=path, -T path` – Create tab-separated data files in the named directory.

- `--tables` – Override the `--databases` option. `mysqldump` regards all name arguments following this option as table names.
- `--where='where_condition', -w 'where_condition'` – Only dump rows selected by the where condition.
- `--xml` – Dump output as XML.

The `--all-databases` option is used when you want to dump the entire contents of a server. On the other hand, the `--databases` option lets you specify particular databases to copy. Both of these options add a `CREATE DATABASE` statement to the dump file. To turn off this feature use the `--no-create-db` option. You can also choose not to save any data by using the `--no-data` option.

The `--tables` option makes it possible to use the `--databases` option and also specify which tables you would like to dump.

Use the `--tab` and `--fields-terminated-by` options, to dump a database in a variety of text formats. For XML format, use the `--xml` option. To select only specific rows from a table use the `--where` option.

For a complete list of all the available options see http://dev.mysql.com/doc/refman5.0/en/mysqldump.html.

24.2.1.1. The `--opt` Group of Options

The `--opt` option is on by default so you don't have to specify it. However, you do need to know what it does. Using `--opt` is shorthand for specifying `--add-locks, --add-drop-table, --create-options, --disable-keys, --extended-insert, --lock-tables, --quick,` and `--set-charset`. Find a brief description of these options in what follows.

- `--add-locks` – Lock tables before inserting data.
- `--add-drop-table` – Remove tables before recreating them.
- `--create-options` – Include all MySQL-specific table options in the `CREATE TABLE` statements.
- `--disable-keys` – Improve speed by disabling indexes before inserting data. (Applies only to MyISAM tables and only to non-unique indexes.)
- `--extended-insert` – Use multiple-row `INSERT` syntax that includes a `VALUES` list for each row.
- `--lock-tables` – Lock tables before dumping them.
- `--quick` – Retrieve rows from a table one row at a time, reducing demands on memory.
- `--set-charset` – Add `SET NAMES` *`default_character_set`* to the output.

Each of these options can be turned off individually by using the `--skip-`*`option-name`* syntax. For example, if you want to ensure that you recreate tables as the server default table type, you can turn off `--create-options` by specifying `--skip-create-options`. No engine or character set will be specified in the `CREATE TABLE` statement. Turn off `--extended-insert` by specifying the `--skip-extended-inserts` option. Doing this creates a separate `INSERT` statement for each row, making it much easier to remove individual `INSERT` statements.

24.2.2. Backing Up Data and Database Objects

To back up the contents of a server and create replacements for all existing databases invoke the `mysqldump` utility specifying your credentials and the `--all-databases` option. Using the option short forms, you can back up a server and redirect output to a file in the following way:

```
shell> mysqldump -u user_name -p -A > dump.sql
```

Using the short forms shown in the preceding listing is equivalent to using the `--user, --password,` and `--all-databases` options. Output is sent to a file using the redirection operator, "`>`". Since the `--host` option is not specified, it defaults to `localhost`. Likewise, `--port` will default to `3306`. Since no password is given at the command line, you will be prompted for one.

Specifying your password at the command line is allowed but note that you cannot leave a space between the option and your password; it must appear as `-p`*`password`*. If a space was allowed, the `-A` option in the preceding listing would be interpreted as the password.

If you wish to copy only specific databases, replace `-A` with the `--databases` option (or its short form, `-B`) followed by the names of the databases that you wish to back up. The file created by this command will contain only the databases specified.

Using `mysqldump` to back up specific databases or all the databases on a server is an easy way to create replacements in the event of lost data or database corruption.

24.2.2.1. Further Refining the Objects and Data Selected

If you wish to copy only one database, you do not need to use the `--databases` option. Simply specify your credentials and the database name in the following fashion:

```
shell> mysqldump -u user_name -p db_name > dump.sql
```

A specific database is selected by using the database name — no option is necessary. Output is again redirected to the file using the redirection operator.

Remember that the `--opt` group of options is on by default. (For a complete list of this group of options see Section 24.2.1.1, “The --opt Group of Options”.) To turn off any one of these options you can use the `--skip-option-name` option.

On the other hand, if you want to turn off most of the `--opt` options, it may be easier to specify `--skip-opt` and then list the options you wish to use.

> **Note**
>
> If you choose to do things this way, make sure that you specify `--skip-opt` first. If it is the last option specified, it will turn off any of the `--opt` group of options that precede it.

In some cases you may want to copy only selected tables from a database. This is done by naming the desired tables immediately following the database name. For example:

```
shell> mysqldump [options] db_name table1 table2
```

When dumping a specific table, the data selection can be further refined by adding a `--where` option in the following way:

```
shell> mysqldump [options] db_name table1 --where='field_name>1000'
```

When using the `--where` option only one table may be specified. The script file created will contain a `CREATE TABLE` statement for reconstructing the table and any data that meets the condition specified using the `--where` option.

> **Note**
>
> If the `--where` option contains spaces or characters special to your command interpreter, then you must enclose everything in the where condition in quotation marks.

Using a database name at the command line creates a copy of the tables and the data from the specified database. However, no database is created when this syntax is used.

To dump only one database and add a `CREATE DATABASE` statement, you must use the `--databases` option. An example using the short form of the `--databases` option follows:

```
shell> mysqldump -u user_name -p -B db_name > dump.sql
```

If you wish your dump file to contain a `CREATE DATABASE` statement and you only wish to dump selected tables use the `--tables` option as shown in the following:

```
shell> mysqldump -u user_name -p -B db_name --tables table1  > dump.sql
```

If you don't specify the `--tables` option, the `-B` option interprets each name as a database.

The next section examines how to restore databases from the script files created by `mysqldump`.

24.2.3. Restoring Database Dumps

Databases are restored by redirecting the script file to the `mysql` client program. If the script file was created using either the `-A` or `-B` options, restore the dumped files in the following way:

```
shell> mysql -u user_name -p < dump.sql
```

> **Warning**
>
> Using the `-A` or `-B` option with `mysqldump` creates a script that drops and recreates databases. Any data in existing databases will be lost. Furthermore, if you backed up all databases then the `mysql` database will be overwritten. Be sure that this is what you intend. For more information see ...

If you created your dump file without using the `-A` or `-B` options, then the database that you copy the tables to must already exist. Name that database when invoking `mysql`:

```
shell> mysql -u user_name -p db_name < dump.sql
```

In this case, you need not worry about overwriting an existing database, but you will overwrite any tables in the existing database that have the same names as tables in the back-up file if the file contains `DROP TABLE` and `CREATE TABLE` statements. To remove these statements from a dump file, create it using the `--skip-add-drop-table` and `--no-create-info` options.

If you are uploading a database dump file to a remote database then you will have to specify the `--host` option. If you don't have access to your MySQL server from a remote location, copy your script file to the server, log in using `ssh`, and then run `mysql`. If neither of these options is available to you, you may be able to upload and execute the script file using a program such as phpMyAdmin.

24.2.4. Exporting From MySQL

To use the data from a MySQL database in another application — a word processor or a spreadsheet, for example — you might want to export data in text format. The most common way of exporting a file in text format is by using the `--tab` or `-T` option and specifying the full pathname to the target directory:

```
shell> mysqldump -u user_name -p db_name -T /tmp
```

Dumping a database specifying this option creates a script file of each table's structure using the table name and the extension `sql` as the file name and a tab-separated file of each table's data using the table name and the extension `txt` as the file name. These files are created in the directory specified with the `-T` option. This directory must be writable and the user indicated by *`user_name`* must have the `FILE` privilege. For more information about the `FILE` privilege see ...

File permissions are not usually a problem on Windows systems but the file separator and spaces in file names can present difficulties. Use a forward slash to separate directories and, if a directory contains spaces, enclose the path in quotation marks, for example, “C:/Documents and Settings/peter/Desktop/”. Failure to include quotation marks results in the following error:

```
mysqldump: Got error: 1049; Unknown database 'and' when selecting the database
```

Despite its name, the `--tab` option can be used to created files with a field terminator other than the `tab` character. The field terminator is changed by using the `--fields-terminated-by` option. For example you can specify a “`,`” as the terminator in order to use a table in a spreadsheet program.

The `--tab` option is designed to extract data from one database only and cannot be used with the `--databases` option, or with the `--all-databases` option. Whenever it is used a database name must be one of the arguments to `mysqldump`.

To further refine the data selected, the `--tab` option can also be used to select data from one table only. This is done by naming the desired table after selecting the database. The `--tab` option can also be used with the `--where` option as shown in Section 24.2.2.1, “Further Refining the Objects and Data Selected”.

Often, when creating text files there is no need for the script file that creates the table structure — you're simply interested in exporting the data. In cases like this it would be nice to have an option to copy only data. No such option exists but we will see how to do this when we discuss `SELECT ... INTO OUTFILE`. For more information see ...

For an XML representation of the data and the database objects use the `--xml` option. This option creates an XML document in the following format:

```
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
  <database name="sakila">
    <table_structure name="actor">
      <field Field="actor_id" Type="smallint(5) unsigned" Null="NO" Key="PRI"
         Extra="auto_increment" />
      <field Field="first_name" Type="varchar(45)" Null="NO" Key=""
         Default="" Extra="" />
      <field Field="last_name" Type="varchar(45)" Null="NO" Key="MUL"
         Default="" Extra="" />
      <field Field="last_update" Type="timestamp" Null="NO" Key=""
         Default="CURRENT_TIMESTAMP" Extra="" />
      <key Table="actor" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
         Column_name="actor_id" Collation="A" Cardinality="0" Null=""
         Index_type="BTREE" Comment="" />
      <key Table="actor" Non_unique="1" Key_name="idx_actor_last_name"
         Seq_in_index="1" Column_name="last_name" Collation="A"
         Cardinality="0" Null="" Index_type="BTREE" Comment="" />
      <options Name="actor" Engine="InnoDB" Version="10" Row_format="Compact"
         Rows="0" Avg_row_length="0" Data_length="16384" Max_data_length="0"
         Index_length="16384" Data_free="0" Auto_increment="1"
         Create_time="2007-04-11 19:35:58" Collation="utf8_general_ci"
         Create_options="" Comment="InnoDB free: 10240 kB" />
    </table_structure>
    <table_data name="actor">
        [table data] ...
    </table_data>
    [more table definitions and data] ...
  </database>
</mysqldump>
```

Given the ease with which a database can be converted to XML you might wonder whether conversion to HTML is also possible. Unfortunately, there is no `mysqldump` option for creating HTML output. However, this can be done by starting `mysql` using the `--html` and `--tee` options. For instructions on doing this see Section 4.2.2, “Other Options”.

24.3. Replication

24.4. Other Options

Chapter 25. MySQL Log Files

25.1. What They Can Tell You

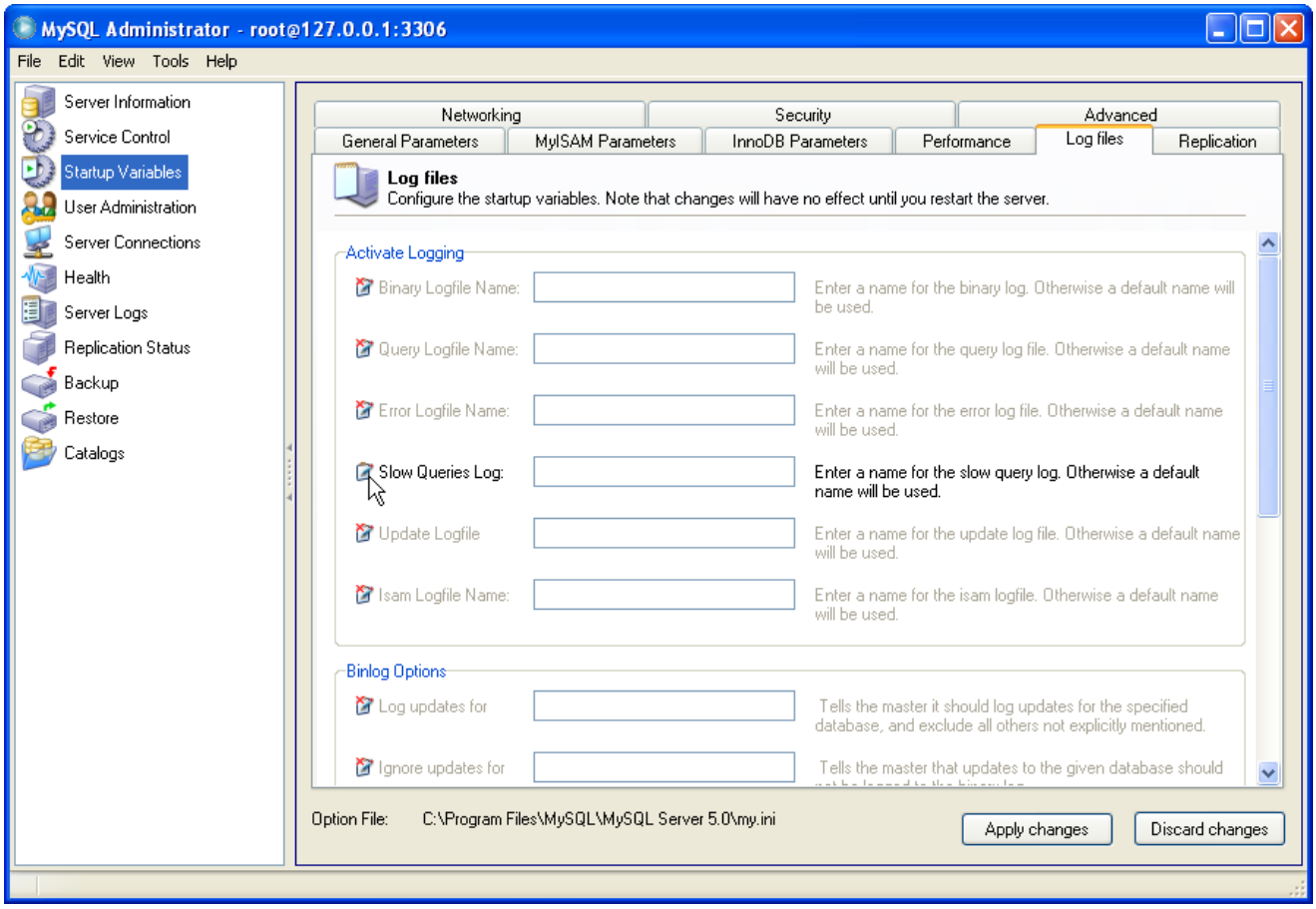
25.2. Error Log

25.3. The Slow Query Log

If you need to identify slow queries on a production MySQL server you may benefit from using the MySQL Slow Query Log. When the MySQL server is started with the `--log-slow-queries` option, it writes all queries that take longer than a configurable number of seconds to a log file. The queries in the Slow Query Log can be further examined and optimized.

The Slow Query Log can also be activated by adding the `log-slow-queries` directive to the `[mysqld]` section of your server option file, or through the MySQL Administrator:

Figure 25.1. Activating the Slow Query Log using MySQL Administrator



The `Slow Queries Log` option is found in the `Log Files` tab of the `Startup Variables` screen. Click the clipboard icon to the left of the option to activate the Slow Query Log and click the APPLY CHANGES button. Once the Slow Query Log is activated, restart the MySQL server using the `Service Control` screen.

The default name of the log file is *`server-name`*`-slow.log`. If your server is named `doomhammer.myserver.org`, the log file will be named `doomhammer.myserver.org-slow.log`.

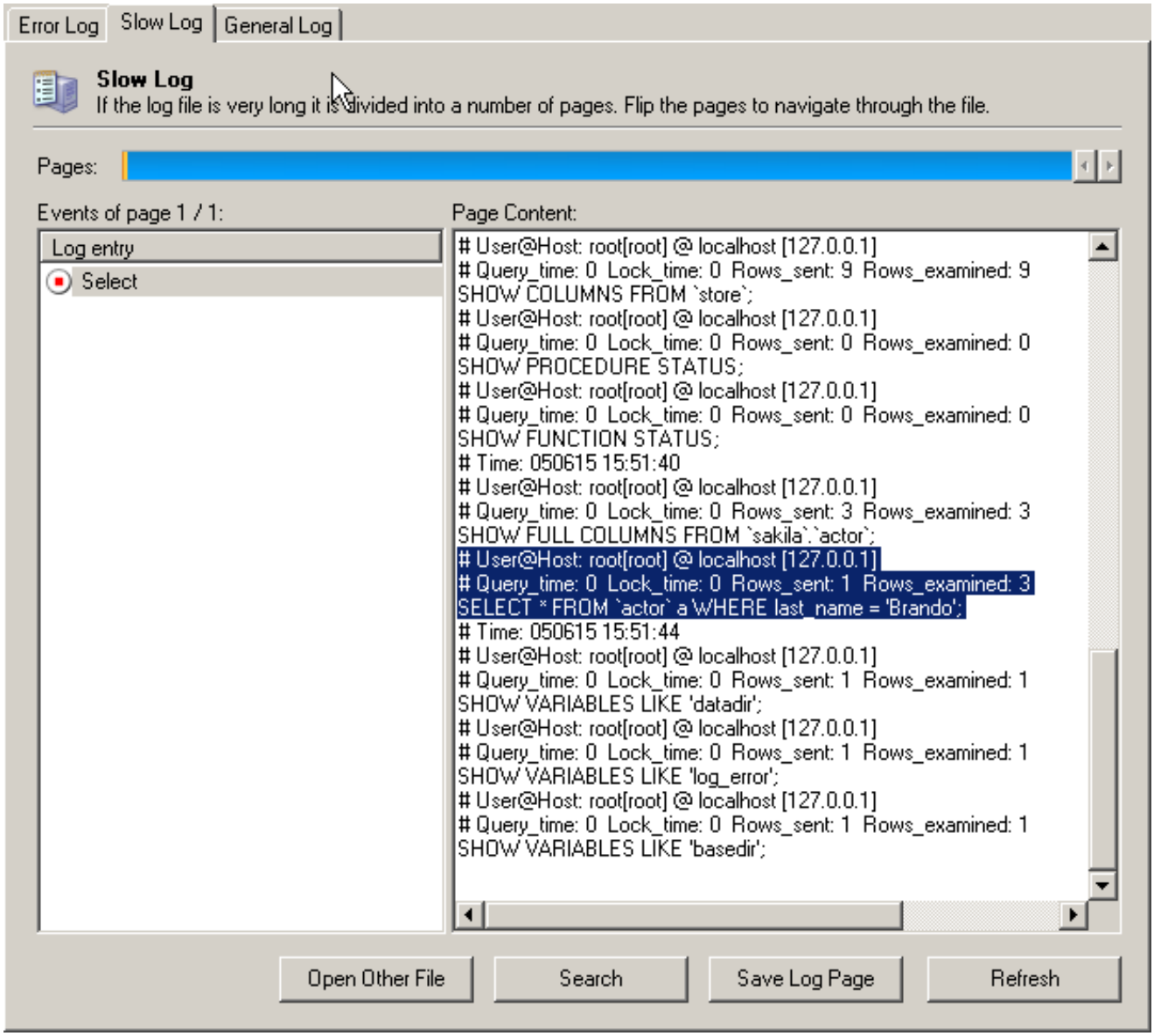
The Slow Query Log is a plain-text file that contains three lines for every query logged:

```
# User@Host: root[root] @ localhost [127.0.0.1]
# Query_time: 0  Lock_time: 0  Rows_sent: 1  Rows_examined: 3
SELECT last_name, first_name, actor_id FROM actor WHERE last_name = 'Brando';
```

The first line lists the username and hostname of the user who executed the query. The second line lists the time taken to execute the query, the time required to acquire the necessary locks, the number of rows returned by the query, and the number of rows the MySQL optimizer needed to examine. The final line of the entry shows the query that was executed.

The Slow Query Log can also be read using the MySQL Administrator using the `Slow Log` tab of the `Server Logs` screen:

Figure 25.2. Viewing the Slow Query Log with MySQL Administrator



The Slow Queries Log determines whether a query is slow by how long the query takes to execute in seconds, not counting the time required to acquire table locks. The default time is two seconds and can be adjusted by setting the `long_query_time` option in the `[mysqld]` section of the server configuration file. The `long_query_time` option can also be set using the `Log Files` tab of the

`Startup Variables` screen of MySQL Administrator.

It should be noted that queries can appear in the Slow Query Log even if they are properly optimized if the server load is high enough to cause the query to take longer than the `long_query_time`.

If you wish to log all queries that do not use indexes, regardless of how long the queries take to execute, add the `log-queries-not-using-indexes` option to the `[mysqld]` section of your MySQL server configuration file, or check the `Log queries that don't use indexes` option of the `Log Files` tab of the `Startup Variables` screen of MySQL Administrator.

25.4. Utilities for Use with the Logs

Part V. Using the MySQL APIs

Table of Contents

Chapter 26. Using the PHP5 mysqli API with MySQL

26.1. Configuring the mysqli Extension

Chapter 27. Using the Connector/J API with MySQL

27.1. Obtaining and Installing Connector/J

Chapter 28. Using the Connector/.NET API with MySQL

28.1. Obtaining and Installing Connector/.NET

Part Part VI. Tutorials

Table of Contents

Chapter 29. Other MySQL Utilities

29.1. Using `<mysqlutility>`

Chapter 30. Migrating a Spreadsheet to MySQL

30.1. Introduction

The most common “database” format, especially for small- to medium-sized businesses, is the spreadsheet. The reason for this is fairly obvious — no special skills are required either for design or for data entry. Not only that, a spreadsheet may well be the best format for presenting and maintaining some kinds of information. If the file is not complicated, it's easy to get a quick overview of the data and sorting on a specific field is usually just a matter of clicking a column heading.

However, as the volume or complexity of information increases, this format becomes more and more cumbersome. Information becomes more difficult to retrieve and you run into the kinds of problems usually associated with flat-table databases — data duplication, for example.

This chapter deals with migrating a spreadsheet to a MySQL database. The solution presented here is operating system (OS) neutral; it works on Mac, Windows, or any Unix-like OS.

Excel is probably the most commonly used spreadsheet format but the procedure described here applies to any spreadsheet. The only requirement is that the spreadsheet data be exported as a text file so that it can be imported into MySQL.

To help facilitate things Query Browser, one of the open source MySQL GUI Tools, will be used. Creating database objects is made especially easy using the `Table Editor`, a feature of the Query Browser also common to other GUI Tools. By pointing and clicking you can quickly build a table without knowing anything about data definition language (DDL). Not only will the table editor help you work more quickly, it's also a good way to learn MySQL's implementation of SQL. Any alterations made to a table using the graphical interface are shown as SQL statements, making it easy to learn the appropriate SQL commands. We'll take advantage of this feature to document as we go.

The example spreadsheet that we'll be importing contains information about the accreditations of members of a professional association. It's not complicated so the process should be fairly easy to follow but at the same time it does highlight the major issues you might encounter and provides general guidelines for importing spreadsheets into MySQL.

The steps we'll take are as follows:

1. Export the spreadsheet to a text file
2. Import this file wholesale into a temporary table
3. Create and populate permanent tables
4. Use the `mysqldump` utility to export the tables and data
5. Upload these tables and data to a production server

30.2. The Spreadsheet File

We want to import the data directly into a table that mirrors the structure of the spreadsheet. The sample spreadsheet has the following fields with maximum required lengths as shown:

- firstname – 50
- lastname – 50
- certification – 10
- expirydate – 10
- streetaddress1 – 50
- streetaddress2 – 50
- city – 50

- state – 2
- zipcode – 10
- certificationnumber – 10

If you wish to follow along, create a spreadsheet with these fields and enter some sample data as shown in the table below.

Table 30.1. Spreadsheet format

firstname	lastname	certification	expirydate	street1	street2	city	state	zip	certnum
John	Doe	RAC	10-Jan-08	10 Mulberry St		New York	NY	30263	C-12345
John	Doe	ARM-1	28-Feb-09	10 Mulberry St		New York	NY	30263	A-44456
Jane	Doe	DAC	10-Dec-09	10 Mulberry St		New York	NY	30263	D-4567
Bob	Smith	RAC	02-Jan-07	10 Main St	Apt 10	Detroit	MI	20789	C-6785

The only requirements are:

- Any identical combination of the `firstname`, `lastname`, and `street1` is always understood to apply to the same individual. The same individual can appear more than once in the file. However, ensure that the certification differs.
- Make sure that the certification numbers are unique.
- Format the date as `01-Dec-07`; that is, use two digits for the day, the standard month abbreviation, and a two digit year. Use a hyphen as a separator.
- Don't exceed the specified field lengths.

If you don't have a spreadsheet program at hand, you can create a tab-separated or comma-separated text file to match the structure defined above. Of course, if you do this, you won't need to export the data.

30.3. Converting a Spreadsheet to a Text File

Before attempting to convert any spreadsheet, it is best to review the data for consistency. For example, make sure that every row has the same number of columns and check that all dates are formatted in the same way. This may save major headaches by helping to spot errors early.

How you convert a spreadsheet to a text file may depend upon the OS you are using. If you are working under Windows with an Excel spreadsheet, open the spreadsheet in Excel and choose the SAVE AS menu option under the FILE menu. From the **SAVE AS TYPE** list box choose the `Text(Tab delimited)` option.

Under Unix, Windows, or Mac you can use OpenOffice Calc to open a variety of spreadsheet formats, including an Excel spreadsheet. Export the spreadsheet from the Calc application by choosing the FILE and SAVE AS menu options. Next choose `Text CSV` from the **FILTER** list box. This opens a dialog box for further refining your choice. Choose the character set `Unicode` or a platform-specific format, if appropriate. As a field delimiter choose the `{Tab}` option and no text delimiter at all. The drop-down list box only offers single or double quotation marks as alternatives; to choose no delimiter simply delete the quotation mark.

If you are using an application that won't let you save the spreadsheet in tab-separated format, a Google spreadsheet for example, then simply save the file as a CSV file. Tab-separated text files are the easiest to import into MySQL but importing a CSV file is almost as simple. Before saving the file, review the contents first and ensure that no commas appear anywhere in the data. A stray comma can cause data corruption or complete failure when importing data in CSV format.

Save the file as `data.tsv` (or as `data.csv` if the format is comma-separated), have a look at the exported data in a text editor. Each record should appear on a separate line. Don't be concerned if each line is not a uniform length. Many programs will export the column headings as the first row of the text file. Delete this row and resave the file, making sure that you save it as a text file and don't introduce

any formatting.

30.4. Creating a Table with Query Browser

Creating a table to match the fields as described in Section 30.2, “The Spreadsheet File” is a fairly straightforward matter. For importing the data our principal concern is to get the right information in the right fields without truncating data. By treating all fields as VARCHAR we can keep things simple and only need to worry about the order of the fields and their length.

As promised we'll use the MySQL Query Browser until we're ready to create a database dump. Query Browser is a fairly intuitive tool but for a quick overview find the documentation online at http://dev.mysql.com/doc/.

Start up Query Browser and enter your credentials and the server hostname and port — we haven't created a database yet so don't worry about the **DEFAULT SCHEMA** text box. When the application opens, you'll find a list of schemata (databases) on the right. The cursor should be active in the text area at the top of the screen. This text area is used for entering queries, which are executed using the EXECUTE button on the right. If a result set is returned, it shows in the main area in the center of the screen.

The first thing to do is create a database. Make sure that the **SCHEMATA** tab on the right is selected, right click anywhere in this window, and choose the CREATE SCHEMA option from the pop-up menu. Name the database `association`. To refresh the databases shown in the **SCHEMATA** window, right click in this window and choose the refresh menu option. Next open a script window — we'll use this window as a scratch pad to save copies of the queries we create. Open a script tab by choosing the NEW SCRIPT TAB option from the FILE menu. After doing this two tabs, one labeled **RESULTSET1** and the other **NEW SCRIPT**, should be visible on the left below the tool bar.

To create a table, right click the `association` database in the **SCHEMATA** panel and choose CREATE TABLE from the pop-up menu.

This opens the table editor, in the default view with the **COLUMNS AND INDICES** tab active. Enter the name `alldata` in the text box at the top of the table editor. Refer to the values shown in Section 30.2, “The Spreadsheet File”, enter a name for each column, choose `VARCHAR` as the data type, and specify a field length. You needn't worry about making any other changes at this point. After all, the `alldata` table is only temporary.

When you are finished, use the APPLY CHANGES button. This button opens a dialog box showing the SQL code that will execute. Before executing this code, copy it and paste it into the script window. The code should look something like this:

```
CREATE TABLE `alldata` (
  `lastname` VARCHAR(50) NOT NULL,
  `firstname` VARCHAR(50) NOT NULL,
  `certification` VARCHAR(10) NOT NULL,
  `expirydate` VARCHAR(10) NOT NULL,
  `streetaddress1` VARCHAR(50) NOT NULL,
  `streetaddress2` VARCHAR(50) NOT NULL,
  `city` VARCHAR(50) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zipcode` VARCHAR(10) NOT NULL,
  `certificationnumber` VARCHAR(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Right click the `association` database in the **SCHEMATA** pane and choose the REFRESH SCHEMATA option. The newly created table should appear beneath the `association` database, ready for imported data.

30.5. Loading the Data into a MySQL Database Table

To import the spreadsheet data we'll use the `LOAD DATA INFILE` syntax. Security considerations can sometimes make this a frustrating exercise, so as we go, we'll try to anticipate any problems that may arise.

Click on the **RESULTSET1** tab and enter the following statement into the query text box (using a path appropriate to your circumstances):

```
LOAD DATA INFILE “/home/peter/Documents/spreadsheet/data.tsv”
  INTO TABLE alldata;
```

Windows pathnames are also specified using forward slashes rather than backslashes. If you do use backslashes, you must double them.

To import a comma separated file on the Windows platform use the following syntax:

```
LOAD DATA INFILE “C:/Documents and Settings/peter/My Documents/spreadsheet/data.csv”
  INTO TABLE alldata
  FIELDS TERMINATED BY “,”;
```

The default field terminator is a tab character so if you use a different terminator you must specify it as shown in the preceding statement.

There are other possible pitfalls when executing a `LOAD DATA INFILE` statement. The rules for using a relative path are a bit tricky so always specify the complete path to the file. Also, a data file must be readable by all. This is usually not an issue under Windows; on Unix operating systems, if you need to adjust the file permissions, you can readily do this using the GUI. To make a file world-readable from the command prompt type:

```
shell> chmod 755 data.tsv
```

Finally, the user who is executing the `LOAD DATA` statement must have the `FILE` privilege. If you need to grant this privilege, log in as root and execute the command:

```
GRANT FILE ON *.*
  TO 'user'@'hostname'
  IDENTIFIED BY 'password';
```

You can do this from the command line or from within Query Browser.

> **Note**
>
> The `FILE` privilege is a global privilege and cannot be restricted to a specific database.

So far so good, but the syntax shown to this point only works if the text file is located on the same system as the server. If your MySQL server is remote, you must add the keyword `LOCAL` to the `LOAD DATA INFILE` syntax as in the following example:

```
LOAD DATA LOCAL INFILE "/home/peter/Documents/spreadsheet/data.tsv"
  INTO TABLE alldata;
```

Using `LOCAL` is not much different syntactically but servers are sometimes started up with the ability to `LOAD DATA LOCAL` disabled. If the server supports `LOCAL`, you can start up the MySQL client with the `--local-infile` option. Another approach is to copy the text file to the server before executing the `LOAD DATA` statement.

> **Note**
>
> Further complications can ensue. For files created on a Windows system, you might have to add `LINES TERMINATED BY '\r\n'` to read the file properly, because Windows programs typically use these two characters as a line terminator. If you need to add this clause, it follows immediately after the table name or, if a `FIELD TERMINATED BY` clause is present, immediately after this clause.

If you run into problems and require more information about `LOAD DATA INFILE` refer to the manual http://dev.mysql.com/doc/refman5.0/en/sql-syntax.html.

Before you continue, paste the appropriate version of the `LOAD DATA INFILE` statement into the script window below the `alldata` table definition.

After executing this statement and loading the data you can check that it has been copied to the `alldata` table using Query Browser. To inspect the data, double click the `alldata` table and find the following statement in the query text box:

```
SELECT * FROM alldata LIMIT 0,1000
```

> **Note**
>
> A `LIMIT` clause may not appear when using Query Browser under Windows.

Click the EXECUTE button and you should be able to view the data in the query window.

You might want to review the integrity of the data again at this point. A visual inspection is fine but you might also want to automate the process with an SQL statement such as the following:

```
SELECT * FROM alldata PROCEDURE ANALYSE();
```

(Note the spelling of `ANALYSE`.)

Among other things, this query shows actual minimum and maximum values for data in the various fields. If any of the maximum field length values equal the field length, then you have probably truncated data. Empty or `NULL` values in some fields may also indicate problems.

30.6. Creating A Temporary Table of Members

Now that the data has been copied into a MySQL database, we need to split it up into different tables — we want a proper relational database and not another flat database. The most obvious entity is a member, having the attributes name and address. The following fields from the `alldata` table belong to this entity exclusively:

```
lastname VARCHAR(50) NOT NULL
firstname VARCHAR(50) NOT NULL
streetaddress1 VARCHAR(50) NOT NULL
streetaddress2 VARCHAR(50) NOT NULL
city VARCHAR(50) NOT NULL
state VARCHAR(2) NOT NULL
zipcode VARCHAR(10) NOT NULL
```

Removing any fields that relate to certification gives us the basis for a members table.

We need to transfer data from the `alldata` table into a members table but, since members can have more than one certification and so appear more than once in the `alldata` table, we can't just copy all records over to a members table. To make sure that we have unique records we need a way of uniquely identifying each member. We can do this by combining a number of fields together to create a unique value — a combination of the `firstname`, `lastname`, and `streetaddress1` columns fits the bill. The combination of these fields could form a primary key, but it would be a very cumbersome one. For this reason, we're also going to add a numeric key value — an integer `AUTO_INCREMENT` field. The two new fields are:

```
unique_value VARCHAR(150)
id INT(11)
```

Create this table using the table editor in the same way that you created the `alldata` table. The only new element is an integer, auto increment field. To create this field select INTEGER as the data type and ensure that all three check boxes in the column options frame, `Primary Key`, `Not NULL`, and `Auto Increment`, are checked. Make sure the size of the `unique_value` column is adequate and add the other columns exactly as you did before.

When you're ready apply your changes and copy the SQL from the dialog box. It should look something like the following:

```
CREATE TABLE `tempmembers` (
  `unique_value` VARCHAR(150) DEFAULT NULL,
  `id` INT(11)  NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(30) NOT NULL DEFAULT '',
  `lastname` VARCHAR(40) NOT NULL DEFAULT '',
  `streetaddress1` VARCHAR(60) NOT NULL DEFAULT '',
  `streetaddress2` VARCHAR(60) NOT NULL DEFAULT '',
  `city` VARCHAR(60) NOT NULL DEFAULT '',
  `state` VARCHAR(10) NOT NULL DEFAULT '',
  `zipcode` VARCHAR(10) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Paste this table definition into the script window and switch to the **RESULTSET1** tab to create a query to populate this table. As described above, we want to concatenate three columns to create a unique value and also add an auto increment column. The remaining columns come directly from the `alldata` table.

To populate the `tempmembers` table enter the following SQL into the query text box and execute it:

```
INSERT INTO tempmembers
  SELECT DISTINCT CONCAT(firstname, lastname, streetaddress1) AS unique_value,
  NULL AS id, firstname, lastname,
  streetaddress1, streetaddress2, city, state, zipcode
  FROM alldata;
```

Using DISTINCT with the `unique_value` field should guarantee that we don't have duplicate members and selecting NULL as the `id` field generates a unique auto increment value for each record. Look at the records in the `tempmembers` table to confirm that unique id numbers have been generated.

This is fairly close to what a final version of a members table would look like — removing the `unique_value` field would be the next step to take but as you'll see shortly, we still need this field.

30.7. Creating a Temporary Member Accreditations Table

The entire spreadsheet that we've imported could be described as a table of members' different accreditations. In the previous section we extracted the member information from the `alldata` table and created a unique id number for each member. The task now is to replace the duplicated member information with a single unique field. In other words, we're going to create a member accreditations table with a foreign key.

The fields in the `alldata` table that apply solely to a member accreditations table are readily identified:

```
`certification` VARCHAR(10) NOT NULL
`expirydate` VARCHAR(10) NOT NULL,
`certificationnumber` VARCHAR(10) NOT NULL
```

So far we've treated the `expirydate` field as text. While we're creating a member accreditations table we can convert this field to the DATE data type. The new definition for this field is:

```
`expirydate` DATE DEFAULT NULL
```

Again we want to concatenate three columns to create a unique value and also add an integer column for the member id — so we can relate the member certifications to their matching records in the members table. The two additional columns are as follows:

```
`unique_value` VARCHAR(150) DEFAULT NULL
`memberid` INT(11) NOT NULL DEFAULT '0',
```

Right click the `association` database in the **SCHEMATA** pane and open the table editor.

You've already added `VARCHAR` and `INTEGER` fields so adding a `DATE` type field should present no problems. Create a table named `tempmemberaccreditations` and apply your changes. The resulting table should look something like this:

```
CREATE TABLE `tempmemberaccreditations` (
  `unique_value` VARCHAR(150) DEFAULT NULL,
  `certification` VARCHAR(10) NOT NULL,
  `memberid` INT(11) NOT NULL DEFAULT '0',
  `certificationnumber` VARCHAR(10) NOT NULL,
  `expirydate` DATE DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Don't forget to paste it into the script window before proceeding. If you do forget, retrieving the table structure is a simple matter of executing the SQL statement:

```
SHOW CREATE TABLE tempmemberaccreditations;
```

Again we need to populate this table from the `alldata` table. We are going to select all the records from the `alldata` table but only selected fields. Click on the **RESULTSET1** tab and enter the following query into the query text box:

```
INSERT INTO tempmemberaccreditations
  SELECT CONCAT(firstname,lastname,streetaddress1) AS unique_value,
  certification, 0 AS memberid,
  certificationnumber,
  STR_TO_DATE(expirydate, "%d-%b-%y")
  FROM alldata;
```

Converting a string value to a date is done using the `STR_TO_DATE` function. This function takes two string arguments; the first is a string expression of the date and the second specifies the date format. In the `alldata` table dates are in the form '12-Dec-07'. The format specifier tells MySQL exactly how to interpret the string representation of the date. In this case the specifier, "%d-%b-%y", means the day of the month comes first and is expressed as two digits — it will have a leading zero even if the value is less than 10, the month is expressed as an abbreviated name, and the year numerically with two digits. All values are separated by a “-”.

> **Note**
>
> The complete list of specifiers is given in the manual immediately following discussion of the `DATE_FORMAT` function.

Review the data after executing the `INSERT` statement. You'll see that dates are now expressed in the default MySQL format, the year has four digits followed by a two digit month, and a two digit day.

At this point, reviewing the data to ensure consistency is a good idea. Any dates that were improperly formatted in the original spreadsheet will not convert to the `DATE` data type.

When reviewing the data you'll also see that the `memberid` field is set to '0' for all records. Let's update this field using the values in the tempmembers table:

```
UPDATE tempmemberaccreditations t2
  INNER JOIN tempmembers t ON  t.unique_value=t2.unique_value
  SET t2.memberid =  t.id;
```

That's the last time we'll need the `unique_value` field for either of our transitional tables. We can now relate these two tables on the numeric id field.

30.8. The Final Tables

With both tables populated with data it's time to get rid of the `unique_value` field and while doing so we should also change the name of our tables since they are no longer temporary or transitional tables.

Select the `tempmembers` table in the **SCHEMATA** pane and open the table editor. Rename the table to `members`, by changing the table name in the text box in the top left of the table editor. Select the `unique_value` field and press the **Delete** key to remove it. Choose APPLY CHANGES to view a dialog box with the following content:

```
ALTER TABLE `tempmembers`
  RENAME TO `members`,
  DROP COLUMN `unique_value`;
```

Making similar changes to the `tempmemberaccreditations` table will result in the following DDL statement:

```
ALTER TABLE `tempmemberaccreditations`
  RENAME TO `memberaccreditations`
  DROP COLUMN `unique_value`;
```

Copy the SQL version of these table alterations to the script window.

Adding indexes to tables is also easily accomplished using the Query Browser. Since we expect searches on the `lastname` and the `city` fields these two columns are ideal candidates for indexing. Again this can be done using the table editor. Open the table editor and click the **INDICES** tab. Click the + button on the bottom left and a new index called `new_index` appears in the list of indexes. Change the name to `lastname_idx` and drag and drop the `lastname` column to the `Columns` text area on the right.

Create an index on the city column in the same way. When you apply your changes you should see something similar to the following:

```
ALTER TABLE `members`
  ADD KEY `lastname_idx` (`lastname`),
  ADD KEY `city_idx` (`city`);
```

The `memberaccreditations` table still lacks a primary key. To remedy this, open the table editor again so that we can add a primary key. To do this click the + button on the lower left and ensure that `PRIMARY` is selected in the **KIND** drop-down list box. Create a primary key composed of two columns by dragging the `memberid` column and the `certification` column to the **COLUMNS** list. When applying your changes you should see:

```
ALTER TABLE `memberaccreditations`
  ADD PRIMARY KEY (`certification`, `memberid`);
```

After altering database objects, it's always an idea to refresh the view in the **SCHEMATA** pane. Do this by right clicking the `association` database and choosing the REFRESH option (Under Unix this option is called REFRESH SCHEMATA.)

Looking at the data there is yet one more change we could apply. The `certification` field may indicate another database entity. Let's create a table of accreditation acronyms with their corresponding descriptions.

One of the simplest ways to create a table and populate it using MySQL is to issue a CREATE TABLE statement in conjunction with a SELECT statement. For instance we could create our final version of the members table in the following way:

```
CREATE TABLE  accreditations
  SELECT DISTINCT certification AS acronym, '' AS description
  FROM alldata;
```

At this point we don't have the information necessary to add a description so we populate this field with an empty string.

Creating and populating a table in this way is a quick and easy way to create a populated table. The downside to creating a table in this way is that the resulting table has no primary key or indexes. I'll leave it to you to add an index to this table.

At this point we've created all the necessary tables and migrated the data to those tables. We just need to check the integrity of the data before copying it to a production server.

30.9. Confirming Data Integrity

It's always wise to check the state of your transformed data. There's no substitute for visual inspection but there are a variety of ways to check your data using SQL.

For example, there should be no orphaned member records. Since we've migrated from a flat-table database that contained all the original data, finding an id in the members table with no corresponding record in the member accreditations table would indicate that something was wrong. The following SQL statement will return all records in the members table that don't have matching records in the member accreditations table:

```
SELECT `t`.`id` ,`t`.`firstname`,`t`.`lastname`
  FROM `members` `t`
  LEFT JOIN `memberaccreditations` `tma`
  ON `t`.`id` = `tma`.`memberid`
  WHERE ISNULL(`tma`.`memberid`);
```

If the above `SELECT` statement returns an empty set, there are no orphaned member records.

An easy way to reuse this SQL statement is to save it as a view. To do this using Query Browser, make sure the `association` database is active, then right click on any one of the tables in the `Schemata` window and choose the CREATE VIEW option. Clicking OKAY after entering a view name opens a new tab displaying the basic syntax for creating a view. Paste the preceding SQL statement into the `AS` clause and execute the query. After refreshing the schemata the new view should show up. You can view the record set associated with this view in exactly the same way that you would view the record set associated with a table.

To check that there are no orphaned records in the member accreditations table execute the following query:

```
SELECT `tma`.`certification`,
  `tma`.`memberid`,`tma`.`certificationnumber`,
  `tma`.`expirydate`
  FROM (`memberaccreditations` `tma`
  LEFT JOIN `members` `t` ON ((`tma`.`memberid` = `t`.`id`)))
  WHERE ISNULL(`t`.`id`);
```

Again, to save this SQL statement, convert it to a view using the procedure described above.

There are also various other ways of querying your records to verify the data. For example, if all certification numbers in the member accreditations table are meant to be unique, executing the following query would determine if there are duplicates:

```
SELECT COUNT(t.`certificationnumber`), t.`certificationnumber`
  FROM memberaccreditations t
  GROUP BY (t.`certificationnumber`)
  HAVING COUNT(t.`certificationnumber`) > 1;
```

Checking the number of records in the `memberaccreditations` table provides further assurance of the integrity of your data. The number should exactly match the number of records in the `alldata` table.

If you notice discrepancies in the data and wish to update records you can do this from within Query Browser. Click the START EDITING button and then select the record you wish to change and place the cursor in the column you wish to change. When you are finished editing click the APPLY CHANGES button.

> **Note**
>
> If a record set is created from a single table having a primary key, it is editable. A disabled START EDITING button indicates that the record set is not editable.

Once you're satisfied with the integrity of the data, drop the `alldata` table. This is easily done by right clicking the table and choosing the DROP option. Before exiting Query Browser make sure that you save the script file of all the queries.

30.10. The Production Database

Once you are convinced of the validity of your data, you can move the tables to your production server. We'll do that by first using the `mysqldump` utility. To export only the final versions of the tables, go to the command line and type:

```
shell> mysqldump -u username -p --databases association > newdb.sql
```

> **Note**
>
> You can open a MySQL console window from within Query Browser. Find this option under the TOOLS menu.

The `mysqldump` utility takes many of the same switches as the MySQL client; as you can see, you specify your user name and password in the same way. You also need to specify the database name you wish to dump. In this case, the output is redirected to a script file named `newdb.sql`. If you do not wish to create a database and only want to dump the tables in the `association` database, execute the preceding command without the `--databases` option. For more information about the many options available with `mysqldump` see http://dev.mysql.com/doc/5.0/en/mysqldump.html.

Have a look at the contents of the script file so that you understand what it does. Any existing tables with the specified table names will be dropped and recreated and then the data will be inserted. If you are overwriting existing data, you may want to back up your data before running the script file.

How you execute the dump script file depends upon how you access your production MySQL server. If you have direct access to the server or access through ssh, transfer the script file to the machine hosting the server, and then issue the command:

```
shell> mysql -u username -p  < newdb.sql
```

> **Note**
>
> If you saved only the database tables, you must specify a database when issuing the preceding command.

If you have remote access to your production server simply add the `-h hostname` option to the preceding command. You may also upload your script using an application such as phpMyAdmin. Finally, you can open and execute the script file from within Query Browser — but more about this in the next section.

30.11. Updating a MySQL Database from a Spreadsheet

Migrating spreadsheet data to a MySQL database can be a relatively simple task especially when using Query Browser. However, migrating users to that database can be much more difficult. You may find that you continue to receive database updates in the form of complete but modified spreadsheets.

It may seem counterintuitive, but such updates can be handled most easily by recreating the entire database again. If we script this process then updates can be done in a matter of seconds. All we need are a few modifications to the script file that we saved as we worked.

The only additions to this script are `DROP TABLE` statements — making it much easier to reuse the database that's already there. This script can be run from the command line as described in the previous section or you can open it within Query Browser.

To open a script file from within Query Browser choose the OPEN SCRIPT option under the FILE menu. Find the script file and select it. A script file tab will open showing the contents of the file. Syntax highlighting is one of the advantages of executing a script from within Query Browser — errors are much more easily spotted. Any errors that occur during execution are displayed in a pop-up dialog, specifying the nature of the error and also the line number. You can also set break points and step through the code one line at a time if you wish.

Using Query Browser made it easy to document our actions in migrating a spreadsheet to MySQL. This documentation is easily turned into a script file so that we can recreate the process. It can also serve as a reference for techniques to use in future migrations. Find a copy of the script file in the next section.

30.12. The Migration Script

```
#use database
USE association;

#First make copy of Excel data
#treat all fields as text
DROP TABLE IF EXISTS `alldata`;

CREATE TABLE `alldata` (
  `lastname` VARCHAR(50) NOT NULL,
  `firstname` VARCHAR(50) NOT NULL,
```

```
  `certification` VARCHAR(10) NOT NULL,
  `expirydate` VARCHAR(10) NOT NULL,
  `streetaddress1` VARCHAR(50) NOT NULL,
  `streetaddress2` VARCHAR(50) NOT NULL,
  `city` VARCHAR(50) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zipcode` VARCHAR(10) NOT NULL,
  `certificationnumber` VARCHAR(10) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

#get data from tab-separated file created from spreadsheet
#Unfortunately, variable below won't work with LOAD DATA
#SET @filename = "/home/peter/Documents/spreadsheet/data.tsv";
#so hard code
LOAD DATA INFILE "/home/peter/Documents/spreadsheet/data.tsv"
  INTO TABLE alldata;
#Our server and client are on the same machine
#Don't need "LOCAL" if file is on the server (local means local to the client)
#will need the FILE privilege though
#but if it's there you can execute this script from somewhere else on your network
#Syntax with comma separated fields -- not as safe as tabs
#LOAD DATA INFILE "C:/Documents and Settings/peter/My Documents/spreadsheet/data.csv"
#  INTO TABLE alldata
#  FIELDS TERMINATED BY ",";

#create temporary tables
#Association members information
DROP TABLE IF EXISTS `tempmembers`;

CREATE TABLE `tempmembers` (
  `unique_value` VARCHAR(150) DEFAULT NULL,
  `id` INT(11)  NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(30) NOT NULL DEFAULT '',
  `lastname` VARCHAR(40) NOT NULL DEFAULT '',
  `streetaddress1` VARCHAR(60) NOT NULL DEFAULT '',
  `streetaddress2` VARCHAR(60) NOT NULL DEFAULT '',
  `city` VARCHAR(60) NOT NULL DEFAULT '',
  `state` VARCHAR(10) NOT NULL DEFAULT '',
  `zipcode` VARCHAR(10) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO tempmembers
  SELECT DISTINCT CONCAT(firstname,lastname,streetaddress1) AS unique_value,
  NULL AS id, firstname, lastname,
  streetaddress1, streetaddress2, city, state, zipcode
  FROM alldata;

#Member accreditations
DROP TABLE IF EXISTS `tempmemberaccreditations`;
CREATE TABLE `tempmemberaccreditations` (
  `unique_value` VARCHAR(150) DEFAULT NULL,
  `certification` VARCHAR(10) NOT NULL,
  `memberid` INT(11) NOT NULL DEFAULT '0',
  `certificationnumber` VARCHAR(10) NOT NULL,
  `expirydate` DATE DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO tempmemberaccreditations
  SELECT CONCAT(firstname,lastname,streetaddress1) AS unique_value,
  certification, 0 AS memberid,
  certificationnumber,
  STR_TO_DATE(expirydate, "%d-%b-%y")
  FROM alldata;
#Above format for 31-Dec-07
#for 12/1/2007 use "%c/%e/%Y"

#now relate the two tables and insert ids into the tempmemberaccreditations

UPDATE tempmemberaccreditations t2
  INNER JOIN tempmembers t ON  t.unique_value=t2.unique_value
  SET t2.memberid =  t.id;

#create final version of tables
DROP TABLE IF EXISTS `members`;
ALTER TABLE `tempmembers`
  RENAME TO `members`,
  DROP COLUMN `unique_value`;

DROP TABLE IF EXISTS `memberaccreditations`;
ALTER TABLE `tempmemberaccreditations`
  RENAME TO `memberaccreditations`,
  DROP COLUMN `unique_value`;

#add indices
ALTER TABLE `members`
  ADD KEY `lastname_idx` (`lastname`),
  ADD KEY `city_idx` (`city`);

ALTER TABLE `memberaccreditations`
```

```
  ADD PRIMARY KEY (`certification`, `memberid`);

#create accreditations table
DROP TABLE IF EXISTS accreditations;
CREATE TABLE  accreditations
  SELECT DISTINCT certification AS acronym, '' AS description
  FROM alldata;

ALTER TABLE `accreditations`
  ADD PRIMARY KEY (`acronym`);
#now add views
DROP VIEW IF EXISTS vwOrphanedMembers;
CREATE VIEW `vwOrphanedMembers` AS
  SELECT `t`.`id` ,`t`.`firstname`,`t`.`lastname`
  FROM `members` `t`
  LEFT JOIN `memberaccreditations` `tma`
  ON `t`.`id` = `tma`.`memberid`
  WHERE ISNULL(`tma`.`memberid`);

DROP VIEW IF EXISTS vwOrphanedAccreditations;
CREATE VIEW `vwOrphanedAccreditations` AS
  SELECT `tma`.`certification`,
  `tma`.`memberid`,`tma`.`certificationnumber`,
  `tma`.`expirydate`
  FROM (`memberaccreditations` `tma`
  LEFT JOIN `members` `t` ON ((`tma`.`memberid` = `t`.`id`)))
  WHERE ISNULL(`t`.`id`);
#remove spreadsheet-based table
DROP TABLE IF EXISTS `alldata`;
```

Chapter 31. Migrating an Access Database to MySQL

Chapter 32. Using PHP Data Objects (PDO) With MySQL

Chapter 33. Using mysqlnd

Chapter 34. Ruby and MySQL

Chapter 35. phpMyAdmin

Part VII. Appendixes

Table of Contents

Appendix A. Date Format Specifiers Table

Appendix B. Functions and Operators Tables

This appendix contains a listing of all the MySQL functions and operators.

Name	Description
ABS()	Return the absolute value
ACOS()	Return the arc cosine
ADDDATE()(v4.1.1)	Add dates
ADDTIME()(v4.1.1)	Add time
AES_DECRYPT()	Decrypt using AES
AES_ENCRYPT()	Encrypt using AES
AND, &&	Logical AND
ASCII()	Return numeric value of left-most character
ASIN()	Return the arc sine
ATAN2(),ATAN()	Return the arc tangent of the two arguments
ATAN()	Return the arc tangent
AVG()	Return the average value of the argument
BENCHMARK()	Repeatedly execute an expression
BETWEEN ... AND ...	Check whether a value is within a range of values
BIN()	Return a string representation of the argument
BINARY	Cast a string to a binary string
BIT_AND()	Return bitwise and
BIT_COUNT()	Return the number of bits that are set
BIT_LENGTH()	Return length of argument in bits
BIT_OR()	Return bitwise or
BIT_XOR()(v4.1.1)	Return bitwise xor
&	Bitwise AND
~	Invert bits
|	Bitwise OR
^	Bitwise XOR
CASE	Case operator
CAST()	Cast a value as a certain type
CEIL()	Return the smallest integer value not less than the argument
CEILING()	Return the smallest integer value not less than the argument
CHAR_LENGTH()	Return number of characters in argument
CHAR()	Return the character for each integer passed
CHARACTER_LENGTH()	A synonym for CHAR_LENGTH()
CHARSET()(v4.1.0)	Return the character set of the argument
COALESCE()	Return the first non-NULL argument
COERCIBILITY()(v4.1.1)	Return the collation coercibility value of the string argument
COLLATION()(v4.1.0)	Return the collation of the string argument
COMPRESS()(v4.1.1)	Return result as a binary string
CONCAT_WS()	Return concatenate with separator
CONCAT()	Return concatenated string
CONNECTION_ID()	Return the connection ID (thread ID) for the connection
CONV()	Convert numbers between different number bases

Name	Description
CONVERT_TZ()(v4.1.3)	Convert from one timezone to another
Convert()	Cast a value as a certain type
COS()	Return the cosine
COT()	Return the cotangent
COUNT(DISTINCT)	Return the count of a number of different values
COUNT()	Return a count of the number of rows returned
CRC32()(v4.1.0)	Compute a cyclic redundancy check value
CURDATE()	Return the current date
CURRENT_DATE(),CURRENT_DATE	Synonyms for CURDATE()
CURRENT_TIME(),CURRENT_TIME	Synonyms for CURTIME()
CURRENT_TIMESTAMP(),CURRENT_TIMESTAMP	Synonyms for NOW()
CURRENT_USER(),CURRENT_USER	Return the username and hostname combination
CURTIME()	Return the current time
DATABASE()	Return the default (current) database name
DATE_ADD()	Add two dates
DATE_FORMAT()	Format date as specified
DATE_SUB()	Subtract two dates
DATE()(v4.1.1)	Extract the date part of a date or datetime expression
DATEDIFF()(v4.1.1)	Subtract two dates
DAY()(v4.1.1)	Synonym for DAYOFMONTH()
DAYNAME()(v4.1.21)	Return the name of the weekday
DAYOFMONTH()	Return the day of the month (1-31)
DAYOFWEEK()	Return the weekday index of the argument
DAYOFYEAR()	Return the day of the year (1-366)
DECODE()	Decodes a string encrypted using ENCODE()
DEFAULT()	Return the default value for a table column
DEGREES()	Convert radians to degrees
DES_DECRYPT()	Decrypt a string
DES_ENCRYPT()	Encrypt a string
DIV(v4.1.0)	Integer division
/	Division operator
ELT()	Return string at index number
ENCODE()	Encode a string
ENCRYPT()	Encrypt a string
<=>	NULL-safe equal to operator
=	Equal operator
EXP()	Raise to the power of
EXPORT_SET()	Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
EXTRACT	Extract part of a date
ExtractValue()(v5.1.5)	Extracts a value from an XML string using XPath notation
FIELD()	Return the index (position) of the first argument in the subsequent arguments
FIND_IN_SET()	Return the index position of the first argument within the second argument
FLOOR()	Return the largest integer value not greater than the argument

Name	Description
FORMAT()	Return a number formatted to specified number of decimal places
FOUND_ROWS()	For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause
FROM_DAYS()	Convert a day number to a date
FROM_UNIXTIME()	Format date as a UNIX timestamp
GET_FORMAT()(v4.1.1)	Return a date format string
GET_LOCK()	Get a named lock
>=	Greater than or equal operator
>	Greater than operator
GREATEST()	Return the largest argument
GROUP_CONCAT()(v4.1)	Return a concatenated string
HEX()	Return a hexadecimal representation of a decimal or string value
HOUR()	Extract the hour
IF()	If/else construct
IFNULL()	Null if/else construct
IN()	Check whether a value is within a set of values
INET_ATON()	Return the numeric value of an IP address
INET_NTOA()	Return the IP address from a numeric value
INSERT()	Insert a substring at the specified position up to the specified number of characters
INSTR()	Return the index of the first occurrence of substring
INTERVAL()	Return the index of the argument that is less than the first argument
IS_FREE_LOCK()	Checks whether the named lock is free
IS NOT NULL	NOT NULL value test
IS NOT	Test a value against a boolean
IS NULL	NULL value test
IS_USED_LOCK()(v4.1.0)	Checks whether the named lock is in use. Return connection identifier if true.
IS	Test a value against a boolean
ISNULL()	Test whether the argument is NULL
LAST_DAY(v4.1.1)	Return the last day of the month for the argument
LAST_INSERT_ID()	Value of the AUTOINCREMENT column for the last INSERT
LCASE()	Synonym for LOWER()
LEAST()	Return the smallest argument
<<	Left shift
LEFT()	Return the leftmost number of characters as specified
LENGTH()	Return the length of a string in bytes
<=	Less than or equal operator
<	Less than operator
LIKE	Simple pattern matching
LN()	Return the natural logarithm of the argument
LOAD_FILE()	Load the named file
LOCALTIME(), LOCALTIME	Synonym for NOW()
LOCALTIMESTAMP, LOCALTIMESTAMP()(v4.0.6)	Synonym for NOW()
LOCATE()	Return the position of the first occurrence of substring

Name	Description
LOG10()	Return the base-10 logarithm of the argument
LOG2()	Return the base-2 logarithm of the argument
LOG()	Return the natural logarithm of the first argument
LOWER()	Return the argument in lowercase
LPAD()	Return the string argument, left-padded with the specified string
LTRIM()	Remove leading spaces
MAKE_SET()	Return a set of comma-separated strings that have the corresponding bit in bits set
MAKEDATE()(v4.1.1)	Create a date from the year and day of year
MAKETIME(v4.1.1)	MAKETIME()
MASTER_POS_WAIT()	Block until the slave has read and applied all updates up to the specified position
MATCH	Perform full-text search
MAX()	Return the maximum value
MD5()	Calculate MD5 checksum
MICROSECOND()(v4.1.1)	Return the microseconds from argument
MID()	Return a substring starting from the specified position
MIN()	Return the minimum value
-	Minus operator
MINUTE()	Return the minute from the argument
MOD()	Return the remainder
%	Modulo operator
MONTH()	Return the month from the date passed
MONTHNAME()(v4.1.21)	Return the name of the month
NAME_CONST()(v5.0.12)	Causes the column to have the given name
NOT BETWEEN ... AND ...	Check whether a value is not within a range of values
!=, <>	Not equal operator
NOT IN()	Check whether a value is not within a set of values
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP
NOT, !	Negates value
NOW()	Return the current date and time
NULLIF()	Return NULL if expr1 = expr2
OCT()	Return an octal representation of a decimal number
OCTET_LENGTH()	A synonym for LENGTH()
OLD_PASSWORD()(v4.1)	Return the value of the old (pre-4.1) implementation of PASSWORD
||, OR	Logical OR
ORD()	Return character code for leftmost character of the argument
PASSWORD()	Calculate and return a password string
PERIOD_ADD()	Add a period to a year-month
PERIOD_DIFF()	Return the number of months between periods
PI()	Return the value of pi
+	Addition operator
POSITION()	A synonym for LOCATE()
POW()	Return the argument raised to the specified power

Name	Description
POWER()	Return the argument raised to the specified power
PROCEDURE ANALYSE()	Analyze the results of a query
QUARTER()	Return the quarter from a date argument
QUOTE()	Escape the argument for use in an SQL statement
RADIANS()	Return argument converted to radians
RAND()	Return a random floating-point value
REGEXP	Pattern matching using regular expressions
RELEASE_LOCK()	Releases the named lock
REPEAT()	Repeat a string the specified number of times
REPLACE()	Replace occurrences of a specified string
REVERSE()	Reverse the characters in a string
>>	Right shift
RIGHT()	Return the specified rightmost number of characters
RLIKE	Synonym for REGEXP
ROUND()	Round the argument
ROW_COUNT()(v5.0.1)	The number of rows updated
RPAD()	Append string the specified number of times
RTRIM()	Remove trailing spaces
SCHEMA()(v5.0.2)	A synonym for DATABASE()
SEC_TO_TIME()	Converts seconds to 'HH:MM:SS' format
SECOND()	Return the second (0-59)
SESSION_USER()	Synonym for USER()
SHA1(),SHA()	Calculate an SHA-1 160-bit checksum
SIGN()	Return the sign of the argument
SIN()	Return the sine of the argument
SLEEP()(v5.0.12)	Sleep for a number of seconds
SOUNDEX()	Return a soundex string
SOUNDS LIKE(v4.1.0)	Compare sounds
SPACE()	Return a string of the specified number of spaces
SQRT()	Return the square root of the argument
STD()	Return the population standard deviation
STDDEV_POP()(v5.0.3)	Return the population standard deviation
STDDEV_SAMP()(v5.0.3)	Return the sample standard deviation
STDDEV()	Return the population standard deviation
STR_TO_DATE()(v4.1.1)	Convert a string to a date
STRCMP()	Compare two strings
SUBDATE()	When invoked with three arguments a synonym for DATE_SUB()
SUBSTR()	Return the substring as specified
SUBSTRING_INDEX()	Return a substring from a string before the specified number of occurrences of the delimiter
SUBSTRING()	Return the substring as specified
SUBTIME()(v4.1.1)	Subtract times
SUM()	Return the sum
SYSDATE()	Return the time at which the function executes

Name	Description
SYSTEM_USER()	Synonym for USER()
TAN()	Return the tangent of the argument
TIME_FORMAT()	Format as time
TIME_TO_SEC()	Return the argument converted to seconds
TIME()(v4.1.1)	Extract the time portion of the expression passed
TIMEDIFF()(v4.1.1)	Subtract time
*	Times operator
TIMESTAMP()(v4.1.1)	With a single argument, this function returns the date or datetime expression. With two arguments, the sum of the arguments
TIMESTAMPADD()(v5.0.0)	Add an interval to a datetime expression
TIMESTAMPDIFF()(v5.0.0)	Subtract an interval from a datetime expression
TO_DAYS()	Return the date argument converted to days
TRIM()	Remove leading and trailing spaces
TRUNCATE()	Truncate to specified number of decimal places
UCASE()	Synonym for UPPER()
-	Change the sign of the argument
UNCOMPRESS()(v4.1.1)	Uncompress a string compressed
UNCOMPRESSED_LENGTH()(v4.1.1)	Return the length of a string before compression
UNHEX()(v4.1.2)	Convert each pair of hexadecimal digits to a character
UNIX_TIMESTAMP()	Return a UNIX timestamp
UpdateXML()(v5.1.5)	Return replaced XML fragment
UPPER()	Convert to uppercase
USER()	Return the current username and hostname
UTC_DATE()(v4.1.1)	Return the current UTC date
UTC_TIME()(v4.1.1)	Return the current UTC time
UTC_TIMESTAMP()(v4.1.1)	Return the current UTC date and time
UUID()(v4.1.2)	Return a Universal Unique Identifier (UUID)
VALUES()(v4.1.1)	Defines the values to be used during an INSERT
VAR_POP()(v5.0.3)	Return the population standard variance
VAR_SAMP()(v5.0.3)	Return the sample variance
VARIANCE()(v4.1)	Return the population standard variance
VERSION()	Returns a string that indicates the MySQL server version
WEEK()	Return the week number
WEEKDAY()	Return the weekday index
WEEKOFYEAR()(v4.1.1)	Return the calendar week of the date (1-53)
XOR	Logical XOR
YEAR()	Return the year
YEARWEEK()	Return the year and week

Appendix C. Options Tables

This appendix contains listings of options for the most-used MySQL programs. These tables contain brief descriptions along with hyperlinks to the manual. Where applicable, listings of commands are also supplied.

C.1. `mysql` Options

Table C.1. `mysql` Option Reference

Format	Config File	Description	Introduction
--auto-rehash	auto-rehash	Enable automatic rehashing	
--batch	batch	Don't use history file	
--bind-address=host_name		Determine which client network interface (IP address or hostname) to use when connecting to the MySQL Server	5.1.22-ndb-6.3.4
--character-sets-dir=name	character-sets-dir	Set the default character set	
--column-names	column-names	Write column names in results	
--column-type-info	column-type-info	Display result set metadata	5.1.14
--comments	comments	Whether to retain or strip comments in statements sent to the server	5.1.23
--compress	compress	Compress all information sent between the client and the server	
--connect_timeout=value	connect_timeout	The number of seconds before connection timeout	
--database=dbname	database	The database to use	
--debug[=debug_options]	debug	Write a debugging log	
--debug-check	debug-check	Print debugging information when the program exits	5.1.21
--debug-info	debug-info	Print debugging information, memory and CPU statistics when the program exits	
--default-character-set=charset_name	default-character-set	Use charset_name as the default character set	
--delimiter=str	delimiter	Set the statement delimiter	
--execute=statement	execute	Execute the statement and quit	
--force	force	Continue even if an SQL error occurs	
--help		Display help message and exit	
--host=host_name	host	Connect to the MySQL server on the given host	
--html	html	Produce HTML output	
--ignore-spaces	ignore-spaces	Ignore spaces after function names	
--line-numbers	line-numbers	Write line numbers for errors	
--local-infile[={0|1}]	local-infile	Enable or disable for LOCAL capability for LOAD DATA INFILE	
--max_allowed_packet=value	max_allowed_packet	The maximum packet length to send to or receive from the server	
--max_join_size=value	max_join_size	The automatic limit for rows in a join when using --safe-updates	
--named-commands	named-commands	Enable named mysql commands	
--net_buffer_length=value	net_buffer_length	The buffer size for TCP/IP and socket communication	
--no-auto-rehash		Disable automatic rehashing	

Format	Config File	Description	Introduction
--no-beep	no-beep	Do not beep when errors occur	
--no-named-commands	no-named-commands	Disable named mysql commands	
--no-pager	no-pager	Deprecated form of --skip-pager	
--no-tee	no-tee	Do not copy output to a file	
--one-database	one-database	Ignore statements except those for the default database named on the command line	
--pager[=command]	pager	Use the given command for paging query output	
--password[=password]	password	The password to use when connecting to the server	
--port=port_num	port	The TCP/IP port number to use for the connection	
--prompt=format_str	prompt	Set the prompt to the specified format	
--protocol={TCP|SOCKET|PIPE|MEMORY}	protocol	The connection protocol to use	
--quick	quick	Do not cache each query result	
--raw	raw	Write column values without escape conversion	
--reconnect	reconnect	If the connection to the server is lost, automatically try to reconnect	
--safe-updates	safe-updates	Allow only UPDATE and DELETE statements that specify key values	
--secure-auth	secure-auth	Do not send passwords to the server in old (pre-4.1.1) format	
--select_limit=value	select_limit	The automatic limit for SELECT statements when using --safe-updates	
--show-warnings	show-warnings	Show warnings after each statement if there are any	
--sigint-ignore	sigint-ignore	Ignore SIGINT signals (typically the result of typing Control-C)	
--silent	silent	Silent mode	
--skip-auto-rehash	skip-auto-rehash	Disable automatic rehashing	
--skip-column-names	skip-column-names	Do not write column names in results	
--skip-line-numbers	skip-line-numbers	Skip line numbers for errors	
--skip-named-commands	skip-named-commands	Disable named mysql commands	
--skip-pager	skip-pager	Disable paging	
--skip-reconnect	skip-reconnect	Disable reconnecting	
--socket=path	socket	For connections to localhost	
--ssl-ca=file_name	ssl-ca	The path to a file that contains a list of trusted SSL CAs	
--ssl-capath=directory_name	ssl-capath	The path to a directory that contains trusted SSL CA certificates in PEM format	
--ssl-cert=file_name	ssl-cert	The name of the SSL certificate file to use for establishing a secure connection	
--ssl-cipher=cipher_list	ssl-cipher	A list of allowable ciphers to use for SSL encryption	
--ssl-key=file_name	ssl-key	The name of the SSL key file to use for establishing a secure connection	
--ssl-verify-server-cert	ssl-verify-server-cert	The server's Common Name value in its certificate is verified against the hostname used when connecting to the server	
--table	table	Display output in tabular format	
--tee=file_name	tee	Append a copy of output to the given file	
--unbuffered	unbuffered	Flush the buffer after each query	

Format	Config File	Description	Introduction
--user=user_name	user	The MySQL username to use when connecting to the server	
--verbose		Verbose mode	
--version		Display version information and exit	
--vertical	vertical	Print query output rows vertically (one line per column value)	
--wait	wait	If the connection cannot be established, wait and retry instead of aborting	
--xml	xml	Produce XML output	

C.2. `mysql` Commands

C.3. `mysqladmin` Options

Table C.2. `mysqladmin` Option Reference

Format	Config File	Description	Introduction
--compress	compress	Compress all information sent between the client and the server	
--connect_timeout=seconds	connect_timeout	The number of seconds before connection timeout	
--count=#	count	The number of iterations to make for repeated command execution	
--debug[=debug_options]	debug	Write a debugging log	
--debug-check	debug-check	Print debugging information when the program exits	5.1.21
--debug-info	debug-info	Print debugging information, memory and CPU statistics when the program exits	5.1.14
--default-character-set=charset_name	default-character-set	Use charset_name as the default character set	
--force	force	Continue even if an SQL error occurs	
--help		Display help message and exit	
--host=host_name	host	Connect to the MySQL server on the given host	
--no-beep	no-beep	Do not beep when errors occur	5.1.17
--password[=password]	password	The password to use when connecting to the server	
--port=port_num	port	The TCP/IP port number to use for the connection	
--protocol={TCP|SOCKET|PIPE|MEMORY}	protocol	The connection protocol to use	
--relative	relative	Show the difference between the current and previous values when used with the --sleep option	
--shutdown_timeout=seconds	shutdown_timeout	The maximum number of seconds to wait for server shutdown	
--silent	silent	Silent mode	
--sleep=delay	sleep	Execute commands repeatedly, sleeping for delay seconds in between	
--socket=path	socket	For connections to localhost	

Format	Config File	Description	Introduction
--ssl-ca=file_name	ssl-ca	The path to a file that contains a list of trusted SSL CAs	
--ssl-capath=directory_name	ssl-capath	The path to a directory that contains trusted SSL CA certificates in PEM format	
--ssl-cert=file_name	ssl-cert	The name of the SSL certificate file to use for establishing a secure connection	
--ssl-cipher=cipher_list	ssl-cipher	A list of allowable ciphers to use for SSL encryption	
--ssl-key=file_name	ssl-key	The name of the SSL key file to use for establishing a secure connection	
--ssl-verify-server-cert	ssl-verify-server-cert	The server's Common Name value in its certificate is verified against the hostname used when connecting to the server	
--user=user_name,	user	The MySQL username to use when connecting to the server	
--verbose		Verbose mode	
--version		Display version information and exit	
--vertical	vertical	Print query output rows vertically (one line per column value)	
--wait	wait	If the connection cannot be established, wait and retry instead of aborting	

C.4. `mysqldump` Options

Table C.3. `mysqldump` Option Reference

Format	Config File	Description	Introduction
--add-drop-database	add-drop-database	Add a DROP DATABASE statement before each CREATE DATABASE statement	
--add-drop-table	add-drop-table	Add a DROP TABLE statement before each CREATE TABLE statement	
--add-locks	add-locks	Surround each table dump with LOCK TABLES and UNLOCK TABLES statements	
--all-databases	all-databases	Dump all tables in all databases	
--allow-keywords	allow-keywords	Allow creation of column names that are keywords	
--all-tablespaces	all-tablespaces	Adds to a table dump all SQL statements needed to create any tablespaces used by an NDB Cluster table	5.1.6
--comments	comments	Add comments to the dump file	
--compact	compact	Produce less verbose output	
--compatible=name[,name,...]	compatible	Produce output that is more compatible with other database systems or with older MySQL servers	
--complete-insert	complete-insert	Use complete INSERT statements that include column names	
--create-options	create-options	Include all MySQL-specific table options in the CREATE TABLE statements	
--databases	databases	Dump several databases	
--debug[=debug_options]	debug	Write a debugging log	
--debug-check	debug-check	Print debugging information when the program exits	5.1.21
--debug-info	debug-info	Print debugging information, memory and CPU statistics when the program exits	5.1.14
--delayed-insert	delayed-insert	Write INSERT DELAYED statements rather than INSERT state-	

Format	Config File	Description	Introduction
		ments	
--delete-master-logs	delete-master-logs	On a master replication server, delete the binary logs after performing the dump operation	
--disable-keys	disable-keys	For each table, surround the INSERT statements with disable and enable keys statements	
--dump-date	dump-date	Include dump date in "Dump completed on" comment if --comments is given	5.1.23
-E	events	Dump events from the dumped databases	
--extended-insert	extended-insert	Use multiple-row INSERT syntax that include several VALUES lists	
--fields-enclosed-by=string	fields-enclosed-by	This option is used with the -T option and has the same meaning as the corresponding clause for LOAD DATA INFILE	
--fields-escaped-by	fields-escaped-by	This option is used with the -T option and has the same meaning as the corresponding clause for LOAD DATA INFILE	
--fields-option-ally-enclosed-by=string	fields-option-ally-enclosed-by	This option is used with the -T option and has the same meaning as the corresponding clause for LOAD DATA INFILE	
--fields-termin-ated-by=string	fields-terminated-by	This option is used with the -T option and has the same meaning as the corresponding clause for LOAD DATA INFILE	
--lock-all-tables	first-slave	Deprecated. Now renamed to --lock-all-tables	
--flush-logs	flush-logs	Flush the MySQL server log files before starting the dump	
--flush-privileges	flush-privileges	Emit a FLUSH PRIVILEGES statement after dumping the mysql database	
--help		Display help message and exit	
--hex-blob	hex-blob	Dump binary columns using hexadecimal notation (for example, 'abc' becomes 0x616263)	
--ig-nore-ta-ble=db_name.tbl_name	ignore-table	Do not dump the given table	
--insert-ignore	insert-ignore	Write INSERT statements with the IGNORE option	
--lines-termin-ated-by=string	lines-terminated-by	This option is used with the -T option and has the same meaning as the corresponding clause for LOAD DATA INFILE	
--lock-all-tables	lock-all-tables	Lock all tables across all databases	
--lock-tables	lock-tables	Lock all tables before dumping them	
--log-error=file_name	log-error	Append warnings and errors to the named file	5.1.18
--master-data[=value]	master-data	Write the binary log filename and position to the output	
--max_allowed_packet=value	max_allowed_packet	The maximum packet length to send to or receive from the server	
--net_buffer_length=value	net_buffer_length	The buffer size for TCP/IP and socket communication	
--no-autocommit	no-autocommit	Enclose the INSERT statements for each dumped table within SET AUTOCOMMIT=0 and COMMIT statements	
--no-create-db	no-create-db	This option suppresses the CREATE DATABASE statements	

Format	Config File	Description	Introduction
--no-create-info	no-create-info	Do not write CREATE TABLE statements that re-create each dumped table	
--no-data	no-data	Do not write any table row information (that is, do not dump table contents)	
--no-set-names	no-set-names	Turn off complete-insert	
--opt	opt	This option is shorthand; it is the same as specifying --add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset.	
--order-by-primary	order-by-primary	Sorts each table's rows by its primary key, or by its first unique index	
--quick	quick	Retrieve rows for a table from the server a row at a time	
--quote-names	quote-names	Quote database, table, and column names within backtick characters	
--replace	replace	Write REPLACE statements rather than INSERT statements	
--result-file=file	result-file	Direct output to a given file	
-R	routines	Dump stored routines (functions and procedures) from the dumped databases	
--set-charset	set-charset	Add SET NAMES default_character_set to the output	
--single-transaction	single-transaction	This option issues a BEGIN SQL statement before dumping data from the server	
--skip-add-drop-table	skip-add-drop-table	Do not add	
--skip-add-locks	skip-add-locks	Do not add locks	
--skip-comments	skip-comments	Do not add comments to the dump file	
--skip-compact	skip-compact	Turn off compact	
--skip-disable-keys	skip-disable-keys	Do not disable keys	
--skip-extended-insert	skip-extended-insert	Turn off extended-insert	
--skip-opt	skip-opt	Turn off the options set by opt	
--skip-quick	skip-quick	Do not retrieve rows for a table from the server a row at a time	
--skip-quote-names	skip-quote-names	Turn off quote names	
-skip-charset	skip-set-charset	Suppress the SET NAMES statement	
--skip-triggers	skip-triggers	Turn off triggers	
--skip-tz-utc	skip-tz-utc	Turn off tz-utc	
--ssl-ca=file_name	ssl-ca	The path to a file that contains a list of trusted SSL CAs	
--ssl-capath=directory_name	ssl-capath	The path to a directory that contains trusted SSL CA certificates in PEM format	
--ssl-cert=file_name	ssl-cert	The name of the SSL certificate file to use for establishing a secure connection	
--ssl-cipher=cipher_list	ssl-cipher	A list of allowable ciphers to use for SSL encryption	
--ssl-key=file_name	ssl-key	The name of the SSL key file to use for establishing a secure connection	
--ssl-verify-server-cert	ssl-verify-server-cert	The server's Common Name value in its certificate is verified against the hostname used when connecting to the server	
--tab=path	tab	Produce tab-separated data files	
--tables	tables	Override the --databases or -B option	
--triggers	triggers	Dump triggers for each dumped table	
--tz-utc	tz-utc	Add SET TIME_ZONE='+00:00' to the dump file	

Format	Config File	Description	Introduction
--verbose		Verbose mode	
--version		Display version information and exit	
--where='where_condition'	where	Dump only rows selected by the given WHERE condition	
--xml	xml	Produce XML output	

Appendix D. GNU Free Documentation License

Version 1.2, November 2002

Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor,
Boston,
MA
02110-1301
USA

Version 1.2, November 2002

D.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

D.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

D.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

D.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

D.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

GNU FDL Modification Conditions

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous ver-

sion if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

D.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

D.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

D.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

D.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

D.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

D.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

D.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> **Sample Invariant Sections list**
>
> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> **Sample Invariant Sections list**
>
> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.