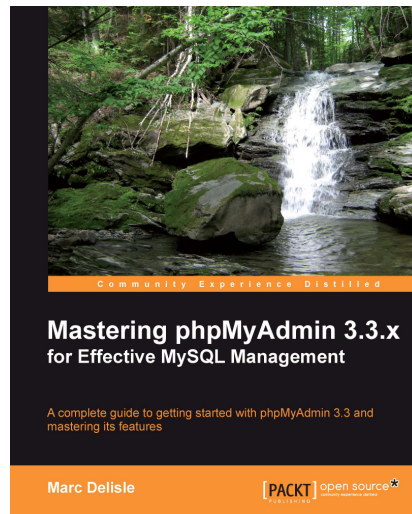




# Mastering phpMyAdmin 3.3.x for Effective MySQL Management

**Marc Delisle**



## Chapter No. 4 "Taking First Steps"

## In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.4 "Taking First Steps"

A synopsis of the book's content

Information on where to buy this book

## About the Author

**Marc Delisle** was awarded "MySQL Community Member of the year 2009" because of his involvement with phpMyAdmin. He started to contribute to the project in December 1998, when he developed the multi-language version. He is still involved with phpMyAdmin as a developer and project administrator.

Marc is a system administrator at Cegep de Sherbrooke, Québec, Canada. He has taught networking, security, and web application development. In one of his classes, he was pleased to meet a phpMyAdmin user from Argentina. Marc lives in Sherbrooke with his wife and they enjoy spending time with their four children.

This book was Marc's first one and was quickly followed by *Creating your MySQL Database: Practical Design Tips and Techniques*, also published by Packt Publishing.

---

I am truly grateful to Louay Fatoohi who approached me for this book project, and to the Packt team whose sound comments were greatly appreciated during the production. My thanks also go to the excellent reviewers Kai 'Oswald' Seidler, Ben Dodson, and Michal Čihař. Their sharp eyes helped in making this book clearer and more complete.

Finally, I wish to thank all contributors to phpMyAdmin's source code, translations, and documentation; their dedication to this project continues to push me forward.

---

### For More Information:

[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](http://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)

# Mastering phpMyAdmin 3.3.x for Effective MySQL Management

phpMyAdmin is an open source tool written in PHP. It handles the administration of MySQL over the World Wide Web (WWW). It can perform various tasks, such as creating, modifying, or deleting databases, tables, fields, or rows. It can also execute SQL statements and manage users and their permissions. When it comes to exploiting phpMyAdmin to its full potential, even experienced developers and system administrators search for tutorials to accomplish their tasks.

Mastering phpMyAdmin 3.3.x for Effective MySQL Management is an easy-to-read, step-by-step practical guide that walks you through every facet of this legendary tool—phpMyAdmin—and takes you a step ahead in taking full advantage of its potential. This book is filled with illustrative examples that will help you to understand every phpMyAdmin feature in detail.

This book jump starts with installing and configuring phpMyAdmin, and then looks into phpMyAdmin's features. This is followed by configuring authentication in phpMyAdmin, and setting parameters that influence the interface as a whole. You will first create two basic tables, and then edit and delete data, tables, and databases. As backups are crucial to a project, you will create up-to-date backups and take intermediary snapshots during development and production phases. Then you will look into importing the data that you have exported. You will also explore the various search mechanisms, and query across multiple tables.

Then, you will learn some advanced features, such as defining inter-table relations and installing the linked-tables infrastructure. Some queries are out of the scope of the interface; you will enter SQL commands to accomplish these tasks.

You will also learn some new features introduced in version 3.3.x, such as synchronizing databases on different servers, and managing MySQL replication in order to improve performance and data security. You will also store queries as bookmarks for their quick retrieval. Towards the end of the book you will learn to document your database, track changes made to the database, and manage user accounts using phpMyAdmin server management features.

This book is an upgrade from the previous version that covered phpMyAdmin Version 3.1. Version 3.3.x introduced features such as new import and export modules, tracking changes, synchronizing structure and data between servers, providing support for replication.

**For More Information:**

**[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](http://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)**

## What This Book Covers

*Chapter 1, Getting Started with phpMyAdmin*, gives us the reasons why we should use phpMyAdmin as a means of managing MySQL databases. It then covers the downloading and installation procedures for phpMyAdmin.

*Chapter 2, Configuring Authentication and Security*, provides an overview of various authentication types used in phpMyAdmin. It then covers the security issues related to the phpMyAdmin installation.

*Chapter 3, Over Viewing the Interface*, gives us an overview of the phpMyAdmin interface. This includes the login panel, the navigation and main panels in both Light mode and Full mode, and the Query window.

*Chapter 4, Taking First Steps*, is all about database creation. It teaches us how to create a table, how to insert data manually, and how to sort the data.

*Chapter 5, Changing Data and Structure*, covers the various aspects of data editing in phpMyAdmin. It teaches us how to handle NULL values, multi-row editing, and data deletion. Finally, it explores the subject of changing the structure of tables, focusing on editing field attributes and index management.

*Chapter 6, Exporting Structure and Data (Backup)*, deals with backups and exports. It lists various ways to trigger an export, available export formats, the options associated with export formats, and the various places where the export files can be sent.

*Chapter 7, Importing Structure and Data*, tells us how to bring back exported data that was created for backup and transfer purposes. It covers the various options available in phpMyAdmin to import data, and different mechanisms involved in importing SQL and CSV files. Finally, it covers the limitations that may be faced while importing files, and the ways to overcome these limitations.

*Chapter 8, Searching Data*, presents the mechanisms that are useful for searching data effectively.

*Chapter 9, Performing Table and Database Operations*, covers ways to perform some operations that influence and can be applied on entire tables or databases as a whole. Finally, it deals with table maintenance operations for table repair and optimization.

*Chapter 10, Benefiting from the Relational System*, is where we start covering the advanced features of phpMyAdmin. The chapter explains how to define inter-table relations. It also explains how to install the linked-tables infrastructure—a prerequisite for the advanced features.

**For More Information:**

**[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](http://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)**

*Chapter 11, Entering SQL Commands*, teaches us how to enter our own SQL commands. The chapter also covers the Query window—the window used to edit an SQL query. Finally, it also shows us how to obtain the history of typed commands.

*Chapter 12, Generating Multi-table Queries*, covers the multi-table query generator, which allows us to produce these queries without actually typing them.

*Chapter 13, Synchronizing Data and Supporting Replication*, teaches us how to synchronize databases on the same server, or from one server to another one. It then covers how to manage MySQL replication.

*Chapter 14, Using Bookmarks*, covers one of the features of the linked-tables infrastructure. It explains how to record bookmarks and how to manipulate them. Finally, it covers how to pass parameters to bookmarks.

*Chapter 15, Documenting the System*, gives an overview of how to produce documentation that explains the structure of a database, by using the tools offered by phpMyAdmin.

*Chapter 16, Transforming Data Using MIME*, explains how to apply transformations to data in order to customize its format at view time.

*Chapter 17, Supporting MySQL 5.0 and 5.1*, covers phpMyAdmin's support for the MySQL features that are new in these versions.

*Chapter 18, Tracking Changes*, teaches us how to record structure and data changes done from the phpMyAdmin interface.

*Chapter 19, Administrating the MySQL Server with phpMyAdmin*, is about the administration of a MySQL server, focusing on user accounts and privileges. The chapter discusses how a system administrator can use phpMyAdmin's server management features for day-to-day user account maintenance, server verification, and server protection.

*Appendix A, The History of phpMyAdmin*, provides a history of the project, from its roots back in 1998 through the project re-launch in 2001, and its subsequent evolution.

*Appendix B, Troubleshooting and Support*, explains how to troubleshoot phpMyAdmin by examining some of its error messages, and proposing appropriate solutions. It also explains how to interact with the development team for support, bug reports, and contributions.

**For More Information:**

**[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](http://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)**

# 4

## Taking First Steps

Having seen the overall layout of phpMyAdmin's panel, we are ready to create a database, create our first table, insert some data into it, and browse it. These first steps are intentionally simple, but they will give you the foundation on which more complex operations will be achieved later. At the end of the chapter, we will have at our disposition the two basic tables on which the remaining exercises are based.

### Creating a database

Before creating a table, we must ensure that we have a database for which the MySQL server's administrator has given us the `CREATE` privilege. Various possibilities exist:

- The administrator has already created a database for us, and we see its name in the navigation panel; we don't have the right to create an additional database
- We have the right to create databases from phpMyAdmin
- We are on a shared host, and the host provider has installed a general Web interface (for example, cPanel) to create MySQL databases and accounts; in this case, we should visit this web interface now and ensure that we have created at least one database and one MySQL account.


Note that a configuration parameter, `$cfg['ShowCreateDb']`, controls the display of the **Create new database** dialog. By default, it's set to `true`, which shows the dialog.

**For More Information:**

[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](http://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)

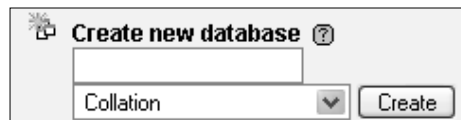
## No privileges?

If you do not have the required privileges to create a database, the home page displays a **No privileges** message under the **Create new database** label. This means that you must work with the databases that have been already created for you, or ask the MySQL server's administrator to give you the necessary `CREATE` privilege.

 If you are the MySQL server's administrator, refer to *Chapter 19, Administrating the MySQL Server with phpMyAdmin*.

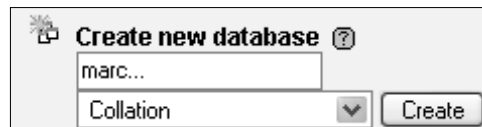
## First database creation is authorized

If phpMyAdmin detects that we have the right to create a database, the home page appears as shown in the following figure:



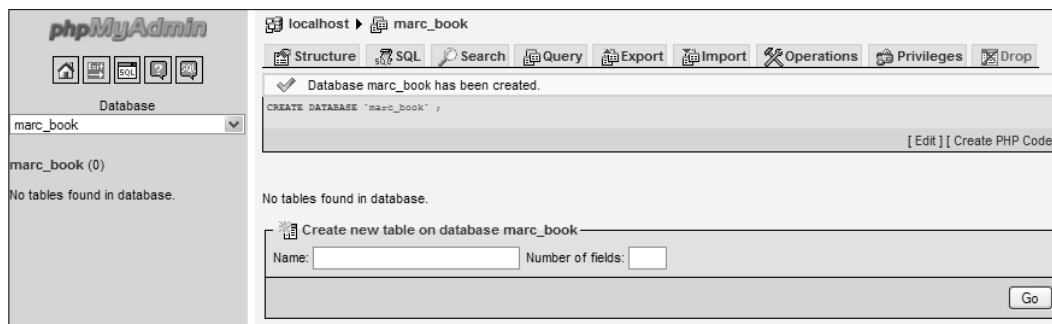
In the input field, a suggested database name appears if the `$cfg['SuggestDBName']` parameter is set to `TRUE` — which is the default setting. The suggested database name is built according to the privileges we possess.

If we are restricted to the use of a prefix, the prefix might be suggested in the input field. (A popular choice for this prefix is the username, which may or may not be followed by an underscore character.) Note that, in this case, the prefix is followed by an ellipsis mark, added by phpMyAdmin. We should remove this ellipsis mark and complete the input field with an appropriate name.




The **Collation** choice can be left unchanged for now. With this dialog, we could pick a default character set and collation for this database. This setting can be changed later (see *Chapter 9, Performing Table and Database Operations*, for more information on this).

We will assume here that we have the right to create a database named **marc\_book**. We enter **marc\_book** in the input field and click on **Create**. Once the database has been created, we will see the following screen:



Notice the following:

- The title of the main panel has changed to reflect the fact that we are now located in this database.
- A confirmation message regarding the creation is displayed.
- The navigation panel has been updated; we see **marc\_book (0)**. Here, the name indicates that the **marc\_book** database has been created, and the number **0** indicates that it contains no tables.
- By default, the SQL query sent to the server by phpMyAdmin to create the database is displayed in color.

 phpMyAdmin displays the query it generated, because `$cfg['ShowSQL']` is set to `TRUE`. Looking at the generated queries can be a good way of learning SQL.

As the generated queries can be large and take much of the on-screen room, the `$cfg['MaxCharactersInDisplayedSQL']` acts as a limit. Its default value of 1000 should be a good balance between seeing too few and seeing too many of the queries, especially when performing large imports.

It's important to examine the phpMyAdmin feedback to ascertain the validity of the operations that we make through the interface. In this way, we can detect errors such as typos in the names, or the creation of a table in the wrong database. phpMyAdmin retrieves error messages from the MySQL server and displays them in the interface.



## Creating our first table

Now that we have a new database, it's time to create a table in it. The example table we will create is named **book**.

## Choosing the fields

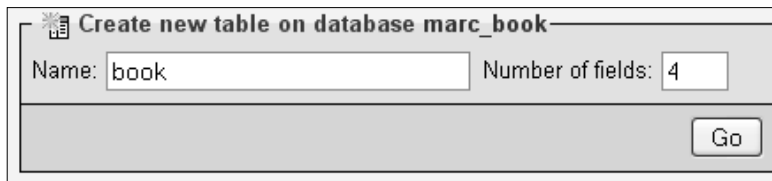
Before creating a table, we should plan the information we want to store. This is usually done during database design. In our case, a simple analysis leads us to the following book-related data we want to keep:

- International Standard Book Number (ISBN)
- Title
- Number of pages
- Author identification

For now, it's not important to have the complete list of fields (or columns) for our **book** table. We will modify it by prototyping the application now and refining it later. At the end of the chapter, we will add a second table, **author**, containing information about each author.

## Creating a table

We have chosen our table name and we know the number of fields. We enter this information in the **Create new table** dialog and click on **Go** to start creating the table. At this point it does not matter if the number of fields is exactly known, as a subsequent panel will allow the addition of fields when creating the table:



The image shows a dialog box titled "Create new table on database marc\_book". It has two input fields: "Name:" with the text "book" and "Number of fields:" with the number "4". At the bottom right, there is a button labeled "Go".

We then see a panel specifying field information. As we asked for **4** fields, we get four rows. Each row refers to information specific to one field. The following image represents the left side for this panel:

localhost ▶ marc\_book ▶ book

Field	Type ?	Length/Values <sup>1</sup>	Default <sup>2</sup>
<input type="text"/>	INT	<input type="text"/>	None
<input type="text"/>	INT	<input type="text"/>	None
<input type="text"/>	INT	<input type="text"/>	None
<input type="text"/>	INT	<input type="text"/>	None

Table comments:  Storage Engine: ? MyISAM Collation:

And the next one represents the right side:

Collation	Attributes	Null	Index	A_I	Comments
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---	<input type="checkbox"/>	<input type="text"/>

Had we asked to create a table with less than four fields, the display would be oriented vertically – we'll see an example of that at the end of the chapter.

The MySQL documentation explains the valid characters for the table and field names (if we search for **Legal names**). This may vary depending on the MySQL version. Usually, any character that is allowed in a filename (except the dot and the slash) is acceptable in a table name, and the length of the name must not exceed 64 characters. The 64-character limit exists for field names as well, but we can use any character.

We enter our field names in the **Field** column. Each field has a type, and the most commonly-used types are located at the beginning of the drop-down list.

The **VARCHAR** (variable character) type is widely used when the field content is alphanumeric, because the contents will occupy only the space needed for it. This type requires a maximum length, which we specify. If we forget to do so, a small pop-up message reminds us later when we save the page. For the page count and the author identification, we have chosen **INT** type (integer), as depicted in the following screenshot:

Field	Type ?	Length/Values <sup>1</sup>
isbn	VARCHAR	25
title	VARCHAR	100
page_count	INT	
author_id	INT	

There are other attributes for fields, but we will leave them empty in this example. You might notice the **Add 1 Field(s)** dialog at the bottom of the screen. We can use it to add some fields to this table creation panel by entering the appropriate value and hitting **Go**. The number of rows will change according to the new number of fields, leaving the information already entered in the first four fields intact. Before saving the page, let's define some keys.

## Choosing keys

A table should normally have a primary key (a field with unique content that represents each row). Having a primary key is recommended for row identification, better performance, and possible cross-table relations. A good value here is the ISBN; so, in the **Index** dialog we select **PRIMARY** for the **isbn** field. Other possibilities for index type include **INDEX**, **UNIQUE**, and **FULLTEXT** (more on this in *Chapter 5, Changing Data and Structure*).































Index management (also referred to as Key management) can be done at initial table creation, or later in the **Structure** subpage of Table view.








To improve the speed of the queries that we will make by author ID, we should add an index on this field. The rightmost part of our screen now looks like this:




Index	A I	Comments
PRIMARY	<input type="checkbox"/>	
—	<input type="checkbox"/>	
—	<input type="checkbox"/>	
INDEX	<input type="checkbox"/>	



At this point, we could pick a different **Storage Engine** from the corresponding drop-down menu. However, for the time being, we will just accept the default storage engine.


Now we are ready to create the table by clicking on **Save**. If all goes well, the next screen confirms that the table has been created; we are now in the **Structure** subpage of the Table view:





	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	isbn	varchar(25)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	title	varchar(100)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	page_count	int(11)			No	None		      
<input type="checkbox"/>	author_id	int(11)			No	None		      

☐ Check All / ☐ Uncheck All *With selected:*       

 Print view  Propose table structure 

 Add 1 field(s) ☒ At End of Table ☐ At Beginning of Table ☐ After isbn  Go

**Indexes:** 

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
 	PRIMARY	BTREE	Yes	No	isbn	0	A		
 	author_id	BTREE	No	No	author_id	0	A		

Of the various tabs leading to other subpages, some are not active, because it would not make sense to browse or search a table if there are no rows in it. However, it would be acceptable to export a table, as we can export a table's structure even if it contains no data.

## Inserting data manually

Now that we have a table, let's put some data in it manually. Before we do that, here are some useful references on data manipulation within this book:

- *Chapter 5, Changing Data and Structure* explains how to change data and structure
- *Chapter 7, Importing Data and Structure* explains how to import data from existing files
- *Chapter 9, Performing Table and Database Operations* explains how to copy data from other tables
- *Chapter 10, Benefiting from the Relational System* explains the relational system (in our case, we will want to link to the **author** table)

For now, click on the **Insert** link, which will lead us to the data-entry (or edit) panel. This screen has room to enter information for two rows, that is two books in our example. This is because the default value of `$cfg['InsertRows']` is 2. In the lower part of the screen, the dialog **Restart insertion with 2 rows** can be used if the default number of rows does not suit our needs. But beware, any data already on the screen would be lost. By default, the **Ignore** checkbox is ticked, which means that the second group of fields will be ignored. But as soon as we enter some information in one field of this group and exit the field, the **Ignore** box is automatically unchecked if JavaScript is enabled in the browser.

We can enter the following sample information for two books:

- ISBN: 1-234567-89-0, title: A hundred years of cinema (volume 1), 600 pages, author ID: 1
- ISBN: 1-234567-22-0, title: Future souvenirs, 200 pages, author ID: 2

We start by entering data for the first and the second rows. The **Value** column width obeys the maximum length for the character fields. For this example, we leave the lower drop-down selector with its default value of **Insert as new row**. We then click on **Go** to insert the data. There is a **Go** button after each set of columns that represent a row, and another one on the lower part of the screen. All of these have the same effect of saving the entered data, but are provided for convenience.

Field	Type	Function	Null	Value
isbn	varchar(25)			1-234567-89-0
title	varchar(100)			A hundred years of cinema (volume 1)
page_count	int(11)			600
author_id	int(11)			1

Go

☐ Ignore

Field	Type	Function	Null	Value
isbn	varchar(25)			1-234567-22-0
title	varchar(100)			Future souvenirs
page_count	int(11)			200
author_id	int(11)			2

Go

Insert as new row  Go back to previous page

Restart insertion with  rows

<sup>1</sup> Use TAB key to move from value to value, or CTRL+arrows to move anywhere

If our intention had been to enter data for more books after these two, we would have selected **Insert another new row** from the first drop-down menu before clicking on **Go**. This would then insert the data we have provided and reload the screen so that it's ready for us to insert more.

## Data entry panel tuning for CHAR and VARCHAR

By default, phpMyAdmin displays an input field on a single line for the field types CHAR and VARCHAR. This is controlled by setting `$cfg['CharEditing']` to 'input'. Sometimes, we may want to insert line breaks (new lines) within the field. This can be done by changing `$cfg['CharEditing']` to 'textarea'. This is a global setting, and will apply to all the fields of all the tables, for all users of this copy of phpMyAdmin. In this mode, the insertion of line breaks may be done manually with the *Enter* key, or by copying and pasting lines of text from another on-screen source.

We can tune the number of columns and rows of this text area with:

```
$cfg['CharTextareaCols'] = 40;
$cfg['CharTextareaRows'] = 2;
```

Here, 2 for `$cfg['CharTextareaRows']` means that we should be able to see at least two lines before the browser starts to display a vertical scroll bar. These settings apply to all **CHAR** and **VARCHAR** fields. Applying those settings would generate a different **Insert** screen, as follows:

Value

3-343434-55-6

Computers:  
the complete history



With this entry mode, the maximum length of each field no longer applies visually. It would be enforced by MySQL at insert time.

## Browse mode

There are many ways to enter Browse mode. In fact, it's used each time the query results are displayed. We can enter this mode by clicking on the table name in the navigation panel, or by clicking on **Browse** when we are in the Table view for a specific table:

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Showing rows 0 - 1 (2 total, Query took 0.0003 sec)

```
SELECT *
FROM 'book'
LIMIT 0 , 30
```

☐ Profiling ☐ Edit ☐ Explain SQL ☐ Create PHP Code ☐ Refresh

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

	isbn	title	page_count	author_id
<input type="checkbox"/>	1-234567-89-0	A hundred years of cinema (volume 1)	600	1
<input type="checkbox"/>	1-234567-22-0	Future souvenirs	200	2

☐ Check All / ☐ Uncheck All With selected:

Show : 30 row(s) starting from record # 0

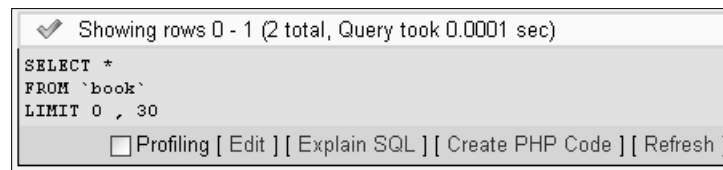
in horizontal mode and repeat headers after 100 cells

Query results operations

Print view Print view (with full texts) Export CREATE VIEW

## SQL query links

In the **Browse** results, the first part displayed is the query itself, along with a few links. The displayed links may vary depending on our actions and some configuration parameters.

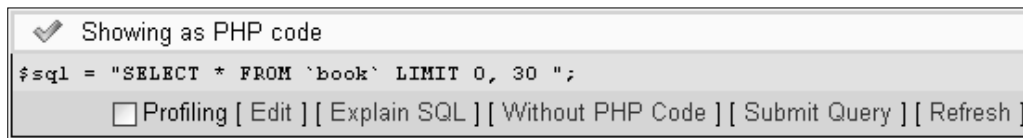


The **Profiling** checkbox is covered in *Chapter 17, Supporting MySQL 5.0 and 5.1*.

The **Edit** link appears if `$cfg['SQLQuery']['Edit']` is set to `TRUE`. Its purpose is to open the **Query window** so that you can edit this query (see *Chapter 11, Entering SQL Commands*, for more details).

**Explain SQL** is displayed if `$cfg['SQLQuery']['Explain']` is set to `TRUE`. We will see in *Chapter 5, Changing Data and Structure* what this link can be used for.

The **Create PHP Code** link can be clicked to reformat the query to the syntax expected in a PHP script. It can then be copied and pasted directly at the place where we need the query in the PHP script we are working on. Note that after a click, this link changes to **Without PHP Code**, which would bring back the normal query display. This link is available if `$cfg['SQLQuery']['ShowAsPHP']` is set to `TRUE`.



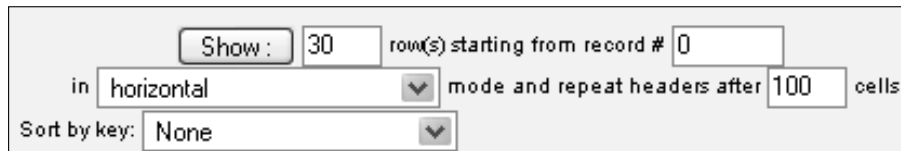
**Refresh** is used to execute the same query again. The results might change, as a MySQL server is a multi-user server, and other users or processes might be modifying the same tables. This link is shown if `$cfg['SQLQuery']['Refresh']` is set to `TRUE`.

All four of these parameters have a default value of `TRUE` in `config.inc.php`.



## Navigation bar

The Navigation bar is displayed at the top of the results and also at the bottom. Column headers can be repeated at certain intervals among results depending on the value entered in **repeat headers after....**



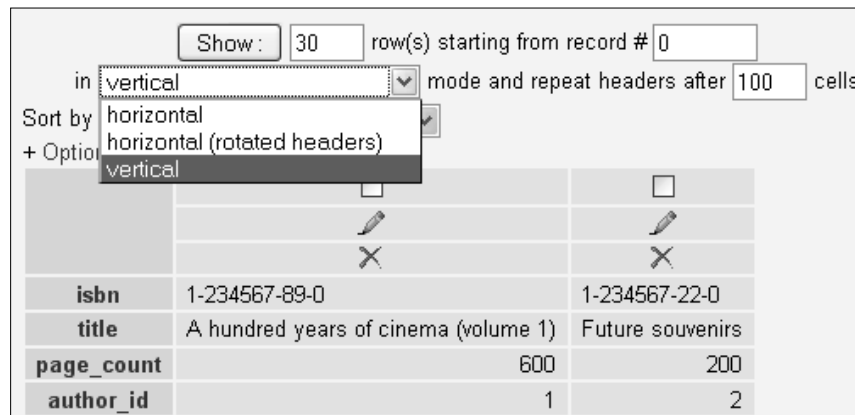
Navigation bar controls:

- Show: 30 row(s) starting from record # 0
- in horizontal mode and repeat headers after 100 cells
- Sort by key: None

The Navigation bar enables us to navigate from page to page, displaying an arbitrary number of records (or rows), starting at some point in the results. As we entered browse mode by clicking on **Browse**, the underlying query that generated the results includes the whole table. However, this is not always the case.

Notice that we are positioned at record number 0, and are seeing records in **horizontal** mode.

The default display mode is 'horizontal', as defined in `$cfg['DefaultDisplay']`. We can also set this to 'vertical' if we usually prefer this mode. Even if our preferred mode is set in this configuration parameter, we can always use the **Show** dialog to change it on the fly.







Navigation bar controls (vertical mode selected):





- Show: 30 row(s) starting from record # 0
- in vertical mode and repeat headers after 100 cells
- Sort by: horizontal
- + Options: horizontal (rotated headers), vertical

isbn	1-234567-89-0	1-234567-22-0
title	A hundred years of cinema (volume 1)	Future souvenirs
page_count	600	200
author_id	1	2

Another possibility for `$cfg['DefaultDisplay']` is the `'horizontalflipped'` value (which can also be selected on screen via the **horizontal (rotated headers)** choice), which rotates the column headers by 90 degrees. If we try this choice, another parameter, `$cfg['HeaderFlipType']`, plays a role. Its default value, `css`, displays true rotated headers. Not every browser supports this (at least Firefox and Chrome do not); however, Internet Explorer 8 and Opera do support this feature, and produce:

←T→			isbn	title	page_count	author_id
<input type="checkbox"/>			1-234567-89-0	A hundred years of cinema (volume 1)	600	1
<input type="checkbox"/>			1-234567-22-0	Future souvenirs	200	2

On other browsers, it seems the best we can achieve is by setting `$cfg['HeaderFlipType']` to `fake`.

←T→			i s b n	t i t l e	p a g e - c o u n t	a u t h o r - i d
<input type="checkbox"/>			1-234567-89-0	A hundred years of cinema (volume 1)	600	1
<input type="checkbox"/>			1-234567-22-0	Future souvenirs	200	2



We are currently using a table containing a small number of rows. With larger tables, we could see a more complete set of navigation buttons. To simulate this, let's use the **Show** dialog to change the default number of rows from **30** to **1**; we then click on **Show**. We can see that the navigation bar adapts itself:

Show : 1 row(s) starting from record # 1
> >>
Page number: 1

in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

←T→			isbn	title	page_count	author_id
<input type="checkbox"/>			1-234567-89-0	A hundred years of cinema (volume 1)	600	1

This time, there are buttons labeled <<, <, >, and >> for easy access to the first page, previous page, next page, and the last page of the results respectively. The buttons appear only when necessary; for example, the **first page** button is not displayed if we already are on the first page. These symbols are displayed in this manner as the default setting of `$cfg['NavigationBarIconic']` is `TRUE`. A `FALSE` here would produce buttons like **Next** and **End**, whereas a value of `'both'` would display **> Next** and **>> End**.

There is also a **Page number** drop-down menu, to go directly to one of the pages located near the current page. As there can be hundreds or thousands of pages, this menu is kept small and contains the commonly requested pages: A few page numbers before and after the current page, a few pages at the beginning and at the end, and a sample of page numbers based on a computed interval.

By design, phpMyAdmin always tries to give quick results, and one way to achieve this result is to add a `LIMIT` clause in `SELECT`. If a `LIMIT` clause is already there in the original query, phpMyAdmin will respect it. The default limit is 30 rows, set in `$cfg['MaxRows']`. If there are many users on the server, limiting the number of rows returned helps to keep the server load to a minimum.

Another button is available on the navigation bar, but must be activated by setting `$cfg['ShowAll']` to `TRUE`. It would be very tempting for users to use this button often. Hence, on a multi-user installation of phpMyAdmin, it's recommended that the button be left to its default value of disabled (`FALSE`). When enabled, the navigation bar is augmented with a **Show all** button. Clicking on this button retrieves all of the rows of the current results set, which might hit the execution time limit in PHP or a memory limit on the server; most browsers would also crash when asked to display thousands of rows. The exact number of rows that can be safely displayed cannot be predicted as it depends on the actual data present in columns and on the browser's capabilities.



If we enter a big number in the **Show...rows** dialog, the same results will be achieved (and we may face the same problems).

## Query results operations

A section labeled **Query results operations** is located under the results. This contains links to print the results (with or without the `FULL TEXT` columns), to export these results (see the *Exporting partial query results* section in *Chapter 6, Exporting Data and Structure (Backup)*), or to create a view from this query (more on this in *Chapter 17, Supporting MySQL 5.0 and 5.1*).

## Sorting results

In SQL, we can never be sure of the order in which the data is retrieved, unless we explicitly sort the data. Some implementations of the retrieving engine may show results in the same order as the one in which data was entered, or according to a primary key. However, a sure way to get results in the order we want is by sorting them explicitly.





When browsing results are displayed, any column header can be clicked to sort on this column, even if it's not part of an index. Let's click on the **author\_id** column header:

←T→	isbn	title	page_count	author_id ▲
<input type="checkbox"/>  	1-234567-89-0	A hundred years of cinema (volume 1)	600	1
<input type="checkbox"/>  	1-234567-22-0	Future souvenirs	200	2

We can confirm that the sorting has occurred, by watching the SQL query at the top of the screen; it now contains an **ORDER BY** clause.

We now see a small red triangle pointing upwards beside the **author\_id** header. This means that the current sort order is "ascending". Moving the mouse cursor over the **author\_id** header makes the red triangle change direction, to indicate what will happen if we click on the header again – a sort by descending **author\_id**.

Another way to sort is by key. The **Sort** dialog shows all of the keys already defined. Here we see a key named **PRIMARY** – the name given to our primary key on the **isbn** field when we checked **Primary** for this field at creation time:

Sort by key: None ▼				
+ Options				
←T→		title	page_count	author_id
<input type="checkbox"/>  		undred years of cinema (volume 1)	600	1
<input type="checkbox"/>  	1-234567-22-0	Future souvenirs	200	2

This might be the only way to sort on multiple fields at once (for multi-field indexes).

The default initial sort order is defined in `$cfg['Order']`, with `ASC` for ascending, `DESC` for descending. The sort order can also be `SMART`, which means that fields of type `DATE`, `TIME`, `DATETIME`, and `TIMESTAMP` would be sorted in descending order, whereas other field types will be sorted in ascending order.

## Headwords

Because we can change the number of records displayed on a page, it's quite possible that we do not see all of the data. In this case, it would help to see headwords — indications about the first and last row of displayed data. This way, you can click on **Next** or **Previous** without scrolling to the bottom of the window.

However, which column should phpMyAdmin base this headwords generation on? A simple assumption has been made: If you click on a column's header to indicate your intention of sorting on this column, phpMyAdmin uses this column's data as a headword. For our current book table, we do not have enough data to clearly notice the benefits of this technique. However, we can nonetheless see that after a sort the top part of the screen now contains this message:





```
Showing rows 0 - 1 (2 total, Query took 0.0006 sec)
[author_id: 1 - 2]
```

Here, the message between square brackets means that **author\_id** number **1** is on the first displayed row and number **2** is on the last one.

## Color-marking rows





When moving the mouse between rows, the row background color can change to the color defined in `$cfg['BrowsePointerColor']`. This parameter can be found in `themes/themename/layout.inc.php`. To enable this, the browse pointer for all themes — `$cfg['BrowsePointerEnable']` — must be set to `TRUE` (the default) in `config.inc.php`.

It may be interesting to visually mark some rows when we have many columns in the table and must constantly scroll left and right to read data. Another option is to highlight the importance of some rows for personal comparison of data, or when showing data to people. Highlighting is done by clicking on the row. Clicking again removes the highlighting from the row. The chosen color is defined by `$cfg['BrowseMarkerColor']` (see `themes/themename/layout.inc.php`). This feature must be enabled by setting `$cfg['BrowseMarkerEnable']` to `TRUE`, this time in `config.inc.php`. This sets the feature for all themes. We can mark more than one row. Marking the row also activates the checkbox for this row.

← T →			isbn	title	page_count	author_id
<input checked="" type="checkbox"/>			1-234567-89-0	A hundred years of cinema (volume 1)	600	1
<input type="checkbox"/>			1-234567-22-0	Future souvenirs	200	2

## Limiting the length of each column

In the previous examples, we always saw the full contents of each column, as each column had a number of characters that was within the limit defined by `$cfg['LimitChars']`. This is a limit enforced on all non-numeric fields. If this limit was lower (say 10), the display would be as follows:

+ Options						
← T →			isbn	title	page_count	author_id
<input type="checkbox"/>			1-234567-8...	A hundred ...	600	1
<input type="checkbox"/>			1-234567-2...	Future sou...	200	2

This would help us see more columns at the same time (at the expense of seeing less of each column).

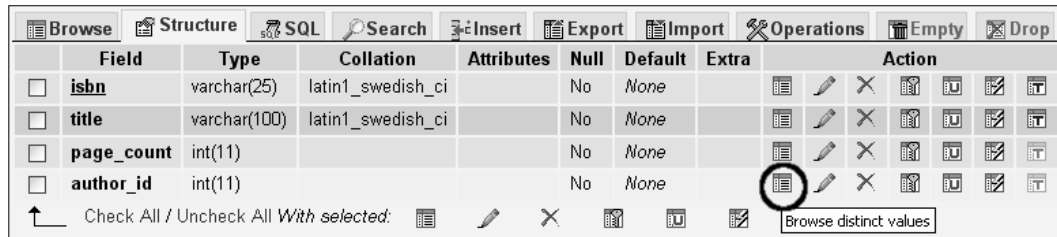
## Display options

In order to see the full texts, we will now make use of the **Options** slider, which reveals some display options. The option that concerns us at the moment is the **Partial Texts/Full Texts** pair; we can choose **Full Texts** to see all of the text that was truncated. Even if we elect not to change the `$cfg['LimitChars']` parameter, there will be a time when asking for full texts will be useful (when we work with the `TEXT` field type—more on this in *Chapter 5, Changing Data and Structure*).

A quicker way of seeing the full texts is to click on the big **T**, which is located just on top of the **Edit** and **Delete** icons. Another click on this **T** toggles the display from full to partial.

## Browsing distinct values

There is a quick way to display all distinct values and the number of occurrences for each value for each field. This feature is available on the **Structure** page. For example, we want to know how many different authors we have in our book table and how many books each one wrote. On the line describing the field we want to browse (here **author\_id**), we click on the **Browse distinct values** icon or link:



We have a limited test set, but can nonetheless see the results:

```
SELECT COUNT( * ) AS `Rows` , `author_id`
FROM `book`
GROUP BY `author_id`
ORDER BY `author_id`
LIMIT 0 , 30
```

Show: 30 row(s) starting from horizontal mode and  
Sort by key: None

+ Options

Rows	author_id
1	1
1	2

## Customizing the browse mode

The following are additional parameters that control the appearance of results. These parameters—except `$cfg['RepeatCells']`—are located in `themes/themenamename/layout.inc.php`.

Additional parameter	Effect on appearance of result
<code>\$cfg['Border']</code>	The HTML tables used to present results have no border by default because this parameter is set to 0; we can set this to a higher number (for example, 1 or 2) to add borders to the tables.
<code>\$cfg['ThBgcolor']</code>	The tables mentioned have headers with #D3DCE3 as the default background color.
<code>\$cfg['BgcolorOne']</code> , <code>\$cfg['BgcolorTwo']</code>	When displaying rows of results, two background colors are used alternately; by default, those are #CCCCCC and #DDDDDD.
<code>\$cfg['RepeatCells']</code>	When many rows of data are displayed, we may lose track of the meaning of each column. By default, at each 100th cell, the column headers are displayed.

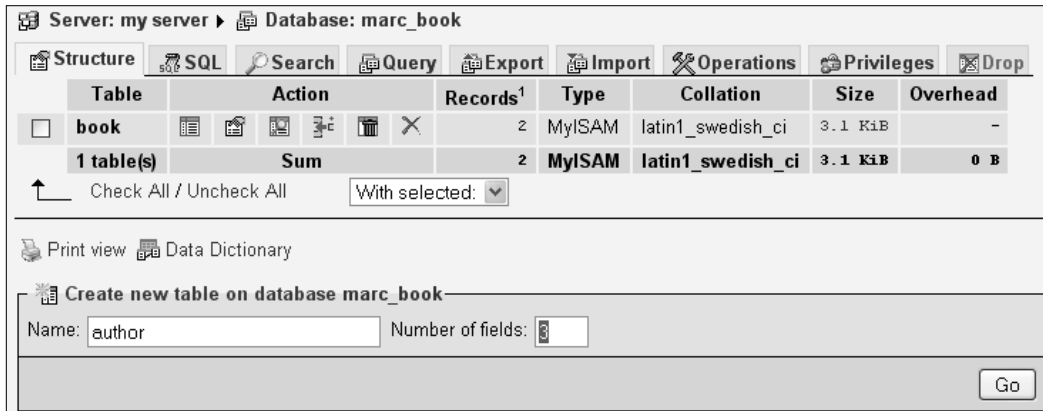
## Creating an additional table

In our (simple) design, we know that we need another table—the **author** table. The **author** table will contain:

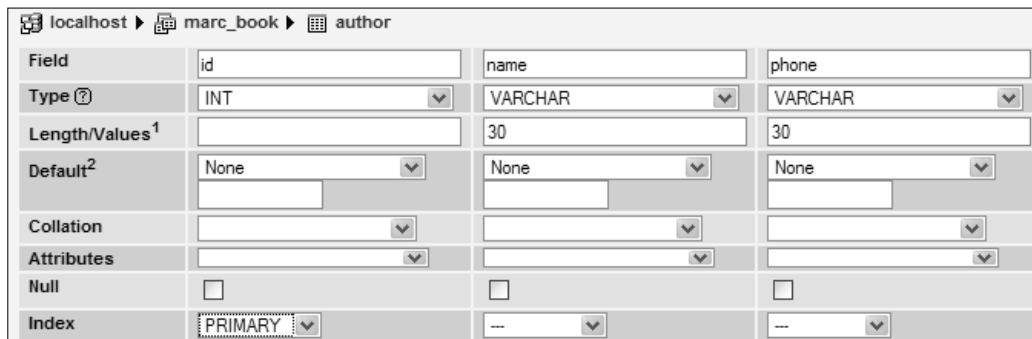
- Author identification
- Full name
- Phone number



To create this table, we must go back to the Database view. We click on **marc\_book** in the navigation panel, and request the creation of another table with three fields, as indicated in the following screenshot:



Using the same techniques used when creating the first table, we enter the data shown in the following screenshot:



As we have three fields or less, the display is now in vertical mode (see the *Vertical mode* section in *Chapter 5, Changing Data and Structure* for more details).

The field name **id**, which is our primary key in this new table, relates to the `author_id` field from the `book` table. After saving the table structure, we enter some data for authors 1 and 2. Use your imagination for this!

## Summary

This chapter explained how to create a database and tables, and how to enter data manually into the tables. It also covered how to confirm the presence of data by using the browse mode, which includes the SQL query links, navigation bar, sorting options, and row marking.

The next chapter explains how to edit data rows and covers the various aspects of deletion of rows, tables, and databases.

## Where to buy this book

You can buy Mastering phpMyAdmin 3.3.x for Effective MySQL Management from the Packt Publishing website: <https://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book>

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



[www.PacktPub.com](http://www.PacktPub.com)

**For More Information:**

[www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book](https://www.packtpub.com/mastering-phpmyadmin-3-3-x-for-effective-mysql-management/book)