

## Relationships and Using Designer in PhpMyAdmin

The first step is to enable Designer in phpmyadmin by editing the C:\XAMP\phpMyAdmin\config.default.php file.

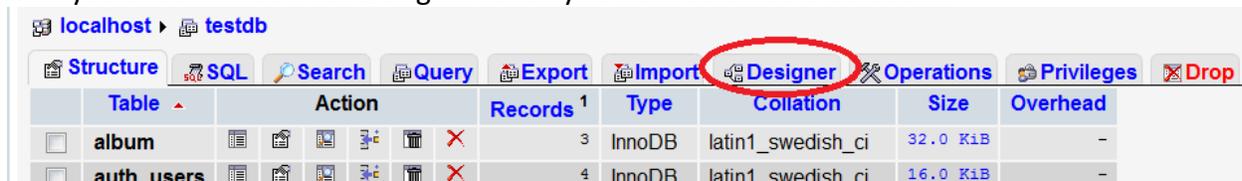
Change these starting on about line 260:

```
$cfg['Servers'][$i]['pmadb'] = '';  
$cfg['Servers'][$i]['bookmarktable'] = '';  
$cfg['Servers'][$i]['relation'] = '';  
$cfg['Servers'][$i]['table_info'] = '';  
$cfg['Servers'][$i]['table_coords'] = '';  
$cfg['Servers'][$i]['pdf_pages'] = '';  
$cfg['Servers'][$i]['column_info'] = '';  
$cfg['Servers'][$i]['history'] = '';  
$cfg['Servers'][$i]['designer_coords'] = '';
```

to

```
$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';  
$cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';  
$cfg['Servers'][$i]['relation'] = 'pma_relation';  
$cfg['Servers'][$i]['table_info'] = 'pma_table_info';  
$cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';  
$cfg['Servers'][$i]['pdf_pages'] = 'pma_pdf_pages';  
$cfg['Servers'][$i]['column_info'] = 'pma_column_info';  
$cfg['Servers'][$i]['history'] = 'pma_history';  
$cfg['Servers'][$i]['designer_coords'] = 'pma_designer_coords';
```

Now you should have a tab Designer when you have selected a database.



The screenshot shows the phpMyAdmin interface for a database named 'testdb'. The 'Designer' tab is highlighted with a red circle. Below the tabs, there is a table listing database tables with columns for Table, Action, Records, Type, Collation, Size, and Overhead.

Table	Action	Records	Type	Collation	Size	Overhead
album	[Icons]	3	InnoDB	latin1_swedish_ci	32.0 KiB	-
auth_users	[Icons]	4	InnoDB	latin1_swedish_ci	16.0 KiB	-

### Creating a One-to-Many Relationship

I have created two tables Auth\_users and Cities. I am going to create a **One-to Many** Relationship on these tables. One city entry can show up many times in the auth\_users table. For example you may have many users from Portland but only need one entry of Portland in the City table. The relationship will be joined on the city\_id. The city\_id will be the Primary Key in the city table (the ONE) and will join to the foreign key in the auth\_users table.



The screenshot shows the 'Structure' view of the 'city' table. The 'city\_id' field is highlighted with a yellow circle and is marked as the primary key. The 'city\_name' field is also visible. Below the table structure, the 'Relation view' tab is highlighted with a red circle.

Field	Type	Collation	Attributes	Null	Default	Extra	Action
city_id	int(11)			No	None		[Icons]
city_name	varchar(25)	latin1_swedish_ci		No	None		[Icons]

localhost > testdb > auth\_users

Structure SQL Search Insert Export Import Operations Empty Drop

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> f_name	varchar(50)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> l_name	varchar(50)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> username	varchar(25)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> password	varchar(100)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> city_id	int(11)			No	None		[Icons]

Check All / Uncheck All With selected: [Icons]

Select the Index on the auth\_users.city\_id field and the index created will show below. The values entered in the auth\_users must be the same values in the city table and the data types must match.

localhost > testdb > auth\_users

Structure SQL Search Insert Export Import Operations Empty Drop

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> f_name	varchar(50)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> l_name	varchar(50)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> username	varchar(25)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> password	varchar(100)	latin1_swedish_ci		Yes	NULL		[Icons]
<input type="checkbox"/> city_id	int(11)			No	None		[Icons] <b>Index</b>

Check All / Uncheck All With selected: [Icons]

#### Indexes:

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
[Icons]	city_id	BTREE	No	No	city_id	4	A		

Now go back to the database level on the top navigation and then select Designer.

localhost > **testdb**

Structure SQL Search Query Export Import **Designer** Operation

Table	Action	Records	Type	Collation	Size
<input type="checkbox"/> album	[Icons]	3	InnoDB	latin1_swedish_ci	32.0 B

Now you have a nice visual of the relationship. If the relationship was not formed automatically, select the join option and follow the prompts.



## Understanding JOINS

There are two basic types of SQL JOINS – **INNER JOINS** and **OUTER JOINS**. If you don't put INNER or OUTER keywords in front of the **SQL JOIN** keyword, then **INNER JOIN** is used.

The **INNER JOIN** will select all rows from both tables as long as there is a match between the columns being compared. This is going to be the most often used **JOIN**.

	f_name	l_name	username	password	city_id
<input type="checkbox"/>	wendy	plourde	wplourde	*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29	1
<input type="checkbox"/>	amy	plourde	wplourde	*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29	1
<input type="checkbox"/>	Sarah	plourde	splourde	*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29	2
<input type="checkbox"/>	Marc	plourde	test	*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29	2
<input type="checkbox"/>	Bob	Smith	bsmith	test	0
<input type="checkbox"/>	Betty	Smith	bsmith	test	0

	city_id	city_name
<input type="checkbox"/>	1	Portland
<input type="checkbox"/>	2	South Portland
<input type="checkbox"/>	3	Scarborough
<input type="checkbox"/>	4	Saco

An **INNER JOIN** on these tables will return the following:

Name	City
wendy	Portland
amy	Portland
Sarah	South Portland
Marc	South Portland

Note that Bob and Betty are not returned because they do not have a City\_id.

The second type of **SQL JOIN** is called **SQL OUTER JOIN** and it has 2 sub-types called **LEFT OUTER JOIN** and **RIGHT OUTER JOIN**.

The **LEFT OUTER JOIN** or simply **LEFT JOIN** selects all the rows from the first table listed after the FROM clause, no matter if they have matches in the second table. Shown below is the result of a **LEFT JOIN**.

Name	City
wendy	Portland
amy	Portland
Sarah	South Portland
Marc	South Portland
Bob	
Betty	

Name	City
wendy	Portland
amy	Portland
Sarah	South Portland
Marc	South Portland
	Scarborough
	Saco

The **RIGHT OUTER JOIN** or just **RIGHT JOIN** behaves exactly as **SQL LEFT JOIN**, except that it returns all rows from the second table (the right table in our **SQL JOIN** statement).

## Write the Script

```

1  <?php
2  $con = mysql_connect("localhost","root","");
3  if (!$con) {
4  die('Could not connect: ' . mysql_error());
5  }
6
7  mysql_select_db("testdb", $con);
8  $result = mysql_query("SELECT auth_users.f_name, city.city_name FROM auth_users INNER JOIN city ON auth_users.city_id =
city.city_id");
9  echo "<table border='1'>
10 <tr>
11 <th>Name</th>
12 <th>City</th>
13 </tr>";
14 while ($row = mysql_fetch_array($result)) {
15 echo "<tr>";
16 echo "<td>" . $row['f_name'] . "</td>";
17 echo "<td>" . $row['city_name'] . "</td>";
18 echo "</tr>";
19 }
20 echo "</table>";
21
22 mysql_close($con);
23 ?>

```