# 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.

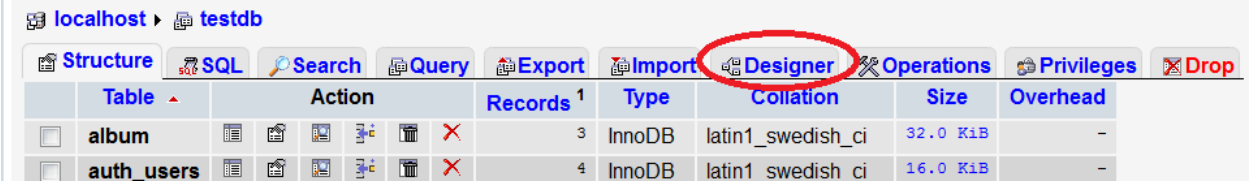
Change these starting on about line 260:

```
$cfg['Servers'][$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

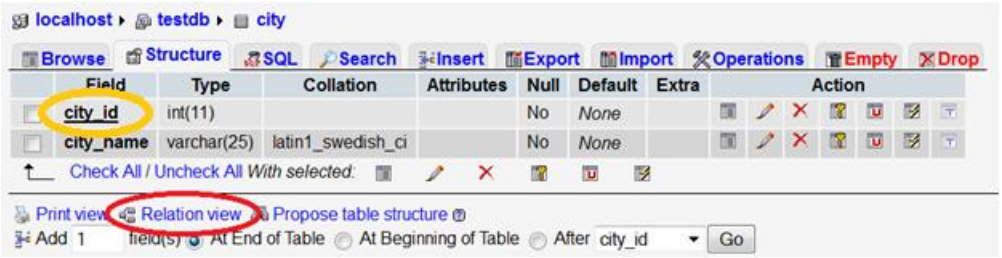
```
$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';
$cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';
$cfg['Servers'][$i]['relation'] = 'pma_relation';
$cfg['Servers'][$i]['table_info'] = 'pma_table_info';
$cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';
$cfg['Servers'][$i]['pdf_pages'] = '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';
```

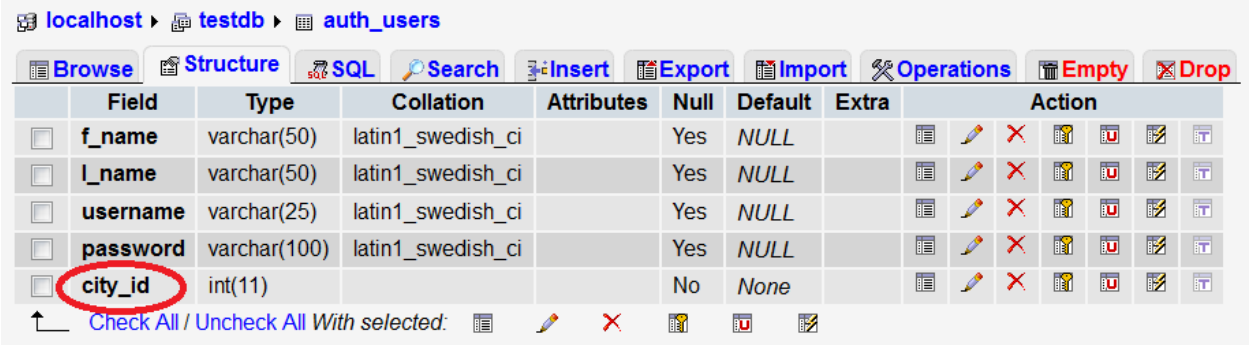
Now you should have a tab Designer when you have selected a database.



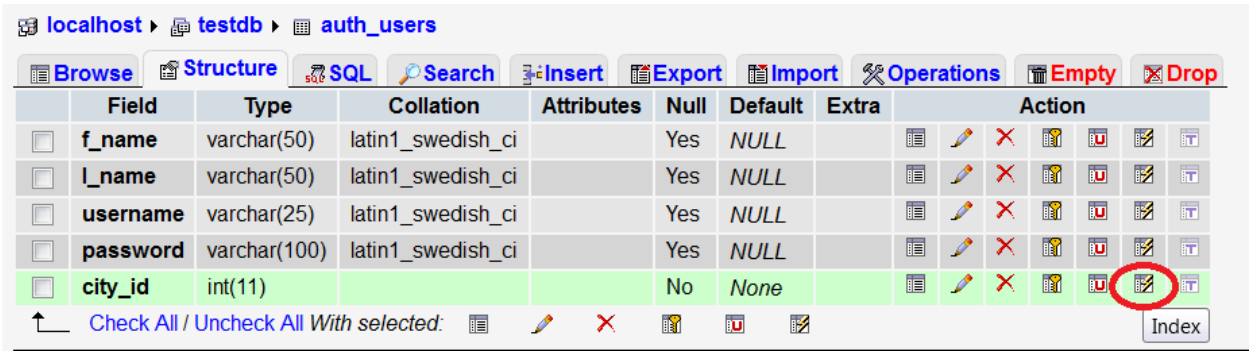

**Creating a One-to-Many Relationship**

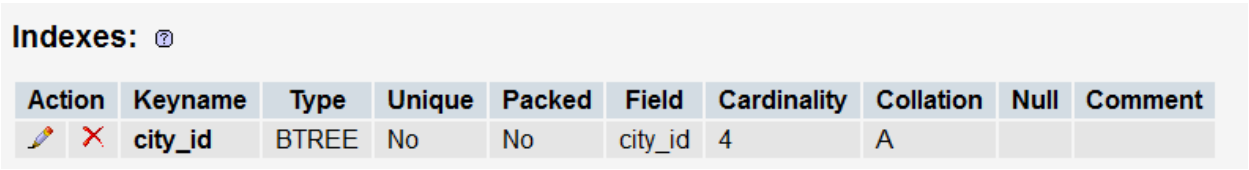
I have created two tables Auth_users and Cities. I am going to create a **One-to Many** Relationship on these tables. One city entry can show up many times in the auth_users table. For example you may have many users from Portland but only need one entry of Portland in the City table. The relationship 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.



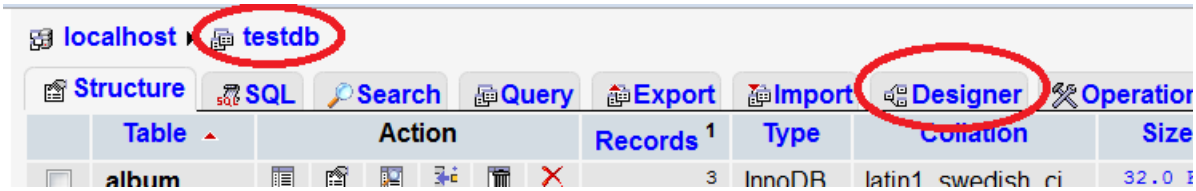



Select the Index on the auth_users.city_id field and the index created will show below. The values entered in the auth_users must be the same values in the city table and the data types must match.

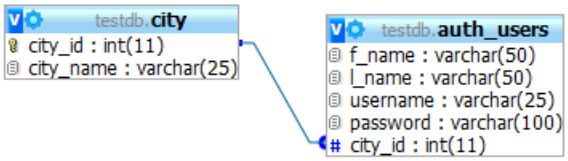



Now go back to the database level on the top navigation and then select Designer.




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 |
|---|---|---|---|---|
| wendy | plourde | wplourde | *94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29 | 1 |
| amy | plourde | wplourde | *94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29 | 1 |
| Sarah | plourde | splourde | *94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29 | 2 |
| Marc | plourde | test | *94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29 | 2 |
| Bob | Smith | bsmith | test | 0 |
| Betty | Smith | bsmith | test | 0 |

| city_id | city_name |
|---|---|
| 1 | Portland |
| 2 | South Portland |
| 3 | Scarbrough |
| 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 |
| | Scarbrough |
| | 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

```php
<?php
  $con = mysql_connect("localhost","root","");
  if (!$con) {
  die('Could not connect: ' . mysql_error());
  }

  mysql_select_db("testdb", $con);
  $result = mysql_query("SELECT auth_users.f_name, city.city_name FROM auth_users INNER JOIN city ON auth_users.city_id =
  city.city_id");
  echo "<table border='1'>
  <tr>
  <th>Name</th>
  <th>City</th>
  </tr>";
  while ($row = mysql_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['f_name'] . "</td>";
  echo "<td>" . $row['city_name'] . "</td>";
  echo "</tr>";
  }
  echo "</table>";

  mysql_close($con);
?>
```