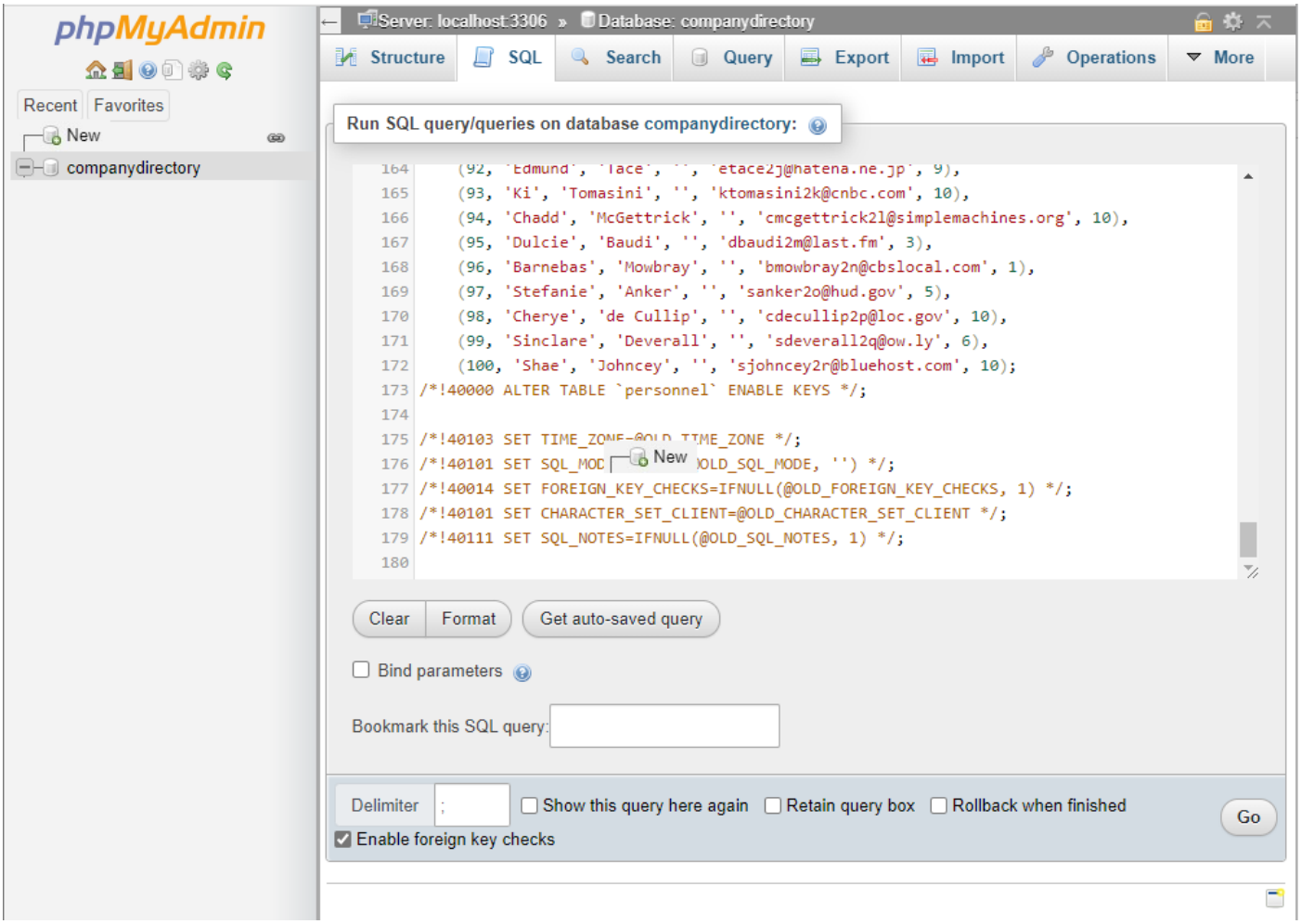
Company Directory

Creating the database on your localhost (XAMPP):

Open phpMyAdmin and create a new database called “companydirectory”. The new database should appear in the list on the left. Click on the database name to select it and then click on the “SQL” tab. Paste the contents of the “company directory.sql” file into it and click on “Go”.

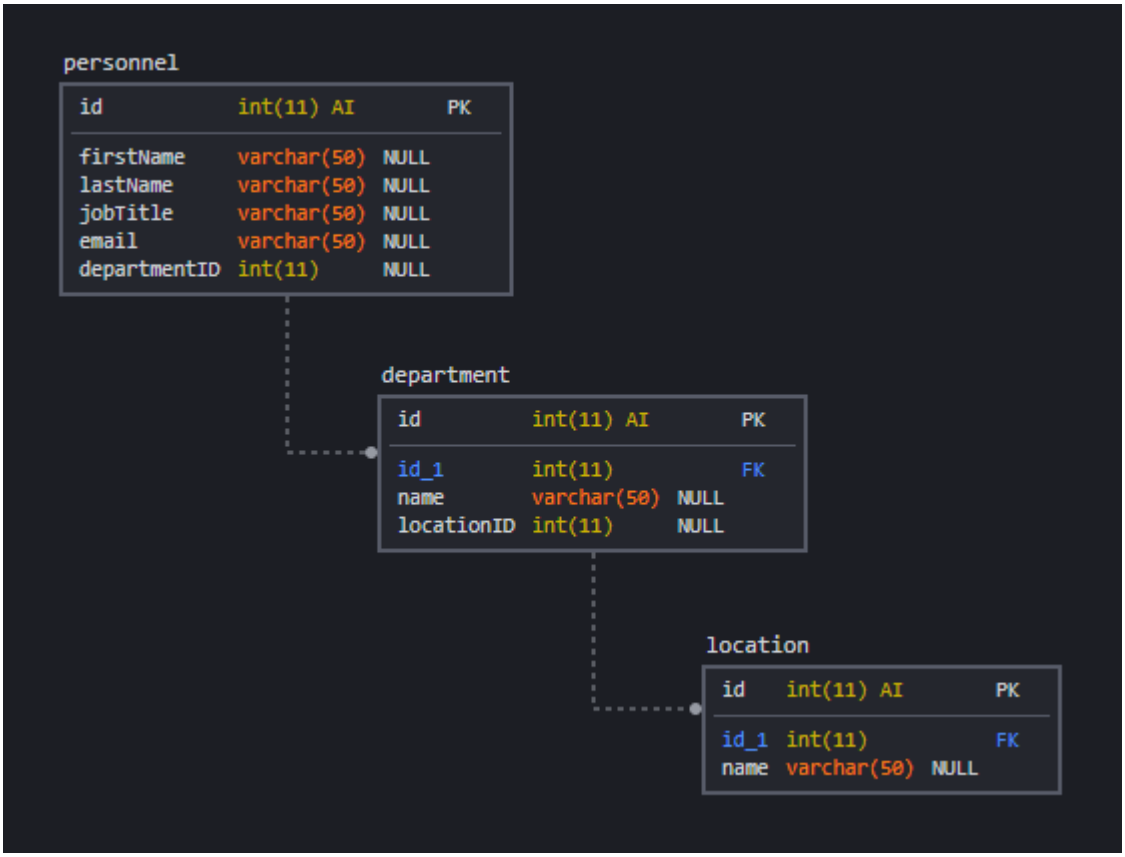


When transferring the tables to your website you usually create the database through the account panel provided by your web hosting company. Once created you can use the same procedure to populate it.

Database Structure:

This is a relational database and so you must follow the foreign keys to see the interdependency between the tables. If you examine the getAll.php file you will see that the relationship is retrieved by using LEFT JOIN:

```
SELECT p.lastName,
       p.firstName,
       p.jobTitle,
       p.email,
       d.name as department,
       l.name as location
FROM   personnel p
       LEFT JOIN department d ON (d.id = p.departmentID)
       LEFT JOIN location l ON (l.id = d.locationID)
ORDER BY p.lastName, p.firstName, d.name, l.name
```



Do not alter the database structure and make sure that your application allows for the addition, update and deletion of entries in all tables.

Before any commit is made the user interface should ask for a confirmation and in cases where there are dependent entries in related tables warn the user that a deletion is not allowed (i.e. where a department has personnel assigned to it or where a location has departments assigned to it).

Connecting your PHP scripts to the database:

All the supplied PHP scripts “include” the following script which holds the connection details for your database. These details must be changed when the script is running on your website.

```php
<?php

    // connection details for MySQL database

    $cd_host = "127.0.0.1";
    $cd_port = 3306;
    $cd_socket = "";

    // database name, username and password

    $cd_dbname = "";
    $cd_user = "";
    $cd_password = "";

?>
```

Running SQL queries:

The examples provided show two similar techniques for building and executing SQL statements. When you build a SQL statement from free form text that has been entered by the user in your system (eg the contents of a form) then we recommend that you use the “prepare” method to guard against SQL injections:

Unprepared statement:

```php
$query = 'SELECT p.lastName, p.firstName FROM personnel';

$result = $conn->query($query);
```

Prepared statement:

```php
$query = $conn->prepare('SELECT p.lastName, p.firstName from personnel WHERE id = ?');

$query->bind_param("i", $_REQUEST['id']);

$result = $query->get_result();
```

Look at this resource in the official PHP documentation for further information on prepared statements.