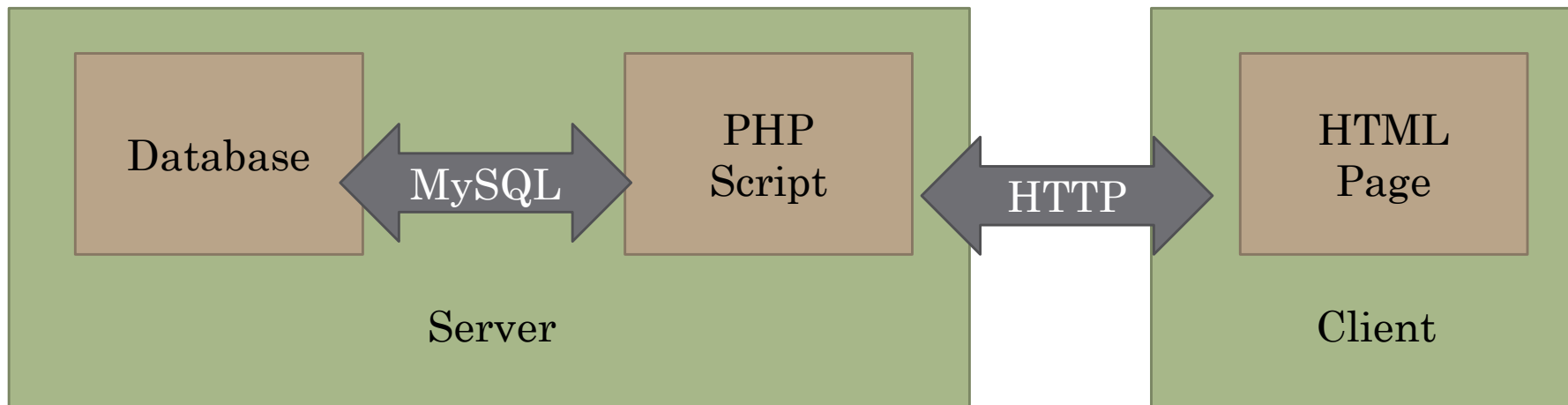


MySQL



How Web Database Architectures Work

1. A user's web browser issues an HTTP request for a particular web page. (search)
2. The web server receives the request, retrieves the file, and passes it to the PHP engine for processing.
3. The PHP engine begins parsing the script. Inside the script, there is a command to connect to the database and execute a query (perform the search). PHP opens a connection to the MySQL server and sends on the appropriate query.
4. The MySQL server receives the database query, processes it, and sends the results back to the PHP engine.
5. The PHP engine finishes running the script. This usually involves formatting the query results nicely in HTML. It then returns the resulting HTML to the web server.
6. The web server passes the HTML back to the browser, where the user can see the result requested.

MySQL

- MySQL is a relational database management system (RDBMS)
- among the most widely used in the world
 - general public : web applications, ...
 - professionals : competing with Oracle, Informix, Microsoft SQL Server
- in 2008 : SUN Microsystems buys MySQL AB
 - Paid \$ 1 billion
- in 2009 : Oracle Corp. purchases SUN
 - 2 competing products: Oracle Database vs. MySQL
- since May 2009, its creator Michael Widenius, My of MySQL, created MariaDB (being the first name of his daughter) to continue its development as an open source project.

MySQL Features

- MySQL is an SQL relational database server
 - worry: performance in reading
 - "select" >>> update
 - multi-threaded and multi-user
- MySQL is free software
 - technical and legal permission: use, study, modification, duplication for dissemination
 - guarantee certain induced freedoms, including the control of the program by the user and the possibility of sharing between individuals
 - 2 types: public law or free license (copyright)
- MySQL is developed under dual license
 - depending on the use made of it
 - in a free product → GNU General Public License (GPL)
 - in a proprietary product → paid license

Using MySQL

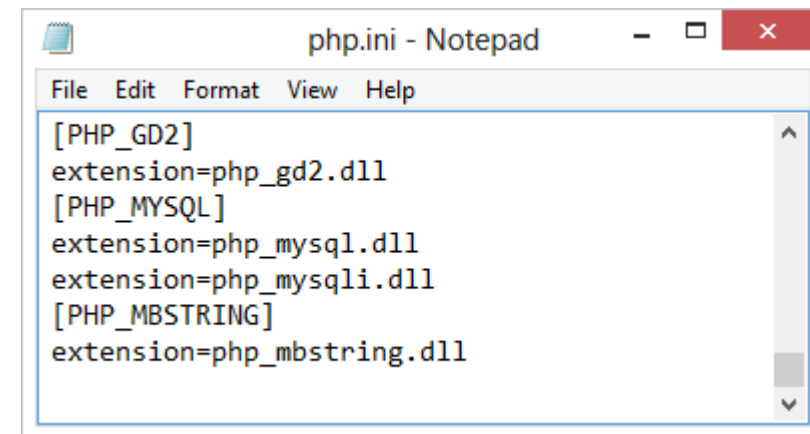
- MySQL + Java
 - Since the acquisition of MySQL AB by Sun Microsystems
 - MySQL is unofficially the database with Java
- PHP + MySQL
 - MySQL is part of the LAMP quartet: Linux + Apache + MySQL + PHP
 - also to its variants WAMP (Windows) and MAMP (Mac)
 - couple very used by websites and proposed by the majority of web hosts

Database engines included in MySQL

- One of the specificities of MySQL is to be able to manage multiple engines in a single database.
- Each table can use a different engine within a database. This is to optimize the use of each table.
- the 2 most important:
 - **MyISAM:**
 - MySQL default engine,
 - simple, uses several files that grow as the database grows
 - does not support transactions or foreign keys;
 - **InnoDB:**
 - engine created and maintained by InnoDB (bought by Oracle in 2005)
 - manages transactions and foreign keys (and therefore the integrity of its tables)
 - in exchange, the bases that use it occupy much more space on the disk;

What are the main PHP API offerings for using MySQL?

- There are three main API options when considering connecting to a MySQL database server:
 - PHP's MySQL Extension
 - PHP's mysqli Extension
 - PHP Data Objects (PDO)
- Each has its own advantages and disadvantages.



```
php.ini - Notepad
File Edit Format View Help
[PHP_GD2]
extension=php_gd2.dll
[PHP_MYSQL]
extension=php_mysql.dll
extension=php_mysqli.dll
[PHP_MBSTRING]
extension=php_mbstring.dll
```

What is PHP's MySQL Extension?

- This is the original extension designed to allow you to develop PHP applications that interact with a MySQL database.
- The *mysql* extension provides a procedural interface and is intended for use only with MySQL versions older than 4.1.3.
- This extension can be used with versions of MySQL 4.1.3 or newer, but not all of the latest MySQL server features will be available.
- Note:
 - If you are using MySQL versions 4.1.3 or later it is *strongly* recommended that you use the *mysqli* extension instead.

What is PHP's mysqli Extension?

- The *mysqli* extension, or as it is sometimes known, the MySQL *improved* extension, was developed to take advantage of new features found in MySQL systems versions 4.1.3 and newer.
- The *mysqli* extension is included with PHP versions 5 and later.
- The *mysqli* extension has a number of benefits, the key enhancements over the *mysql* extension being:
 - Object-oriented interface
 - Support for Prepared Statements
 - Support for Multiple Statements
 - Support for Transactions
 - Enhanced debugging capabilities
 - Embedded server support
- As well as the object-oriented interface the extension also provides a procedural interface.

What is PDO?

- PHP Data Objects, or PDO, is a database abstraction layer specifically for PHP applications.
- PDO provides a consistent API for your PHP application regardless of the type of database server your application will connect to.
- In theory, if you are using the PDO API, you could switch the database server you used, from say Firebird to MySQL, and only need to make minor changes to your PHP code.

What is PDO?

- While PDO has its advantages, such as a clean, simple, portable API, its main disadvantage is that it **doesn't allow you to use all of the advanced features** that are available in the latest versions of MySQL server.
- For example, PDO does not allow you to use MySQL's support for Multiple Statements.
- Other examples of database abstraction layers include JDBC for Java applications and DBI for Perl.

Example #1

Easy migration from the old mysql extension

```
<?php
    $mysqli = mysqli_connect("localhost", "user", "password", "database");
    $res = mysqli_query($mysqli, "SELECT 'Please, do not use ' AS _msg FROM DUAL");
    $row = mysqli_fetch_assoc($res);
    echo $row['_msg'];

    $mysql = mysql_connect("localhost", "user", "password");
    mysql_select_db("database");
    $res = mysql_query("SELECT 'the mysql extension for new developments.'
                        AS _msg FROM DUAL", $mysql);

    $row = mysql_fetch_assoc($res);
    echo $row['_msg'];
?>
```

- The above example will output:
Please, do not use the mysql extension for new developments.

The object-oriented interface

- In addition to the classical procedural interface, users can choose to use the object-oriented interface.
- The documentation is organized using the object-oriented interface.
- The object-oriented interface shows functions grouped by their purpose, making it easier to get started.
- The reference section gives examples for both syntax variants.
- There are no significant performance differences between the two interfaces.
- Users can base their choice on personal preference.

Example #2

Object-oriented and procedural interface

```
<?php
$mysqli = mysqli_connect("localhost", "user", "password", "database");
if (mysqli_connect_errno($mysqli)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

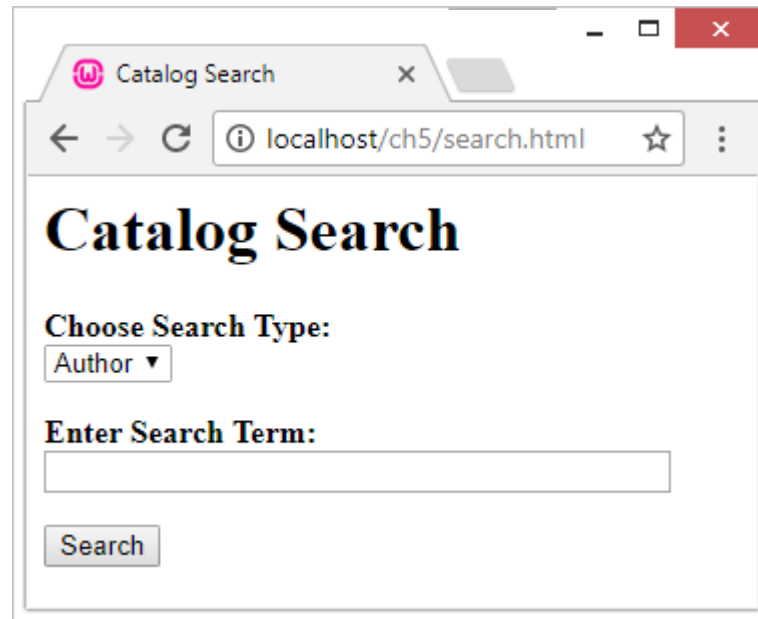
$res = mysqli_query($mysqli, "SELECT 'A world full of ' AS _msg FROM DUAL");
$row = mysqli_fetch_assoc($res);
echo $row['_msg'];

$mysqli = new mysqli("localhost", "user", "password", "database");
if ($mysqli->connect_errno) {
    echo "Failed to connect to MySQL: " . $mysqli->connect_error;
}

?>
```

search.html

Database Search Page



The screenshot shows a web browser window with a single tab titled 'Catalog Search'. The address bar displays 'localhost/ch5/search.html'. The page content includes a main heading 'Catalog Search', a label 'Choose Search Type:' followed by a dropdown menu showing 'Author', a label 'Enter Search Term:' followed by an empty text input field, and a 'Search' button at the bottom.

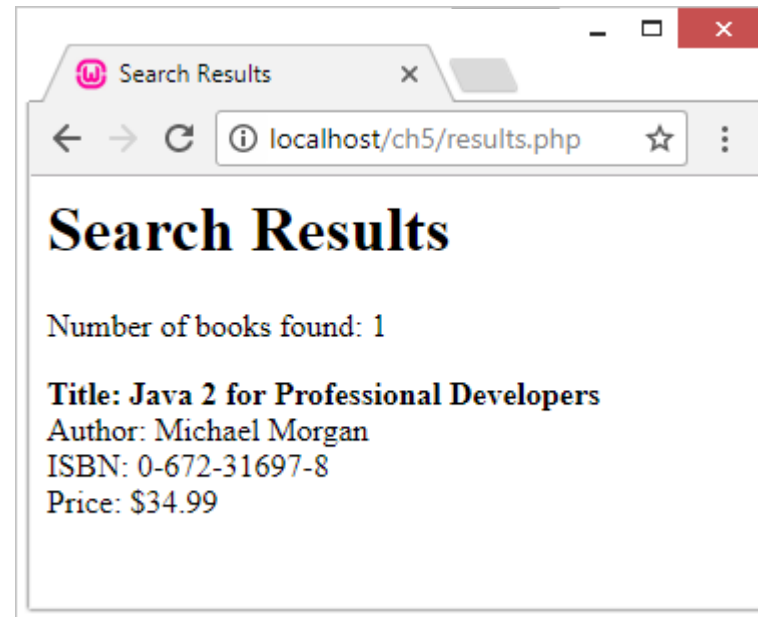
search.html

Database Search Page

```
<head><title>Catalog Search</title></head>
<body>
  <h1>Catalog Search</h1>
  <form action="results.php" method="post">
    <p><strong>Choose Search Type:</strong><br>
    <select name="searchtype">
      <option value="Author">Author</option>
      <option value="Title">Title</option>
      <option value="ISBN">ISBN</option>
    </select></p>
    <p><strong>Enter Search Term:</strong><br>
    <input name="searchterm" type="text" size="40"></p>
    <p><input type="submit" name="submit" value="Search"></p>
  </form>
</body></html>
```


results.php

Database Search Page



results.php

Database Search Page

```
<html>
<head><title>Search Results</title></head>
<body>
<h1>Search Results</h1>
<?php

// create short variable names
$searchtype=$_POST['searchtype'];
$searchterm=trim($_POST['searchterm']);
if (!$searchtype || !$searchterm) {
    echo '<p>You have not entered search details.<br/>
    Please go back and try again.</p>';
    exit;
}
// whitelist the searchtype
switch ($searchtype)
{
    case 'Title':
    case 'Author':
    case 'ISBN':
        break;
    default:
        echo '<p>That is not a valid search type. <br/>
        Please go back and try again.</p>';
        exit;
}
```

results.php

Database Search Page

```
$db = new mysqli('127.0.0.1', 'user', 'pass', 'books');
```

```
if (mysqli_connect_errno()) {  
    echo '<p>Error: Could not connect to database.<br/>  
    Please try again later.</p>';  
    exit;}  

```

```
$query = "SELECT ISBN, Author, Title, Price FROM Books WHERE $searchtype = ?";  
$stmt = $db->prepare($query);  
$stmt->bind_param('s', $searchterm);  
$stmt->execute();  
$stmt->store_result();  
$stmt->bind_result($isbn, $author, $title, $price);  
echo "<p>Number of books found: ".$stmt->num_rows."</p>";  
while($stmt->fetch())  
{  
    echo "<p><strong>Title: ".$title."</strong><br />Author: ".$author;  
    echo "<br />ISBN: ".$isbn."<br />Price: \$.number_format($price,2)."</p>";  
}  
$stmt->free_result();  
$db->close();  
?>  
</body></html>
```

Querying a Database from the Web

- In any script used to access a database from the Web, you follow some basic steps:
 - 1. Check and filter data coming from the user.
 - 2. Set up a connection to the appropriate database.
 - 3. Query the database.
 - 4. Retrieve the results.
 - 5. Present the results back to the user.

Checking and Filtering Input Data

- strip any whitespace that the user might have inadvertently entered at the beginning or end of his search term.
 - function `trim()`
- next step → verify that the user has entered or selected something
 - after trimming whitespace

Validate Data

- When you plan to use any data input by a user, you need to filter it appropriately for any control characters.
- You need to validate data when submitting any user input to a database such as MySQL.
- use
 - whitelist
 - a prepared statement

Setting Up a Connection

- object-oriented approach:
 - `@$db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');`
 - instantiates the `mysqli` class and creates a connection to host `localhost`
 - invoke methods on this object to access the database
- procedural approach
 - `@$db = mysqli_connect('localhost', 'bookorama', 'bookorama123', 'books');`
 - returns a resource rather than an object
 - will need to pass this resource in to all the other `mysqli` functions
 - similar to file-handling functions

- The result of your attempt at connection is worth checking because none of the rest of code will work without a valid database connection.

```
if (mysqli_connect_errno()) {  
    echo '<p>Error: Could not connect to database.<br/>  
    Please try again later.</p>';  
    exit;  
}
```

- (This code is the same for the object-oriented and procedural versions.)
- The `mysqli_connect_errno()` function returns an error number on error, or zero on success.

- error suppression operator, @→ handle any errors gracefully
- could also be done with exceptions
- there is a limit to the number of connections that can exist at the same time
 - max_connections→ for MySQL, my.conf
 - MaxClients→ in Apache, httpd.conf

Choosing a Database to Use

- `$db->select_db(dbname)`
- or as
- `mysqli_select_db(db_resource, db_name)`

Querying the Database

- `mysqli_query()` function
- before → set up the query you want to run:
 - `$query = "SELECT ISBN, Author, Title, Price FROM Books WHERE $searchtype = ?";`
 - placed the `$searchtype` variable directly in the query
 - Where you might expect to see the `$searchterm` variable, you'll instead see a question mark character (?) placeholder.
 - prepared statement

- **placeholders can only be used for data, and not for column, table, or database names**
- To be safe here, we used a whitelisting approach to specify valid values for the \$searchtype variable
- Remember that the query you send to MySQL does not need a semicolon at the end of it, unlike a query you type into the MySQL monitor

Using Prepared Statements

- useful for speeding up execution when you are performing large numbers of the same query with different data
- also help protect against SQL injection-style attacks
- The basic concept
 - you send a template of the query you want to execute to MySQL and then send the data separately
- We use a prepared statement in the results.php script, as follows:
 - `$query = "SELECT ISBN, Author, Title, Price FROM Books WHERE $searchtype = ?";`
 - `$stmt = $db->prepare($query);`
 - `$stmt->bind_param('s', $searchterm);`
 - `$stmt->execute();`

- `$db->prepare()`
 - `mysqli_stmt_prepare()` in the procedural version
 - constructs a statement object or resource that you will then use to do the actual processing.
- The statement object has a method called `bind_param()`
 - In the procedural version, it is called `mysqli_stmt_bind_param()`
 - tell PHP which variables should be substituted for the question marks
- Params
 - 's' → means that the parameter is a string
 - i for integer
 - After this parameter, you should list the same number of variables as you have question marks in your statement. They will be substituted in this order.

Retrieving the Query Results

- As well as binding parameters, you can bind results.
- For SELECT-type queries, you can use
- `$stmt->bind_result()` (or `mysqli_stmt_bind_result()`) to provide a list of variables that you would like the result columns to be filled into.
- Each time you call `$stmt->fetch()` (or `mysqli_stmt_fetch()`), column values from the next row in the result set are filled into these bound variables.
- For example, in the book search script you looked at earlier, you could use
 - `$stmt->bind_result($isbn, $author, $title, $price);`
 - to bind these four variables to the four columns that will be returned from the query.
- After calling
 - `$stmt->execute();`
- you can call
 - `$stmt->fetch();`
- in the loop.

Num rows

- We'd like to get a count of the number of rows returned.
- To do this, we first tell PHP to retrieve and buffer all of the rows returned from the query:

```
$stmt->store_result();  
echo "<p>Number of books found: ".$stmt->num_rows."</p>";
```
- When you use a procedural approach,

```
$num_results = mysqli_num_rows($result);
```


- Next, we retrieve each row from the result set, in a loop, and display it as follows:

```
while($stmt->fetch()) {  
echo "<p><strong>Title: ".$title."</strong>";  
echo "<br />Author: ".$author;  
echo "<br />ISBN: ".$isbn;  
echo "<br />Price: \$.number_format($price,2)."</p>"};
```

- Each call to `$stmt->fetch()`
 - (or, in the procedural version, `mysqli_stmt_fetch()`)
 - retrieves the next row from the result set and populates the four bind variables with the values from that row, and we can then display them.
- There are other approaches to fetching data from a query result other than using `mysqli_stmt_fetch()`. To use these, first we must extract a result set resource from the statement. You can do this using the `mysqli_stmt_get_result()` function, as follows:
 - `$result = $stmt->get_result();`

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

- `mysqli_fetch_array()`
 - (and related `mysqli_fetch_assoc()`),
 - returns the next row from the result set as an array.
- The `mysqli_fetch_assoc()` version uses the column names as keys,
 - although you can also get this behavior from `mysqli_fetch_array()`.
 - The `mysqli_fetch_array()` function takes a second parameter for the type of array to return.
 - `MYSQLI_ASSOC` will get you the column names as keys,
 - `MYSQLI_NUM` will result in numbered keys,
 - `MYSQLI_BOTH` will give you an array containing two sets of the data, one with column names as keys and one with numerical keys.
- `mysqli_fetch_all()`
 - returns all of the rows returned by the query as an array of arrays where each of the inner arrays is one of the rows returned.
- `mysqli_fetch_object()`
 - returns the next row from the result set as an object,
 - where each value is stored in an attribute carrying the name of the column.

Disconnecting from the Database

- You can free your result set by calling either
 - `$result->free();`
 - or
 - `mysqli_free_result($result);`
- You can then use
 - `$db->close();`
 - or
 - `mysqli_close($db);`
- to close a database connection. Using this command isn't strictly necessary because the connection will be closed when a script finishes execution anyway.

Administration with phpMyAdmin web tool

What is PhpMyAdmin?

- **PhpMyAdmin** is one of the most popular applications for MySQL databases management.
- It is a free tool written in PHP.
- Through this software you can create, alter, drop, delete, import and export MySQL database tables.
- You can run MySQL queries, optimize, repair and check tables, change collation and execute other database management commands.

PhpMyAdmin Features

- The main PhpMyAdmin features are as follows:
 - User-friendly web interface;
 - Support for most MySQL functions like browse, drop, create, copy and alter databases, tables, views, fields and indexes, execute MySQL queries, manage stored procedures and functions;
 - Import data from CSV and SQL files;
 - Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, Excel, LATEX and others;
 - Searching globally in a database or a subset of it;
 - And much more.

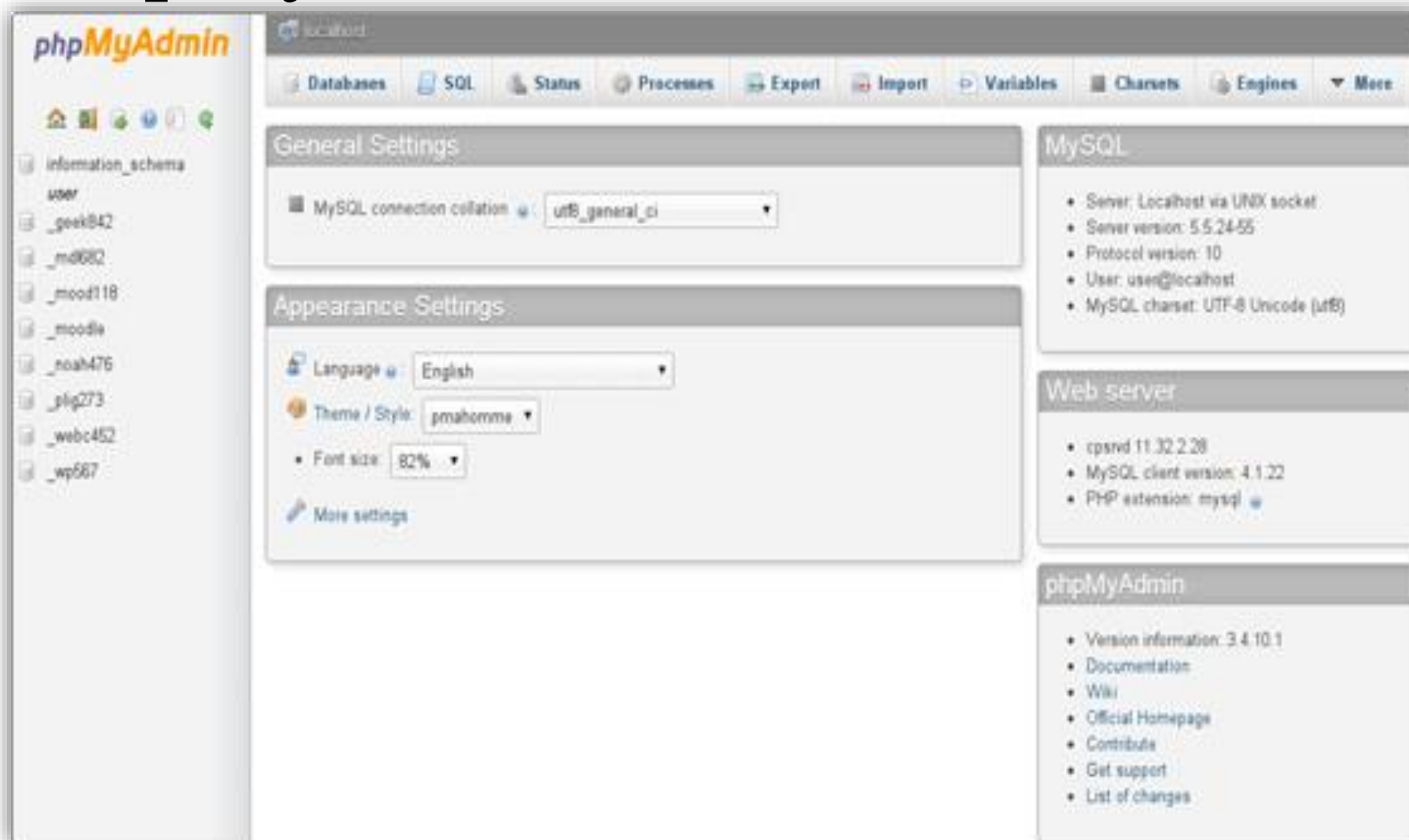
How to install PhpMyAdmin ?

- by Xampp Application the PhpMyAdmin installation .
- Or you can install PhpMyAdmin from website and add it into your server as Apache server .

PhpMyAdmin administration

- Once you enter your PhpMyAdmin application, you will see different areas.

PhpMyAdmin administration



PhpMyAdmin administration

- In the upper part you will find the server hostname. The databases which you will manage are stored on the same server as the software and the hostname is: **localhost**.
- Under it there is information regarding the MySQL server, the MySQL client and the PhpMyAdmin version.
- Next, you will see the MySQL charset and you will be able to define the MySQL connection collation.
- In the right column you can change the default language, alter the style, customize the theme color and the font size. Also there you will notice links to PhpMyAdmin resources.

The screenshot displays the phpMyAdmin web interface. At the top, a navigation bar includes tabs for 'Databases', 'SQL', 'Status', 'User accounts', 'Export', 'Import', 'Settings', 'Replication', 'Variables', 'Charsets', and 'More'. The 'Server: localhost' tab is selected and highlighted with a red box. Below the navigation bar, the 'General settings' section contains a 'Change password' link and a 'Server connection collation' dropdown menu set to 'utf8mb4_unicode_ci', which is also highlighted with a red box. The 'Appearance settings' section includes a 'Language' dropdown menu set to 'English' (highlighted with a red box), a 'Theme' dropdown menu set to 'pmahomme', and a 'Font size' dropdown menu set to '82%'. A 'More settings' link is located at the bottom of this section. On the right side, the 'Database server' and 'Web server' sections are highlighted with red boxes. The 'Database server' section lists the following details: Server: localhost via TCP/IP, Server type: MySQL, Server version: 5.7.19-log - MySQL Community Server (GPL), Protocol version: 10, User: root@localhost, and Server charset: UTF-8 Unicode (utf8). The 'Web server' section lists: Apache/2.4.27 (Win32) OpenSSL/1.0.2l PHP/7.1.8, Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e \$, PHP extension: mysqli and mbstring, and PHP version: 7.1.8.

Server: localhost

Databases SQL Status User accounts Export Import Settings Replication Variables Charsets More

General settings

[Change password](#)

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English

Theme: pmahomme

Font size: 82%

[More settings](#)

Database server

- Server: localhost via TCP/IP
- Server type: MySQL
- Server version: 5.7.19-log - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

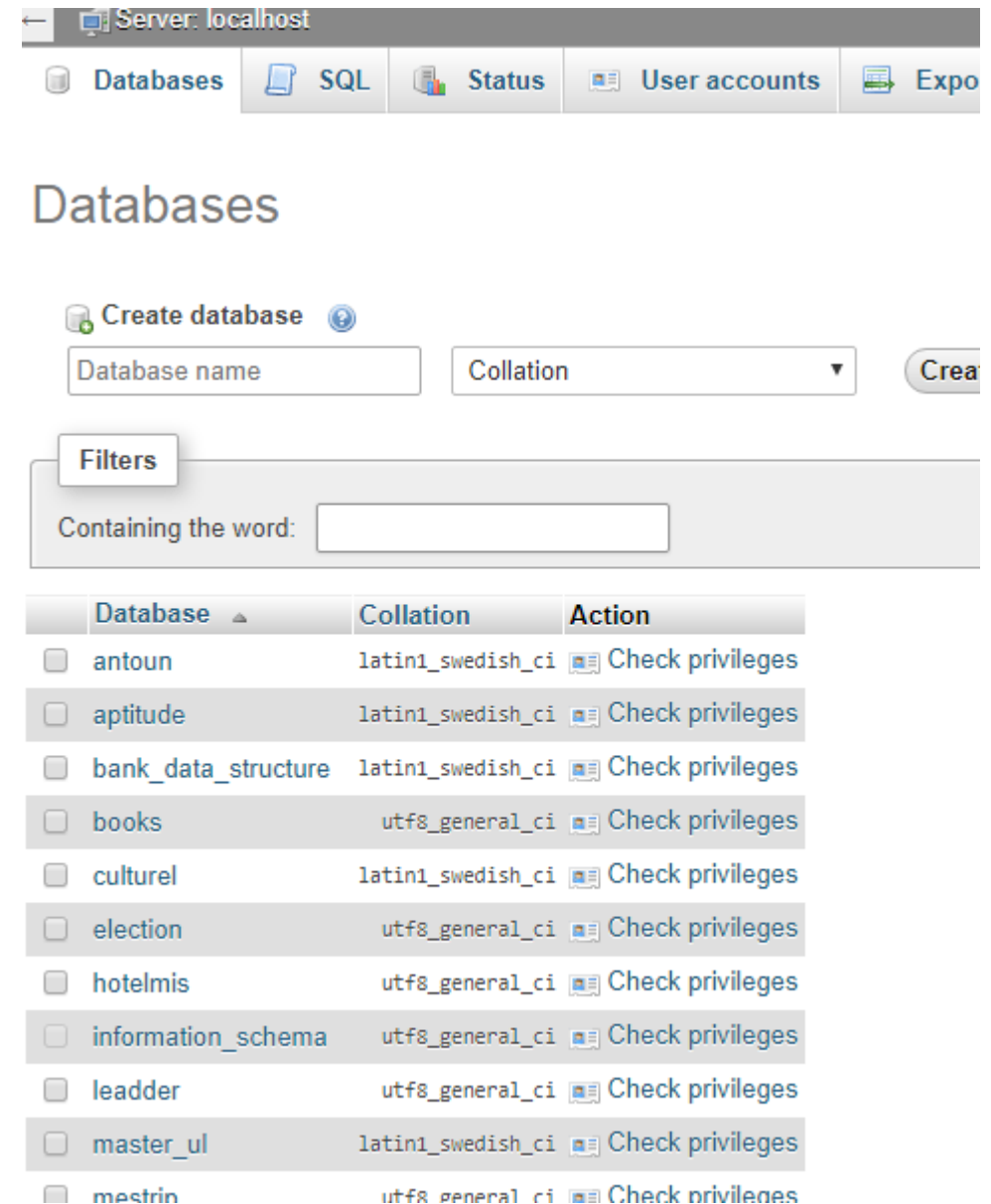
- Apache/2.4.27 (Win32) OpenSSL/1.0.2l PHP/7.1.8
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e \$
- PHP extension: mysqli mbstring
- PHP version: 7.1.8

PhpMyAdmin administration

- The main PhpMyAdmin areas are as follows:
 - 1) Databases
 - 2) Status
 - 3) Variables
 - 4) Processes
 - 5) Charsets
 - 6) Engines
 - 7) Export
 - 8) Import

Databases area

- In the **Databases** tab you will find a list with all the databases which can be managed through the localhost server .
- Once you click on a chosen database, you can start its management.

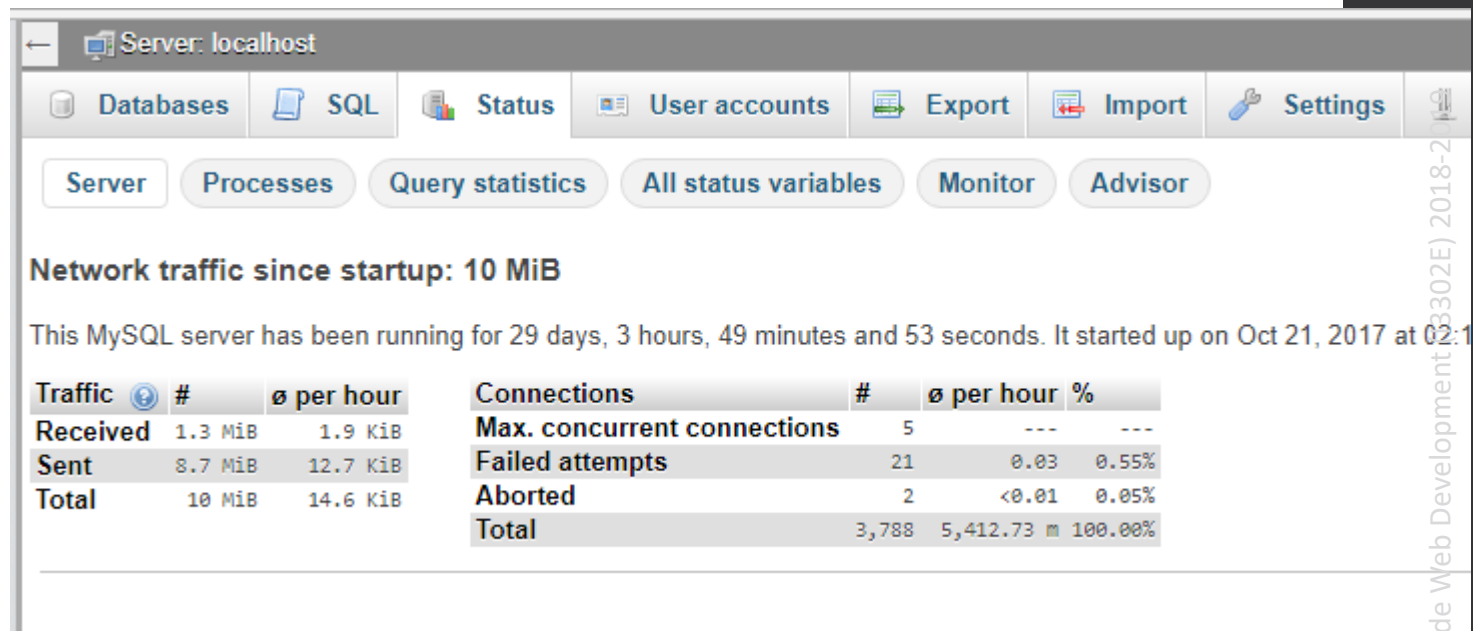


The screenshot shows the 'Databases' tab in a web interface for a MySQL server on localhost. At the top, there are navigation tabs: 'Databases', 'SQL', 'Status', 'User accounts', and 'Export'. Below the tabs, the title 'Databases' is displayed. A 'Create database' section includes a text input for 'Database name', a dropdown for 'Collation', and a 'Create' button. Below this is a 'Filters' section with a text input labeled 'Containing the word:'. The main part of the interface is a table listing existing databases. Each row includes a checkbox, the database name, its collation, and a 'Check privileges' link.

	Database	Collation	Action
<input type="checkbox"/>	antoun	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	aptitude	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	bank_data_structure	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	books	utf8_general_ci	Check privileges
<input type="checkbox"/>	culturel	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	election	utf8_general_ci	Check privileges
<input type="checkbox"/>	hotelmis	utf8_general_ci	Check privileges
<input type="checkbox"/>	information_schema	utf8_general_ci	Check privileges
<input type="checkbox"/>	leadder	utf8_general_ci	Check privileges
<input type="checkbox"/>	master_ul	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	mestrin	utf8_general_ci	Check privileges

Status area

- There you will find detailed information regarding the MySQL server since the last restart.
- You will see the traffic handled by the MySQL server, the maximum number of simultaneous connections, the total number of connections, the failed and the aborted attempts, the total number of queries sent to the server and more related details.



Server: localhost

Databases SQL Status User accounts Export Import Settings

Server Processes Query statistics All status variables Monitor Advisor

Network traffic since startup: 10 MiB

This MySQL server has been running for 29 days, 3 hours, 49 minutes and 53 seconds. It started up on Oct 21, 2017 at 02:1

Traffic	#	Ø per hour
Received	1.3 MiB	1.9 KiB
Sent	8.7 MiB	12.7 KiB
Total	10 MiB	14.6 KiB

Connections	#	Ø per hour	%
Max. concurrent connections	5	---	---
Failed attempts	21	0.03	0.55%
Aborted	2	<0.01	0.05%
Total	3,788	5,412.73	100.00%

Variables area

- You will see a list with the MySQL server system variables and their values.

The screenshot shows the MySQL Server Administration web interface for 'localhost'. The 'Variables' tab is selected in the top navigation bar. The main heading is 'Server variables and settings'. Below this is a 'Filters' section with a search box labeled 'Containing the word:'. The main content area displays a table of system variables.

Action	Variable	Session value / Global value
Edit	auto increment increment?	1
Edit	auto increment offset?	1
Edit	autocommit?	ON
Edit	automatic sp privileges?	ON

Processes area

- By clicking on the **Processes** link you will see all the processes running by your localhost server.

The screenshot shows the MySQL Server Processes area in phpMyAdmin. The top navigation bar includes links for Databases, SQL, Status, User accounts, Export, Import, Settings, and Replication. Below this, a secondary navigation bar highlights the 'Processes' link, along with 'Server', 'Query statistics', 'All status variables', 'Monitor', and 'Advisor'. A 'Filters' section contains a checkbox for 'Show only active' and a 'Refresh' button. The main area displays a table of processes with columns: Processes, ID, User, Host, Database, Command, Time, Status, Progress, and SQL query. Two processes are listed: one with ID 3804, User root, Host localhost:20380, Database mysql, Command Query, Status 0 starting, and Progress ---; and another with ID 3805, User root, Host localhost:20381, Database None, Command Sleep, Status 0, Progress ---, and SQL query ---. A note at the bottom states: 'Note: Enabling the auto refresh here might cause heavy traffic between the web server and the MySQL server.' Below the note, there is a 'Refresh rate' dropdown set to '5 seconds' and a 'Start auto refresh' button.

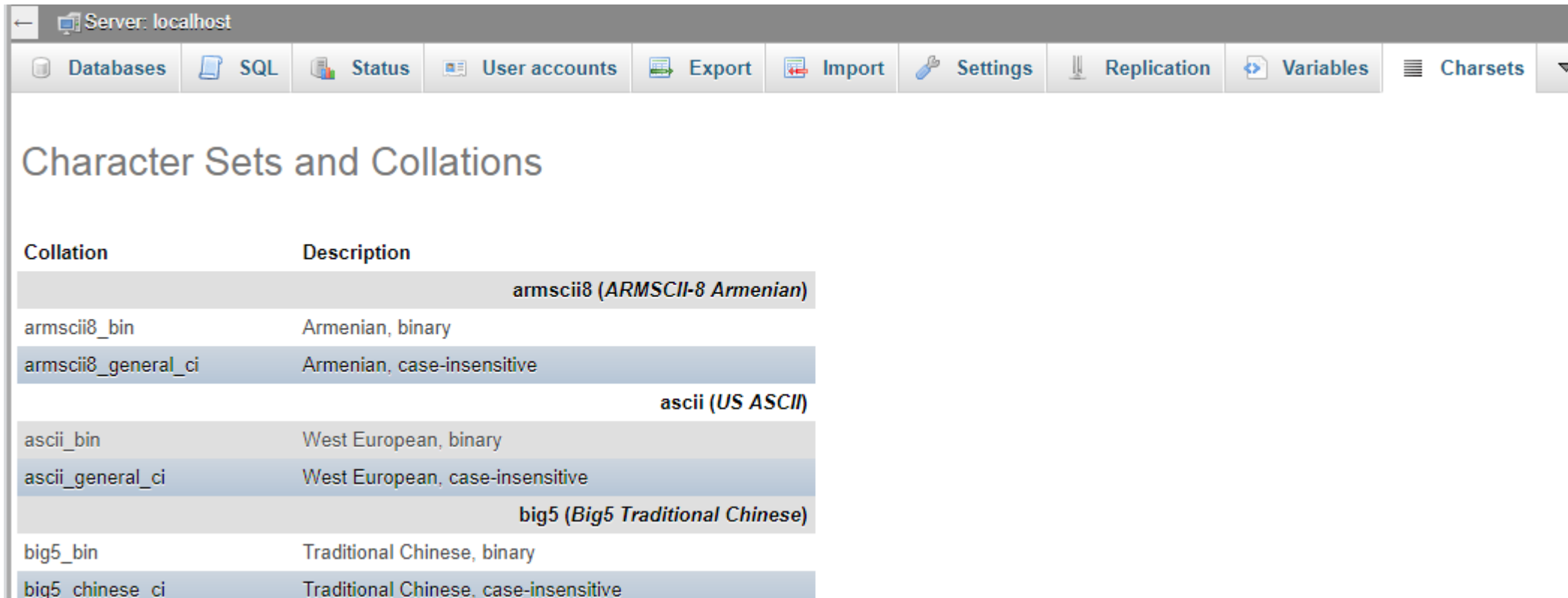
Processes	ID	User	Host	Database	Command	Time	Status	Progress	SQL query
Kill	3804	root	localhost:20380	mysql	Query	0	starting	---	SHOW PROCESSLIST
Kill	3805	root	localhost:20381	None	Sleep	0	---	---	---

Note: Enabling the auto refresh here might cause heavy traffic between the web server and the MySQL server.

Refresh rate: 5 seconds Start auto refresh

Charsets area

- The **Character Sets and Collations** link leads to the **Charset** area. There you will find all the charsets and collations supported by the MySQL server.

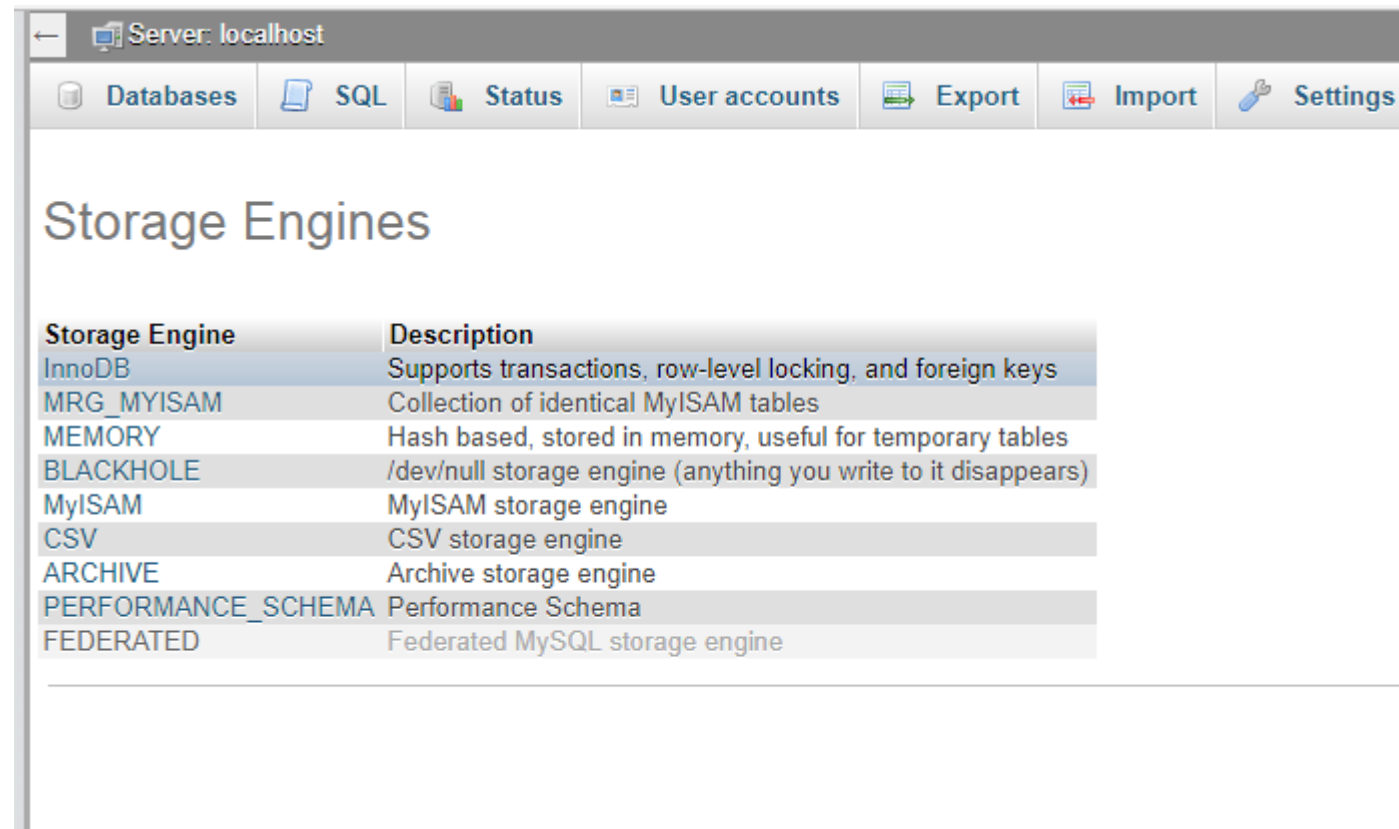


The screenshot shows the MySQL web interface for the 'Server: localhost'. The 'Charsets' tab is selected in the top navigation bar. The main content area is titled 'Character Sets and Collations' and displays a table of collations. The table has two columns: 'Collation' and 'Description'. The collations are grouped by character set: armSCII8, ascii, and big5. Each group has a header row and two data rows (binary and case-insensitive).

Collation	Description
armSCII8 (ARMSCII-8 Armenian)	
armSCII8_bin	Armenian, binary
armSCII8_general_ci	Armenian, case-insensitive
ascii (US ASCII)	
ascii_bin	West European, binary
ascii_general_ci	West European, case-insensitive
big5 (Big5 Traditional Chinese)	
big5_bin	Traditional Chinese, binary
big5_chinese_ci	Traditional Chinese, case-insensitive

Engines area

- The **Storage Engines** link opens a list with all the engines supported by the MySQL server. The default one is MyISAM. Another popular storage engine, used by many databases is InnoDB. More about the MySQL storage engines can be found in the [official documentation](#).

A screenshot of a web browser showing the MySQL Storage Engines page. The browser's address bar shows 'Server: localhost'. The page has a navigation bar with links: Databases, SQL, Status, User accounts, Export, Import, and Settings. The main heading is 'Storage Engines'. Below it is a table with two columns: 'Storage Engine' and 'Description'. The table lists several storage engines: InnoDB, MRG_MYISAM, MEMORY, BLACKHOLE, MyISAM, CSV, ARCHIVE, PERFORMANCE_SCHEMA, and FEDERATED, each with a brief description.

Storage Engine	Description
InnoDB	Supports transactions, row-level locking, and foreign keys
MRG_MYISAM	Collection of identical MyISAM tables
MEMORY	Hash based, stored in memory, useful for temporary tables
BLACKHOLE	/dev/null storage engine (anything you write to it disappears)
MyISAM	MyISAM storage engine
CSV	CSV storage engine
ARCHIVE	Archive storage engine
PERFORMANCE_SCHEMA	Performance Schema
FEDERATED	Federated MySQL storage engine

Export area

- In the **Export** section you can export your database tables content in different formats (CSV, SQL, PDF, Microsoft Excel, Microsoft Word, XML, and many more). You can select all the database tables or just pick some of them.

Server: localhost

Databases SQL Status User accounts Export Import Settings

Exporting databases from the current server

Export method:

☐ Quick - display only the minimal options
☒ Custom - display all possible options

Format:

SQL

Databases:

Select all / Unselect all

- antoun
- aptitude
- bank_data_structure
- books
- culturel
- election
- hotelmis
- leader
- master_ul
- mestrip

Output:

☐ Rename exported databases/tables/columns

☐ Save output to a file

Export area

- You can add custom comments in the header of the exported content.
You can decide whether to export just the database structure, the data or both of them. You can export the database tables in a file and compress it or you can visualize the queries directly on the screen.

Import area

- In the **Import** section you can import your database tables from a file, saved on your local computer.
- You should browse for the file and pick its character set from the drop-down menu.
- If the file is too big, the MySQL server timeout can be reached. In such a case you can interrupt the import action. Then you can continue with the data import defining the number of the queries to be skipped from the file beginning. In this way you will skip the imported queries and continue from the point of the interruption.
- Additionally you can pick the SQL server mode of the imported file. You can find more details in the [Server SQL Modes](#) documentation.

← Server: localhost

DatabasesSQLStatusUser accountsExportImportSettingsReplicationVariables

Importing into the current server

File to import:

File may be compressed (gzip, zip) or uncompressed.
A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Browse your computer: No file chosen (Max: 120MiB)

You may also drag and drop a file on any page.

Character set of the file:

Partial import:

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however)*

Skip this number of queries (for SQL) starting from the first one:

Other options:

☒ Enable foreign key checks

Format:

Manage MySQL database tables with PhpMyAdmin ?

- The main functionality of the PhpMyAdmin tool is to manage your databases.
- Click on the **Databases** link. Pick the preferred database which you want to manage and click on its name.

Databases

Create database ?

Database name

Collation

Create

Filters

Containing the word:

	Database	Collation	Action
<input type="checkbox"/>	antoun	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	aptitude	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	bank_data_structure	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	books	utf8_general_ci	Check privileges
<input type="checkbox"/>	culturel	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	election	utf8_general_ci	Check privileges
<input type="checkbox"/>	hotelmis	utf8_general_ci	Check privileges
<input type="checkbox"/>	information_schema	utf8_general_ci	Check privileges
<input type="checkbox"/>	leadder	utf8_general_ci	Check privileges
<input type="checkbox"/>	master_ul	latin1_swedish_ci	Check privileges
<input type="checkbox"/>	mestrip	utf8_general_ci	Check privileges
<input type="checkbox"/>	msafe_users	utf8_general_ci	Check privileges

Manage MySQL database tables with PhpMyAdmin ?

- In the new screen you will see a list with the database tables, the allowed actions with them, the number of the records, the storage engine, the collation, the tables' sizes and the overhead.

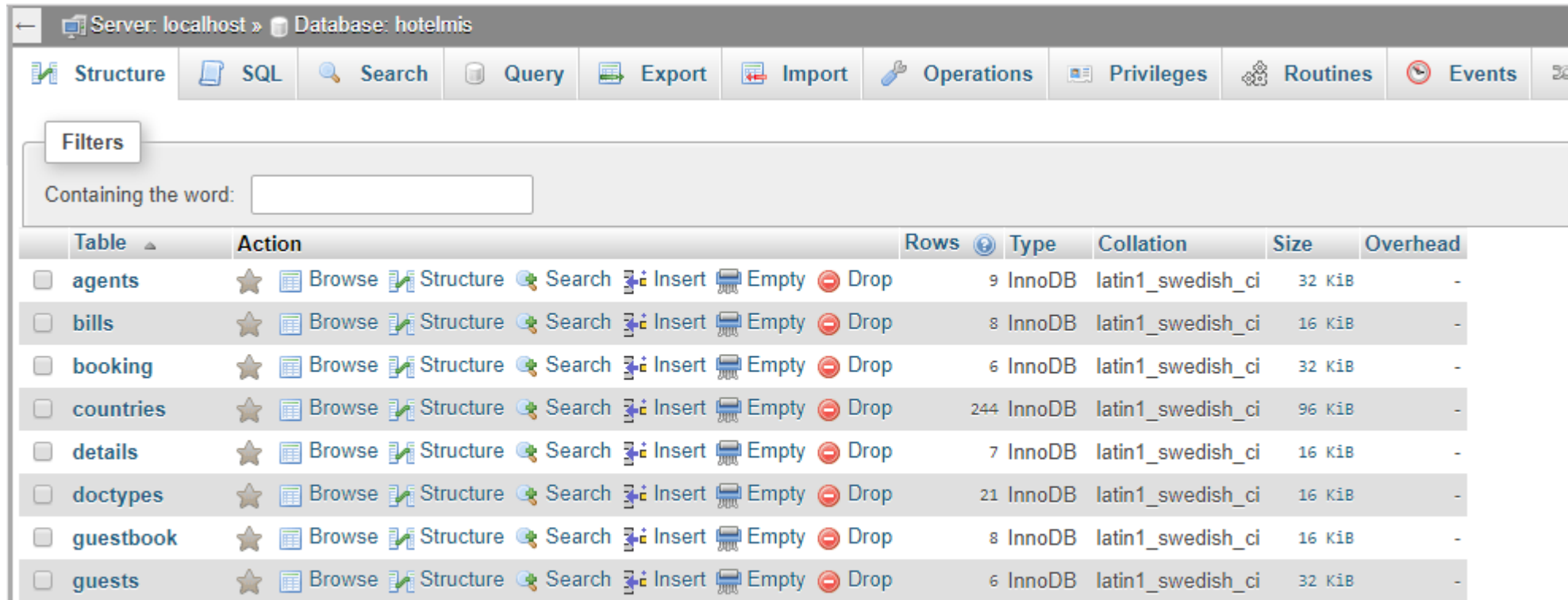


Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> agents	★ Browse Structure Search Insert Empty Drop	9	InnoDB	latin1_swedish_ci	32 KiB	-
<input type="checkbox"/> bills	★ Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	16 KiB	-
<input type="checkbox"/> booking	★ Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	32 KiB	-
<input type="checkbox"/> countries	★ Browse Structure Search Insert Empty Drop	244	InnoDB	latin1_swedish_ci	96 KiB	-
<input type="checkbox"/> details	★ Browse Structure Search Insert Empty Drop	7	InnoDB	latin1_swedish_ci	16 KiB	-
<input type="checkbox"/> doctypes	★ Browse Structure Search Insert Empty Drop	21	InnoDB	latin1_swedish_ci	16 KiB	-
<input type="checkbox"/> guestbook	★ Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	16 KiB	-
<input type="checkbox"/> guests	★ Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	32 KiB	-

Manage MySQL database tables with PhpMyAdmin ?

- The possible actions which you can perform to a chosen table are:
 1. Browse
 2. Structure
 3. Search
 4. Insert
 5. Empty
 6. Drop

Browse Database Tables

- Only the tables with existing records can be browsed. Once you click on the **Browse** icon a new window with the records list will be opened.

Server: localhost » Database: hotelmis » Table: agents "InnoDB free: 120832 kB"

Browse Structure SQL Search Insert Export Import Privileges Operations

✓ Showing rows 0 - 8 (9 total, Query took 0.0616 seconds.)

`SELECT * FROM `agents``

☐ Profiling [Edit inline]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	agentid	agentname	agents_ac_no	contact_person	telephone	fax	email	billing_address
<input type="checkbox"/> Edit Copy Delete	1	Kemri	K001	Hamida	254-(0)41-522063	NULL	NULL	230
<input type="checkbox"/> Edit Copy Delete	2	Plan	K002	Johnson	NULL	NULL	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	5	Kiteco	K003	Charo				NULL
<input type="checkbox"/> Edit Copy Delete	6	Kilifi District Hospital	K006	Saulo				NULL
<input type="checkbox"/> Edit Copy Delete	7	Mnarani	K009	Steve				NULL
<input type="checkbox"/> Edit Copy Delete	8	KDDP	K010	Kombe				NULL
<input type="checkbox"/> Edit Copy Delete	9	TTN	K015	Tony				NULL
<input type="checkbox"/> Edit Copy Delete	10	World Vision	K055	Terry	500263	NULL	NULL	748
<input type="checkbox"/> Edit Copy Delete	11	Wonder Nuts	KN008	Pandre	522063	NULL	NULL	389

☐ Check all | With selected: Edit Copy Delete Export

Browse Database Tables

- By clicking on the **Pen** icon you can edit the chosen record.
- You will see the record structure and you can alter the values of the records.

Structure Database Tables

- The next option is named **Structure**. In the **Structure** screen you will see the database's table structure.

Table structure Relation view										
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action	
<input type="checkbox"/>	1	agentid	int(11)		No	None		AUTO_INCREMENT	Change	Drop Primary Unique Index Spatial More
<input type="checkbox"/>	2	agentname	varchar(65) latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	3	agents_ac_no	char(6) latin1_swedish_ci		No	None			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	4	contact_person	varchar(65) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	5	telephone	varchar(21) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	6	fax	varchar(21) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	7	email	varchar(50) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	8	billing_address	varchar(15) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	9	town	varchar(35) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	10	postal_code	int(10)		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	11	road_street	varchar(65) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	12	building	varchar(65) latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/>	13	ratesid	int(11)		Yes	NULL			Change Drop Primary Unique Index Spatial More	
<input type="checkbox"/> Check all With selected: Browse Change Drop Primary Unique Index										

Structure Database Tables

- You will see the fields' names, their types, collations, attributes, additional extra information, the default values and whether the fields' values can be NULL. You can browse for distinct values by clicking on the corresponding action icon. Also, you can edit a field's structure or delete a field. You can define different indexes: Primary, Unique, Index and Full text. More about the indexes can be found in the MySQL Indexes documentation.
- In the **Indexes** area you will find the indexes assigned for the table and the fields for which they are set. You can edit and delete them.
- Additionally, in the same screen you can check the **Space Usage** and the **Row Statistics**.

Search Database Tables

- Through the **Search** action you can generate a search query for the chosen table.

Server: localhost » Database: hotelmis » Table: agents "InnoDB free: 120832 kB"

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Privileges](#)
[Operations](#)
[Triggers](#)

[Table search](#)
[Zoom search](#)
[Find and replace](#)

Do a "query by example" (wildcard: "%")

Column	Type	Collation	Operator	Value
agentid	int(11)		=	
agentname	varchar(65)	latin1_swedish_ci	LIKE	
agents_ac_no	char(6)	latin1_swedish_ci	LIKE	
contact_person	varchar(65)	latin1_swedish_ci	LIKE	
telephone	varchar(21)	latin1_swedish_ci	LIKE	
fax	varchar(21)	latin1_swedish_ci	LIKE	
email	varchar(50)	latin1_swedish_ci	LIKE	
billing_address	varchar(15)	latin1_swedish_ci	LIKE	
town	varchar(35)	latin1_swedish_ci	LIKE	
postal_code	int(10)		=	
road_street	varchar(65)	latin1_swedish_ci	LIKE	
building	varchar(65)	latin1_swedish_ci	LIKE	
ratesid	int(11)		=	

Server-side Web Development (I3302E) 2018-2019

Search Database Tables

- You can either write the WHERE clause or you can use the "query by example" functionality. You should click on the **Go** button to execute the query.
- For example, if you want to visualize all the records with a field value that starts with **a** you should select the fields which you want to show. Pick the LIKE operator from the drop-down menu and enter in the corresponding field value **a%** (% stands for a wildcard string). Click on the **Go** button to see the result.

Insert into Database Tables

- Using the **Insert** action you can insert records in your database table.
- Once you fill in the corresponding values click on the **Go** button and the new record will be inserted.

Server: localhost » Database: hotelmis » Table: agents "InnoDB free: 120832 kB"

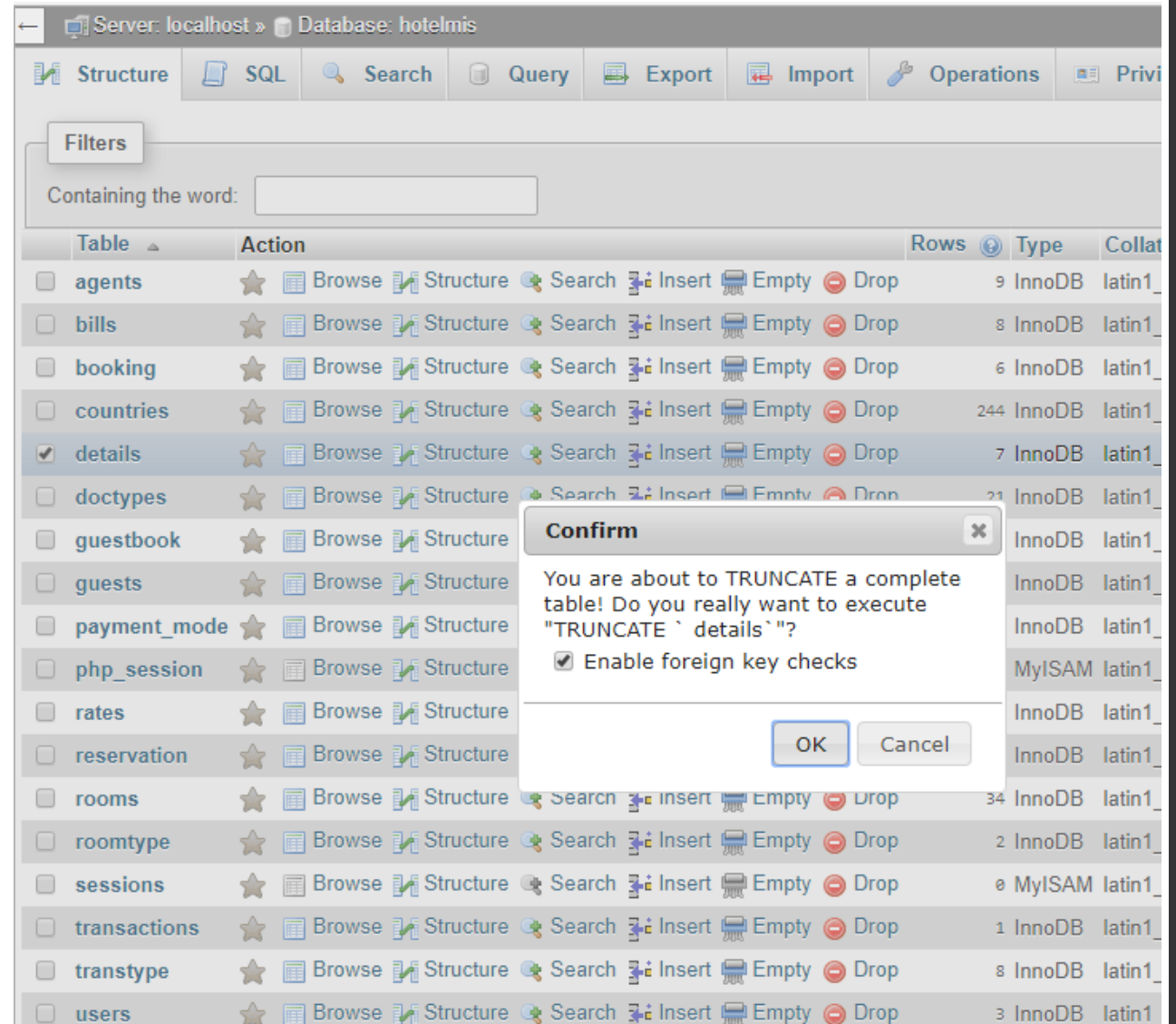
Browse Structure SQL Search Insert Export Import Privileges Operations Trigger

Column	Type	Function	Null	Value
agentid	int(11)			
agentname	varchar(65)			
agents_ac_no	char(6)			
contact_person	varchar(65)		<input checked="" type="checkbox"/>	
telephone	varchar(21)		<input checked="" type="checkbox"/>	
fax	varchar(21)		<input checked="" type="checkbox"/>	
email	varchar(50)		<input checked="" type="checkbox"/>	
billing_address	varchar(15)		<input checked="" type="checkbox"/>	
town	varchar(35)		<input checked="" type="checkbox"/>	
postal_code	int(10)		<input checked="" type="checkbox"/>	
road_street	varchar(65)		<input checked="" type="checkbox"/>	
building	varchar(65)		<input checked="" type="checkbox"/>	
ratesid	int(11)		<input checked="" type="checkbox"/>	

Go

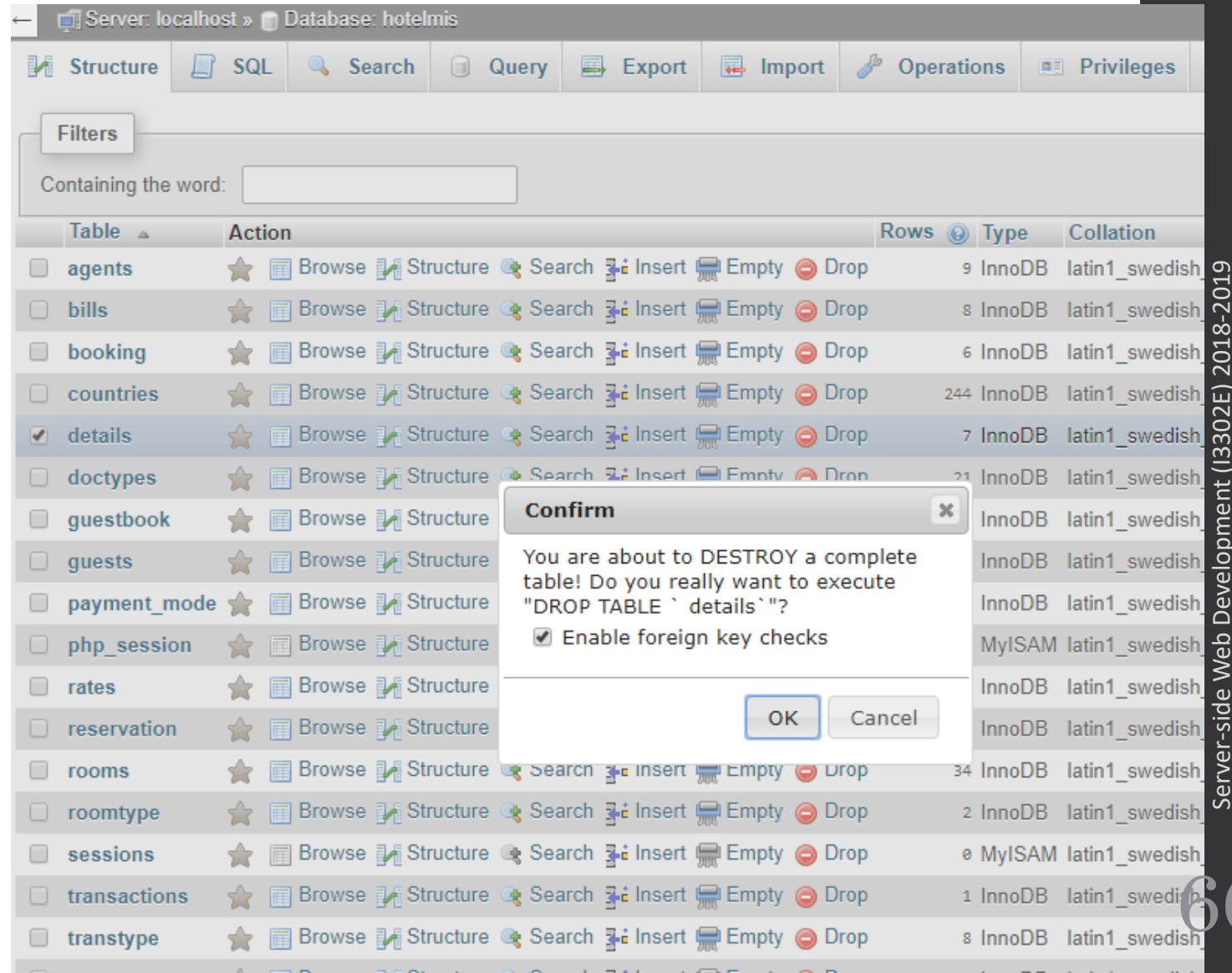
Empty Database Tables

- The **Empty** action allows you to empty your database table, removing the data and keeping the empty table.



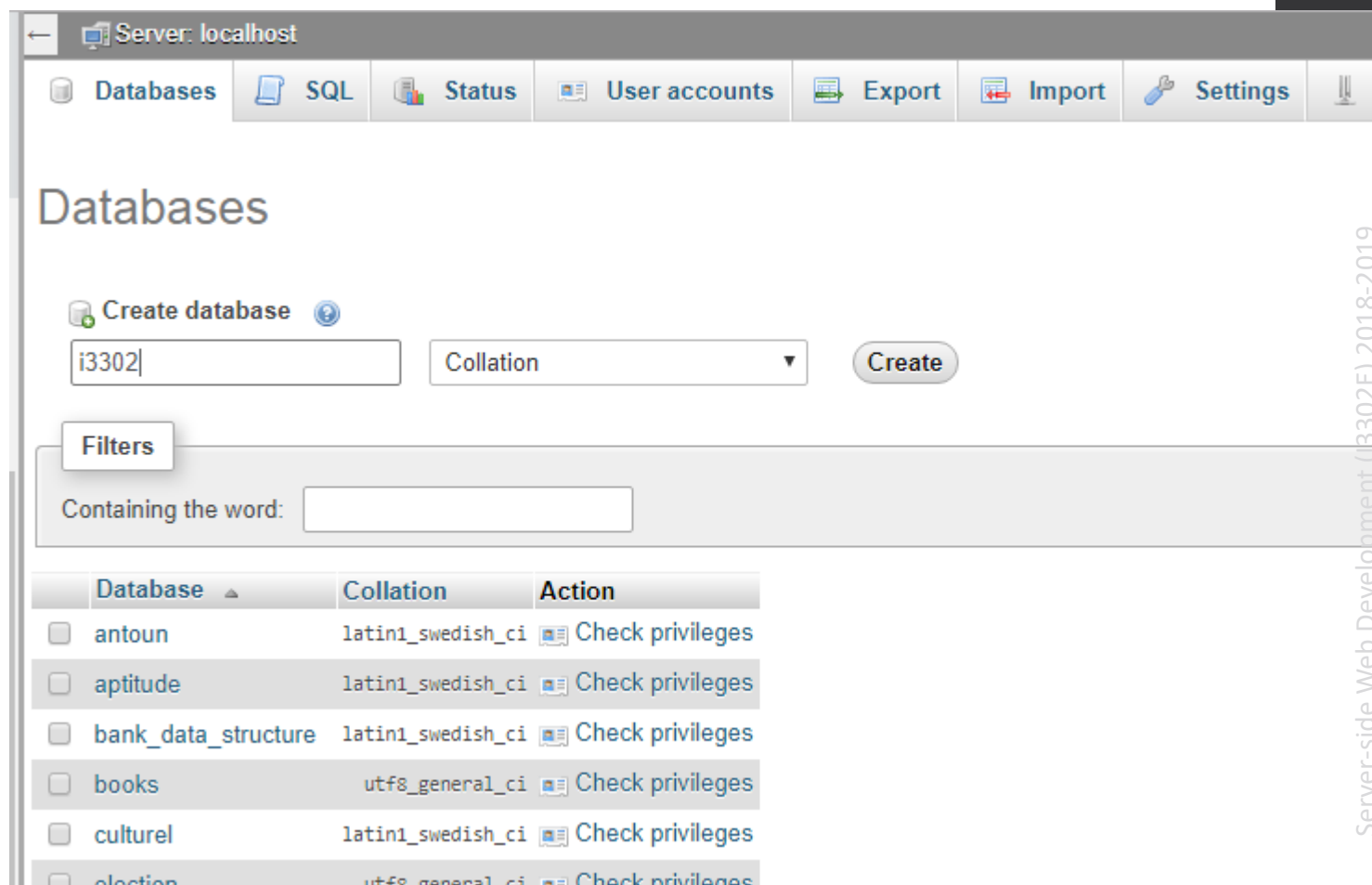
Drop Database Tables

- Through the **Drop** action you can delete the whole table and all the records stored in it.



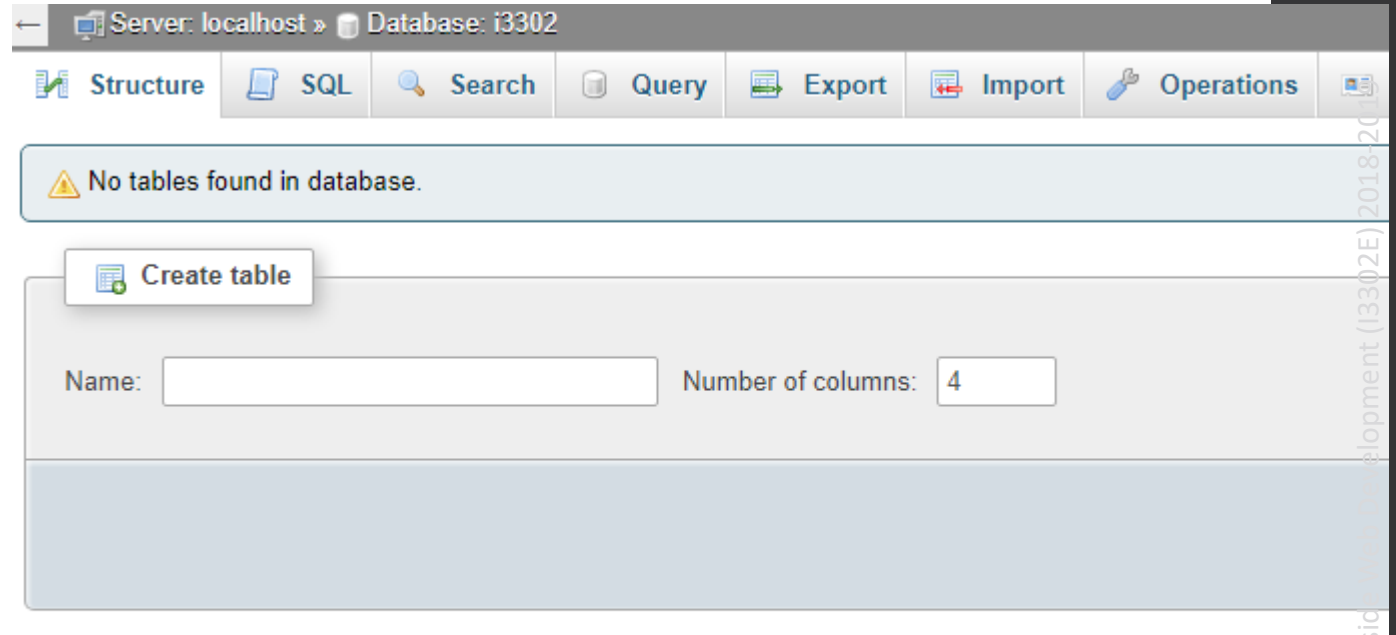
Create a MySQL Database?

- Enter the database name in the text field and click on the **Create** button.



Add MySQL Database Tables

- Navigate to your **PhpMyAdmin** tool and open the newly create database. It is empty and there are no tables.
- Enter the table name and the number of fields. Click on the **Go** button to create the table.



The screenshot shows the PhpMyAdmin web interface. At the top, the browser address bar indicates 'Server: localhost » Database: i3302'. Below this is a navigation bar with tabs for 'Structure', 'SQL', 'Search', 'Query', 'Export', 'Import', and 'Operations'. A message box states 'No tables found in database.' Below this, a 'Create table' button is visible. The 'Name:' field is empty, and the 'Number of columns:' field is set to '4'. The 'Go' button is not visible in this view.

Add MySQL Database Tables

- On the next screen you should enter the fields' names and the corresponding properties. The properties are:
 - **Type**
 - Here you should pick the type of the data, which will be stored in the corresponding field. More details about the possible choices can be found in the official MySQL Data Types documentation.
 - **Length/Values**
 - Here you should enter the length of the field. If the field type is "enum" or "set", enter the values using the following format: 'a','b','c'...

Add MySQL Database Tables

- **Collation**

- Pick the data collation for each of the fields: A collation is a set of rules that defines how to compare and sort character strings. Each collation in MySQL belongs to a single character set. Every character set has at least one collation, and most have two or more collations. A collation orders characters based on weights.

Add MySQL Database Tables

- **Attributes**

- The possible attributes' choices are:
- `BINARY` - the collation for the field will be binary, for example `utf8_bin`;
- `UNSIGNED` - the field numeric values will be positive or 0;
- `UNSIGNED ZEROFILL` - the field numeric values will be positive or 0 and leading zeros will be added to a number;
- `ON UPDATE CURRENT_TIMESTAMP` - the value for a data type field has the current timestamp as its default value, and is automatically updated;

Add MySQL Database Tables

- **Null**

- Here you define whether the field value can be NULL. More about the NULL value can be found in the corresponding MySQL documentation.

- **Default**

- This property allows you to set the default value for the field.

Add MySQL Database Tables

- **Extra**

- In the Extra property you can define whether the field value is auto-increment.
- The radio buttons that come below define whether there is an **Index** defined for the particular field and specify the **Index**type.

- **Comments**

- Here you can add comments, which will be included in the database sql code.
- At the end you can include **Table comments** and pick the MySQL **Storage Engine** and the **Collation**. Once you are ready, click on the **Save** button.

Server: localhost » Database: i3302 » Table: i3302

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Triggers

Table name: Add column(s)

Structure ⓘ

Name	Type ⓘ	Length/Values ⓘ	Default ⓘ	Collation	Attributes	Null	Index
<input type="text"/>	INT ▼	<input type="text"/>	None ▼	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---
<input type="text"/>	INT ▼	<input type="text"/>	None ▼	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---
<input type="text"/>	INT ▼	<input type="text"/>	None ▼	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---
<input type="text"/>	INT ▼	<input type="text"/>	None ▼	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	---

Table comments:

Collation:

Storage Engine: ⓘ

InnoDB ▼

PARTITION definition: ⓘ

Partition by: ()

Partitions:

Add MySQL Database Tables

- If you want to add more fields you should specify their number and click on the **Go** button instead of **Save**.
- The database table will be created and you will see the corresponding MySQL query.

The screenshot shows the MySQL Table Structure tool interface. At the top, there are tabs for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, and Operations. Below these, there are buttons for Table structure and Relation view. The main area displays a table structure with the following columns: #, Name, Type, Collation, Attributes, Null, Default, Comments, Extra, and Action.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique
2	name	varchar(250)	utf8_general_ci		No	None			Change Drop Primary Unique

Below the table, there are buttons for Check all, With selected, Browse, Change, Drop, Primary, Unique, and Index. At the bottom, there are buttons for Print, Propose table structure, Move columns, and Improve table structure. At the very bottom, there is a form to add a new column: "Add 1 column(s) after name" with a "Go" button.

- Now we will proceed with the populating of the table with data.

Add Content in a Database Table

- In order to add records in a database table click on the **Insert** tab.
- Enter the data in the corresponding fields and click on the **Go** button to store it.
- At the bottom of the page you will see a drop-down menu labelled **Restart insertion with x rows** . There you can pick the number of the rows that you can populate with data and insert at once. By default the value is 2.
- The **Ignore** check box will allow you to ignore the data entered below it. It will not be added.

← Server: localhost » Database: i3302 » Table: i3302

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#)

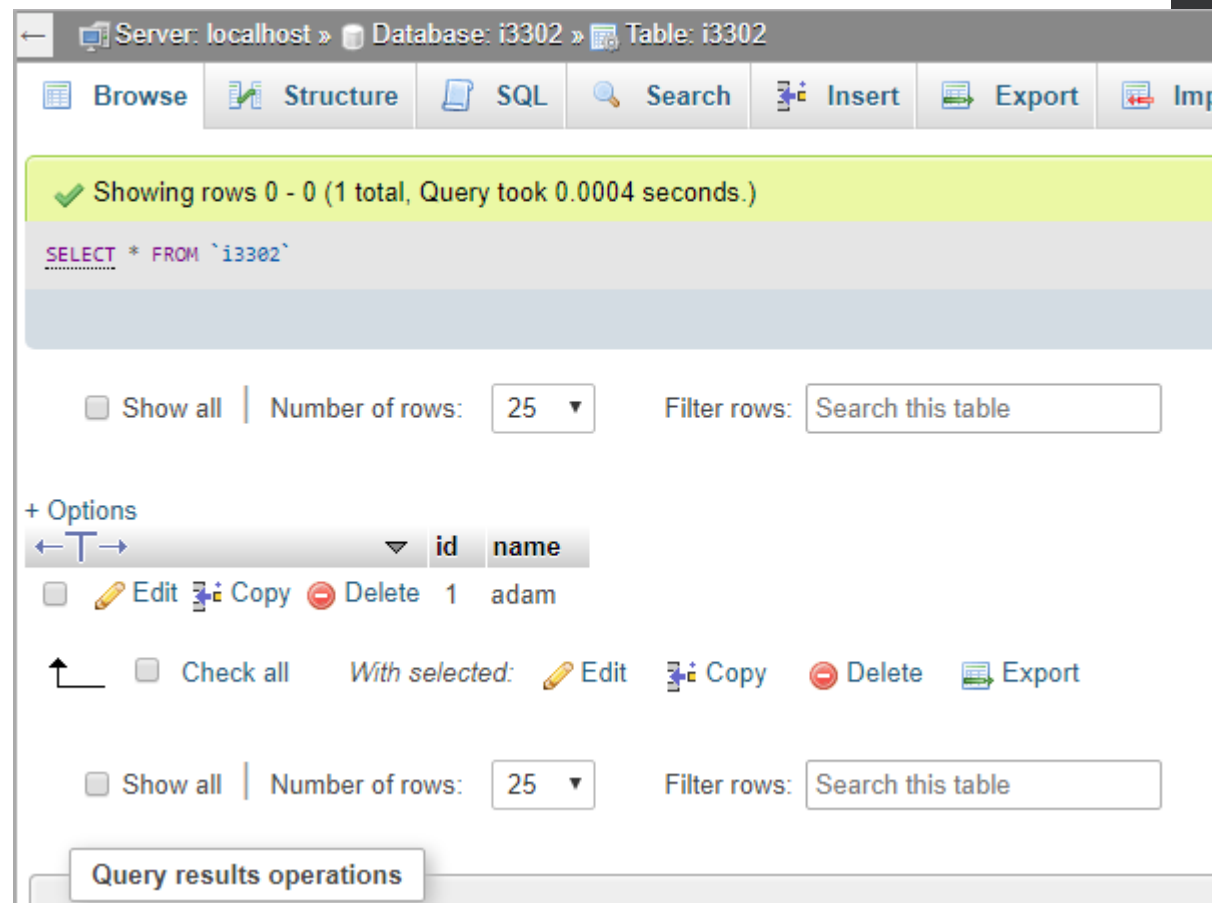
Column	Type	Function	Null	Value
id	int(11)	<input type="text"/>		<input type="text"/>
name	varchar(250)	<input type="text"/>		<input type="text"/>

☒ Ignore

Column	Type	Function	Null	Value
id	int(11)	<input type="text"/>		<input type="text"/>

Add Content in a Database Table

- You can see the newly inserted record by clicking on the **Browse** tab.
- You can edit or delete the record by clicking on the corresponding icons.
- To insert more records, return to the **Insert** tab and repeat the procedure.



Backup a Database

- Once you are ready, you can create a backup of your database through the **Export** tab.
- Select the tables which you want to be exported.
- Leave the radio button selection to the **SQL** option. The **Structure** and the **Data** check boxes should remain checked.
- Select the **Save as file** check box and then click on the **Go** button.
- In this way you will save the dump SQL file with your database structure and content on your local computer.

← Server: localhost » Database: i3302 » Table: i3302

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Exporting rows from "i3302" table

Export method:

☐ Quick - display only the minimal options

☒ Custom - display all possible options

Format:

SQL ▼

Rows:

☐ Dump some row(s)

Number of rows:

Row to begin at:

☒ Dump all rows

Output:

☐ Rename exported databases/tables/columns

☐ Use LOCK TABLES statement

Restore a Database Backup

- You can restore your database backup from the **Import** tab.
- Click on the **Browse** button to select your database backup file from your local computer.
- Pick the charset of the file from the corresponding drop-down menu.
- If the file is too big, the MySQL server timeout can be reached. In such a case you can interrupt the import action. Then you can continue with the data import defining the number of the queries to be skipped from the file beginning. In this way you will skip the imported queries and continue from the point of the interruption.

← Server: localhost » Database: i3302 » Table: i3302

BrowseStructureSQLSearchInsertExportImportPrivileges

Importing into the table "i3302"

File to import:

File may be compressed (gzip, zip) or uncompressed.
A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Browse your computer: No file chosen (Max: 120MiB)

You may also drag and drop a file on any page.

Character set of the file:

Partial import:

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to in*

Skip this number of queries (for SQL) starting from the first one:

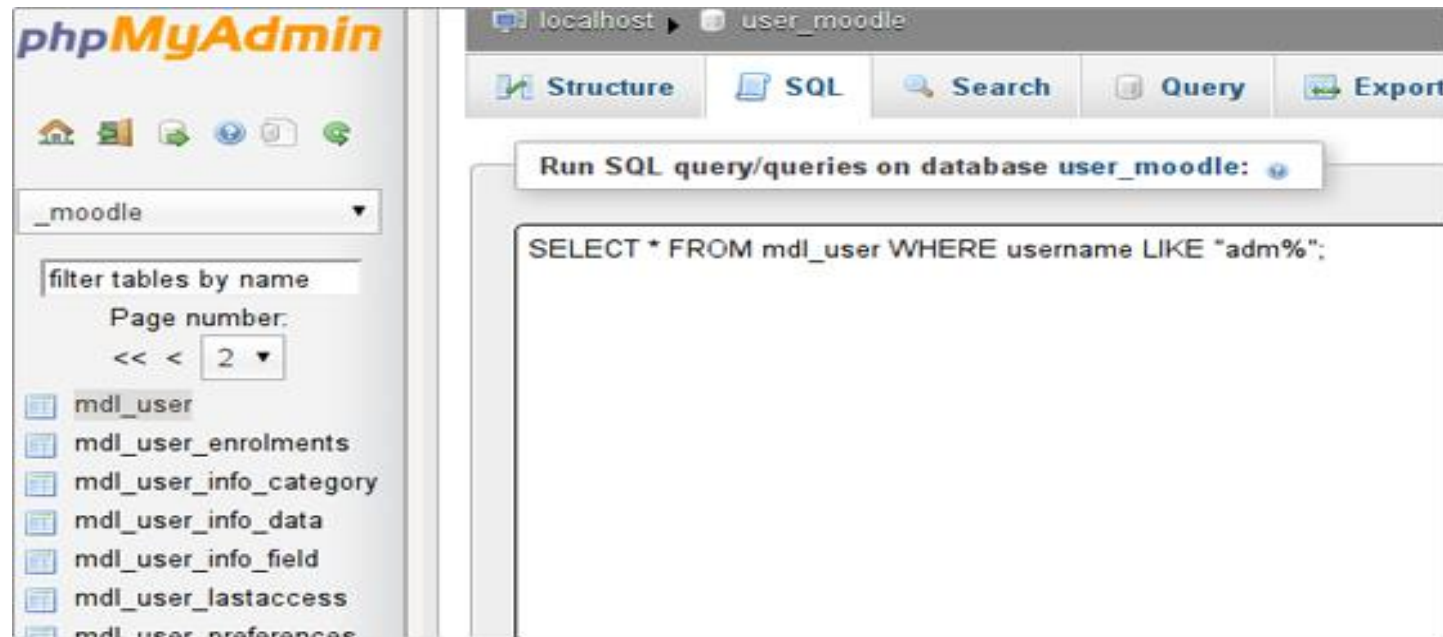
Other options:

☒ Enable foreign key checks

Format:

Run MySQL Query

- The one for advanced users is through the **SQL** tab. There you should enter the entire SQL query code and click on the **Go** button to execute it.



Run MySQL Query

- The other option is to construct a MySQL query in the **Query** tab.

The screenshot shows a web application interface for running MySQL queries. The browser address bar indicates the URL is localhost/user_moodle. The interface has a top navigation bar with tabs: Structure, SQL, Search, Query (selected), Export, Import, and Operations. Below the tabs, there are two columns for configuring the query. The first column has a 'Column:' dropdown set to 'mdl_user.*', a 'Sort:' dropdown, and a 'Show:' checkbox that is checked. The second column has a 'Column:' dropdown set to 'mdl_user.username', a 'Sort:' dropdown, and a 'Criteria:' dropdown set to 'LIKE "adm%"'. Below these, there are sections for 'Criteria' and 'Modify' with radio buttons for 'Ins' and 'Del' and 'Or' and 'And' operators. At the bottom, there are two dropdowns for 'Add/Delete criteria rows' and 'Add/Delete columns', both set to 0, and an 'Update Query' button.

Run MySQL Query

- There you can define different search conditions, sort the results and query multiple tables.
- You should select the tables used in the query from the **Use Tables** list.
- The fields which will be included in the **SELECT** MySQL statement should be picked from the **Field** drop-down menus. The **Show** check box should be selected.
- In the **Criteria** text field you should enter the criteria according to which the search will be completed.
- Through the **Sort** drop-down menu you can visualize the result sorted in an ascending or a descending order.
- The text window located below allows you do add extra search conditions.

Run MySQL Query

- Additionally, you can use the **Ins** and the **Del** check boxes to add or delete text rows for search conditions. The same can be performed through the **Add/Delete Criteria Row** drop-down menu. To add or delete columns use the corresponding **Ins** and **Del** check boxes or the **Add/Delete Field Columns** drop-down menu.
- In the **Modify** section you can define the relations between the fields (whether they are connected through the **AND** or the **OR** logical operators).
- You need to click on the **Update Query** button to complete the modifications.
- To run the query click on the **Submit Query** button.
- The query which we have included in our example is:
- *SELECT `AT_admins`. * FROM AT_admins WHERE (`AT_admins`.`login` LIKE "a%");*
- It shows all the records from the **AT_admins** table for which the **login** field starts with "a".