



Introduction to MySQL

Road Map

- Introduction to MySQL
- Connecting and Disconnecting
- Entering Basic Queries
- Creating and Using a Database

Attribution

- Most of these slides are based directly on the MySQL Documentation.
- Most information comes from Chapter 3, MySQL Tutorial:
- http://www.mysql.com/documentation/mysql/bychapter/manual_Tutorial.html#Tutorial

MySQL

- MySQL is a very popular, open source database.
- Officially pronounced “my Ess Que Ell” (not my sequel).
- Handles very large databases; very fast performance.
- Why are we using MySQL?
 - Free (much cheaper than Oracle!)
 - Each student can install MySQL locally.
 - Easy to use Shell for creating tables, querying tables, etc.
 - Easy to use with Java JDBC and PHP

Crash Course Fundamentals

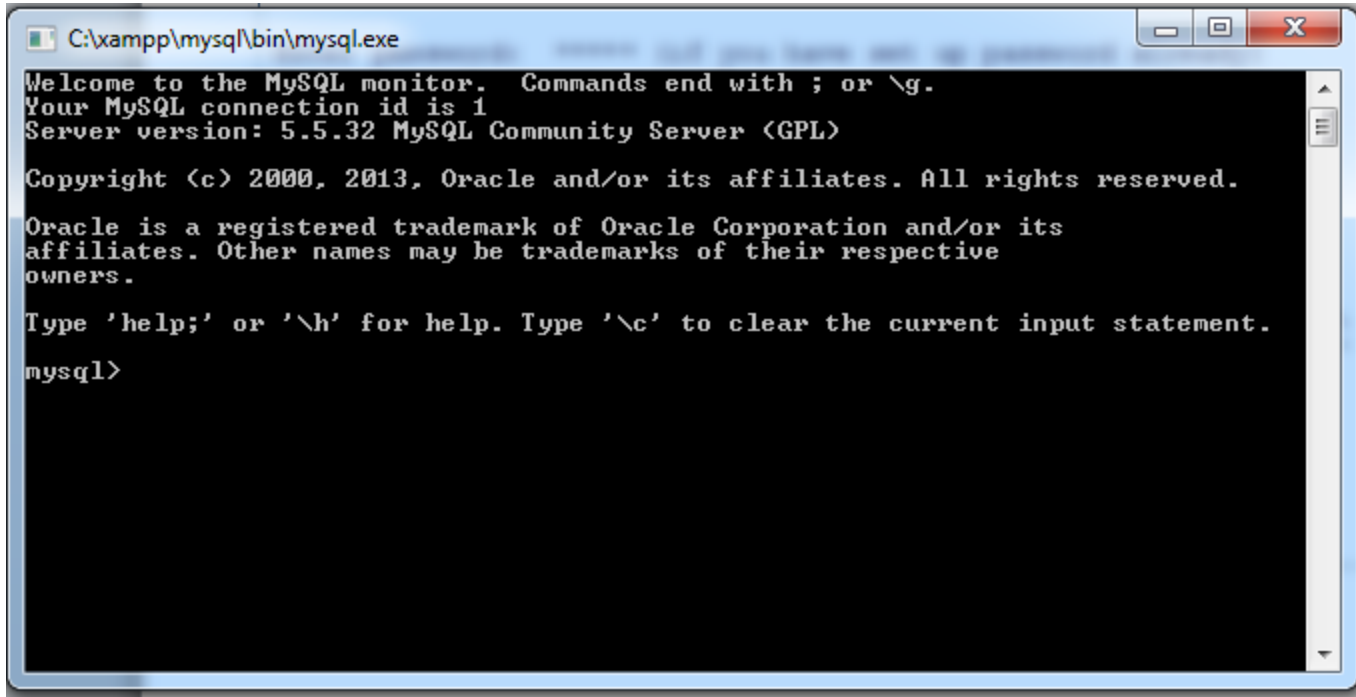
- In order to use JDBC, you need:
 - a database.
 - basic understand of SQL (Structured Query Language)
- Some students may still not sure how database works
- The purpose of this lecture is to get all students up to speed on database fundamentals.

Connecting to MySQL

- MySQL provides an interactive shell for creating tables, inserting data, etc.
- In XAMPP environment, Start MySQL at XAMPP Control Panel, then just go to XAMPP\mysql\bin, and click mysql

Sample Session

- For example:

A screenshot of a Windows command prompt window titled "C:\xampp\mysql\bin\mysql.exe". The window has a black background with white text. The text displayed is: "Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.32 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>"

```
C:\xampp\mysql\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.32 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

- To exit the MySQL Shell, just type QUIT or EXIT:

```
mysql> QUIT
mysql> exit
```

Basic Queries

- Once logged in, you can try some simple queries.
- For example:

```
mysql> SELECT VERSION() , CURRENT_DATE;  
+-----+-----+  
| VERSION() | CURRENT_DATE |  
+-----+-----+  
| 5.5.32    | 2013-11-05    |  
+-----+-----+  
1 row in set (0.00 sec)
```

- Note that most MySQL commands end with a semicolon (;)
- MySQL returns the total number of rows found, and the total time to execute the query.

Basic Queries

- Keywords may be entered in any lettercase.
- The following queries are equivalent:

```
mysql> SELECT VERSION() , CURRENT_DATE;  
mysql> select version() , current_date;  
mysql> SeLeCt vErSiOn() , current_DATE;
```

Basic Queries

- Here's another query. It demonstrates that you can use mysql as a simple calculator:

```
mysql> SELECT SIN(PI()/4) , (4+1)*5;  
+-----+-----+  
| SIN(PI()/4) | (4+1)*5 |  
+-----+-----+  
|      0.707107 |      25 |  
+-----+-----+
```

Basic Queries

- You can also enter multiple statements on a single line. Just end each one with a semicolon:

```
mysql> SELECT VERSION() ; SELECT NOW() ;
```

```
+-----+  
| VERSION() |  
+-----+  
| 5.5.32a-log |  
+-----+  
  
+-----+  
| NOW() |  
+-----+  
| 2013-11-05 20:15:33 |  
+-----+
```

Multi-Line Commands

- mysql determines where your statement ends by looking for the terminating semicolon, not by looking for the end of the input line.
- Here's a simple multiple-line statement:

```
mysql> SELECT
-> USER()
-> ,
-> CURRENT_DATE;

+-----+-----+
| USER()                | CURRENT_DATE |
+-----+-----+
| ODBC@localhost        | 2013-11-05   |
+-----+-----+
```

Canceling a Command

- If you decide you don't want to execute a command that you are in the process of entering, cancel it by typing \c

```
mysql> SELECT  
      -> USER()  
      -> \c  
mysql>
```

Using a Database

- To get started on your own database, first check which databases currently exist.
- Use the SHOW statement to find out which databases currently exist on the server:

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| mysql    |  
| test     |  
+-----+  
2 rows in set (0.01 sec)
```

Using a Database

- To create a new database, issue the “create database” command:
 - `mysql> create database webdb;`
- To the select a database, issue the “use” command:
 - `mysql> use webdb;`

Creating a Table

- Once you have selected a database, you can view all database tables:

```
mysql> show tables;
```

```
Empty set (0.02 sec)
```

- An empty set indicates that I have not created any tables yet.

Creating a Table

- Let's create a table for storing pets.

- Table: pets

➤ name: VARCHAR(20)


➤ owner: VARCHAR(20)

➤ species: VARCHAR(20)

➤ sex: CHAR(1)

➤ birth: DATE

➤ date: DATE



VARCHAR is usually used to store string data.

Creating a Table

- To create a table, use the CREATE TABLE command:

```
mysql> CREATE TABLE pet (  
    -> name VARCHAR(20) ,  
    -> owner VARCHAR(20) ,  
    -> species VARCHAR(20) ,  
    -> sex CHAR(1) ,  
    -> birth DATE, death DATE) ;  
Query OK, 0 rows affected (0.04 sec)
```

Showing Tables

- To verify that the table has been created:

```
mysql> show tables;
```

```
+-----+
```

```
| Tables_in_test |
```

```
+-----+
```

```
| pet            |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

Describing Tables

- To view a table structure, use the DESCRIBE command:

```
mysql> describe pet;
```

| Field | Type | Null | Key | Default | Extra |
|---------|-------------|------|-----|---------|-------|
| name | varchar(20) | YES | | NULL | |
| owner | varchar(20) | YES | | NULL | |
| species | varchar(20) | YES | | NULL | |
| sex | char(1) | YES | | NULL | |
| birth | date | YES | | NULL | |
| death | date | YES | | NULL | |

```
6 rows in set (0.02 sec)
```

Deleting a Table

- To delete an entire table, use the DROP TABLE command:

```
mysql> drop table pet;
```

```
Query OK, 0 rows affected (0.02 sec)
```

Loading Data

- Use the INSERT statement to enter data into a table.
- For example:

```
INSERT INTO pet VALUES  
    ('Fluffy', 'Harold', 'cat', 'f',  
     '2013-02-04', NULL);
```

- The next slide shows a full set of sample data.

More data...

| name | owner | species | sex | birth | death |
|-----------------|---------------|----------------|------------|-------------------|-------------------|
| Fluffy | Harold | cat | f | 2013-02-04 | |
| Claws | Gwen | cat | m | 2011-03-17 | |
| Buffy | Harold | dog | f | 2010-05-13 | |
| Fang | Benny | dog | m | 2012-08-27 | |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |
| Chirpy | Gwen | bird | f | 2008-09-11 | |
| Whistler | Gwen | bird | | 2012-12-09 | |
| Slim | Benny | snake | m | 2009-04-29 | |

Loading Sample Data

- You could create a text file `pet.txt' containing one record per line.
- Values must be separated by tabs, and given in the order in which the columns were listed in the CREATE TABLE statement.
- Then load the data via the LOAD DATA Command.

Sample Data File

| | | | | | |
|----------|--------|-------|----|------------|------------|
| Fluffy | Harold | cat | f | 2013-02-04 | \N |
| Claws | Gwen | cat | m | 2011-03-17 | \N |
| Buffy | Harold | dog | f | 2010-05-13 | \N |
| Fang | Benny | dog | m | 2012-08-27 | \N |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |
| Chirpy | Gwen | bird | f | 2008-09-11 | \N |
| Whistler | Gwen | bird | \N | 2012-12-09 | \N |
| Slim | Benny | snake | m | 2009-04-29 | \N |

To Load pet.txt:

mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;

For each of the examples,
assume the following set of data.

| name | owner | species | sex | birth | death |
|-----------------|---------------|----------------|------------|-------------------|-------------------|
| Fluffy | Harold | cat | f | 2013-02-04 | |
| Claws | Gwen | cat | m | 2011-03-17 | |
| Buffy | Harold | dog | f | 2010-05-13 | |
| Fang | Benny | dog | m | 2012-08-27 | |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |
| Chirpy | Gwen | bird | f | 2008-09-11 | |
| Whistler | Gwen | bird | | 2012-12-09 | |
| Slim | Benny | snake | m | 2009-04-29 | |

SQL Select

- The SELECT statement is used to pull information from a table.
- The general format is:

```
SELECT what_to_select  
FROM which_table  
WHERE conditions_to_satisfy
```

Selecting All Data

- The simplest form of SELECT retrieves everything from a table

```
mysql> select * from pet;
```

| name | owner | species | sex | birth | death |
|----------|--------|---------|-----|------------|------------|
| Fluffy | Harold | cat | f | 2013-02-04 | NULL |
| Claws | Gwen | cat | f | 2011-03-17 | NULL |
| Buffy | Harold | dog | f | 2010-05-13 | NULL |
| Fang | Benny | dog | m | 2012-08-27 | NULL |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |
| Chirpy | Gwen | bird | f | 2008-09-11 | NULL |
| Whistler | Gwen | bird | | 2012-12-09 | NULL |
| Slim | Benny | snake | m | 2009-04-29 | NULL |

```
8 rows in set (0.00 sec)
```

Selecting Particular Rows

- You can select only particular rows from your table.
- For example, if you want to verify the change that you made to Bowser's birth date, select Bowser's record like this:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
```

| | | | | | | |
|--------|-------|---------|-----|------------|------------|--|
| name | owner | species | sex | birth | death | |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 | |

```
1 row in set (0.00 sec)
```

Selecting Particular Rows

- To find all animals born after 2008

```
SELECT * FROM pet WHERE birth >= "2008-1-1";
```

- To find all female dogs, use a logical AND

```
SELECT * FROM pet WHERE species = "dog" AND sex = "f";
```

- To find all snakes or birds, use a logical OR

```
SELECT * FROM pet WHERE species = "snake"  
OR species = "bird";
```

Selecting Particular Columns

- If you don't want to see entire rows from your table, just name the columns in which you are interested, separated by commas.
- For example, if you want to know when your pets were born, select the name and birth columns.
- (see example next slide.)

Selecting Particular Columns

```
mysql> select name, birth from pet;
```

| name | birth |
|----------|------------|
| Fluffy | 2013-02-04 |
| Claws | 2011-03-17 |
| Buffy | 2010-05-13 |
| Fang | 2012-08-27 |
| Bowser | 1996-08-31 |
| Chirpy | 2008-09-11 |
| Whistler | 2012-12-09 |
| Slim | 2009-04-29 |

```
8 rows in set (0.01 sec)
```


Sorting Data

- To sort a result, use an ORDER BY clause.
- For example, to view animal birthdays, sorted by date:

```
mysql> SELECT name, birth FROM pet  
ORDER BY birth;
```

- To sort in reverse order, add the DESC (descending keyword)

```
mysql> SELECT name, birth FROM pet  
ORDER BY birth DESC;
```

Working with NULLs

- NULL means missing value or unknown value.
- To test for NULL, you cannot use the arithmetic comparison operators, such as =, < or <>.
- Rather, you must use the IS NULL and IS NOT NULL operators instead.

Working with NULLs

- For example, to find all your dead pets (what a morbid example!)

```
mysql> select name from pet where death  
      >IS NOT NULL;
```

```
+-----+
```

```
| name  |
```

```
+-----+
```

```
| Bowser |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

Pattern Matching

- MySQL provides:
 - standard SQL pattern matching; and
 - regular expression pattern matching, similar to those used by Unix utilities such as vi, grep and sed.
- SQL Pattern matching:
 - To perform pattern matching, use the LIKE or NOT LIKE comparison operators
 - By default, patterns are case insensitive.
- Special Characters:
 - _ Used to match any single character.
 - % Used to match an arbitrary number of characters.

Pattern Matching Example

- To find names beginning with 'b':

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
```

| name | owner | species | sex | birth | death |
|--------|--------|---------|-----|------------|------------|
| Buffy | Harold | dog | f | 2010-05-13 | NULL |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |

Pattern Matching Example

- To find names ending with `fy`:

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";
```

| name | owner | species | sex | birth | death |
|--------|--------|---------|-----|------------|-------|
| Fluffy | Harold | cat | f | 2013-02-04 | NULL |
| Buffy | Harold | dog | f | 2010-05-13 | NULL |

Pattern Matching Example

- To find names containing a 'w':

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";
```

| name | owner | species | sex | birth | death |
|----------|-------|---------|------|------------|------------|
| Claws | Gwen | cat | m | 2011-03-17 | NULL |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |
| Whistler | Gwen | bird | NULL | 2012-12-09 | NULL |

Pattern Matching Example

- To find names containing exactly five characters, use the `_` pattern character:

```
mysql> SELECT * FROM pet WHERE name LIKE "_____";
```

| name | owner | species | sex | birth | death |
|-------|--------|---------|-----|------------|-------|
| Claws | Gwen | cat | m | 2011-03-17 | NULL |
| Buffy | Harold | dog | f | 1996-05-13 | NULL |

Regular Expression Matching

- The other type of pattern matching provided by MySQL uses extended regular expressions.
- When you test for a match for this type of pattern, use the REGEXP and NOT REGEXP operators (or RLIKE and NOT RLIKE, which are synonyms).

Regular Expressions

- Some characteristics of extended regular expressions are:
 - `.` matches any single character.
 - A character class `[...]` matches any character within the brackets. For example, `[abc]` matches a, b, or c. To name a range of characters, use a dash. `[a-z]` matches any lowercase letter, whereas `[0-9]` matches any digit.
 - `*` matches zero or more instances of the thing preceding it. For example, `x*` matches any number of x characters, `[0-9]*` matches any number of digits, and `.*` matches any number of anything.
 - To anchor a pattern so that it must match the beginning or end of the value being tested, use `^` at the beginning or `$` at the end of the pattern.

Reg Ex Example

- To find names beginning with b, use ^ to match the beginning of the name:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
```

| name | owner | species | sex | birth | death |
|--------|--------|---------|-----|------------|------------|
| Buffy | Harold | dog | f | 2010-05-13 | NULL |
| Bowser | Diane | dog | m | 1996-08-31 | 2012-07-29 |

Reg Ex Example

- To find names ending with `fy`, use `\$` to match the end of the name:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
```

| name | owner | species | sex | birth | death |
|--------|--------|---------|-----|------------|-------|
| Fluffy | Harold | cat | f | 2013-02-04 | NULL |
| Buffy | Harold | dog | f | 2010-05-13 | NULL |

Counting Rows

- Databases are often used to answer the question, "How often does a certain type of data occur in a table?"
- For example, you might want to know how many pets you have, or how many pets each owner has.
- Counting the total number of animals you have is the same question as "How many rows are in the pet table?" because there is one record per pet.
- The COUNT() function counts the number of non-NULL results.

Counting Rows Example

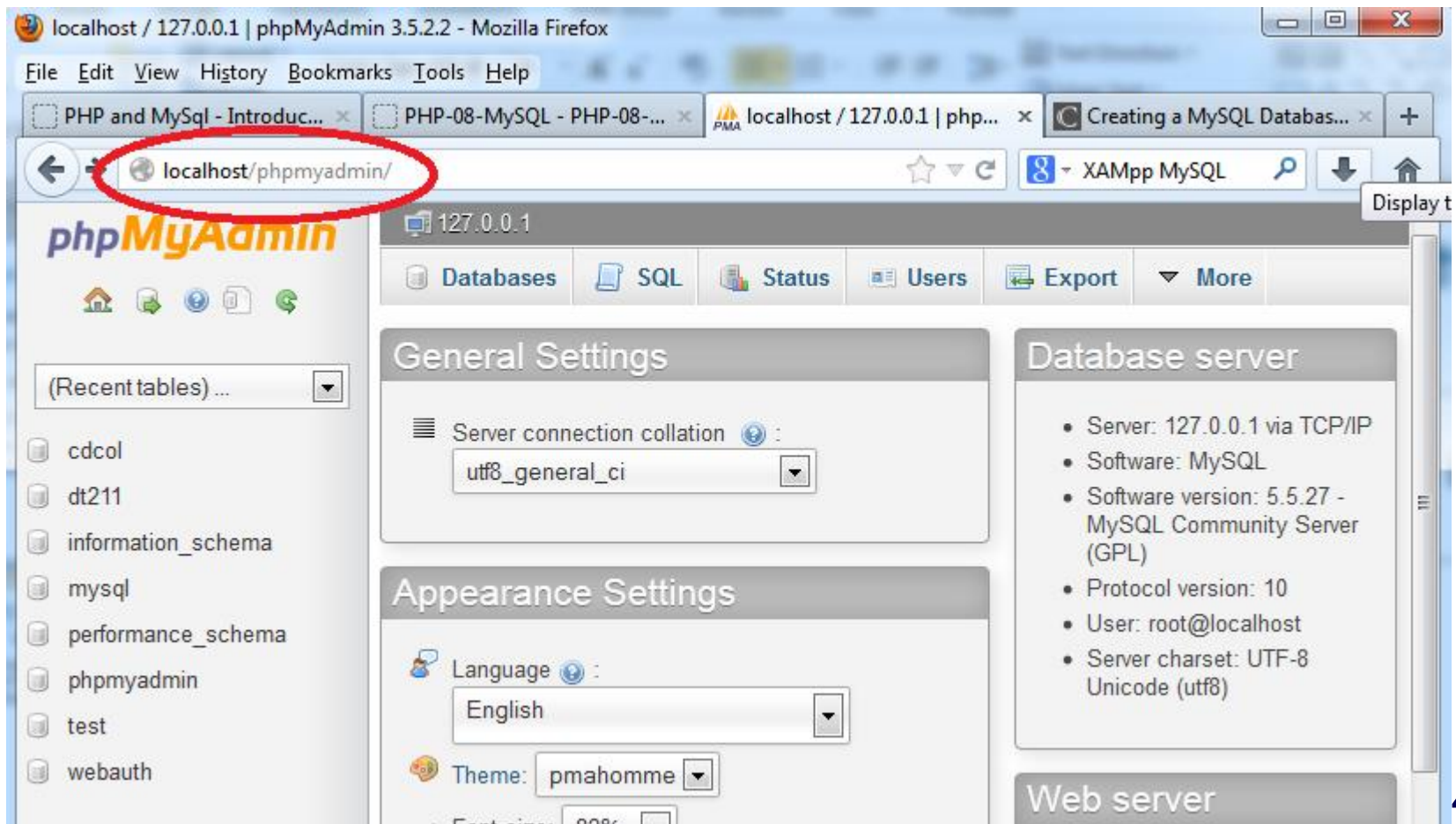
- A query to determine total number of pets:

```
mysql> SELECT COUNT(*) FROM pet;
```

```
+-----+  
| COUNT(*) |  
+-----+  
|          9 |  
+-----+
```

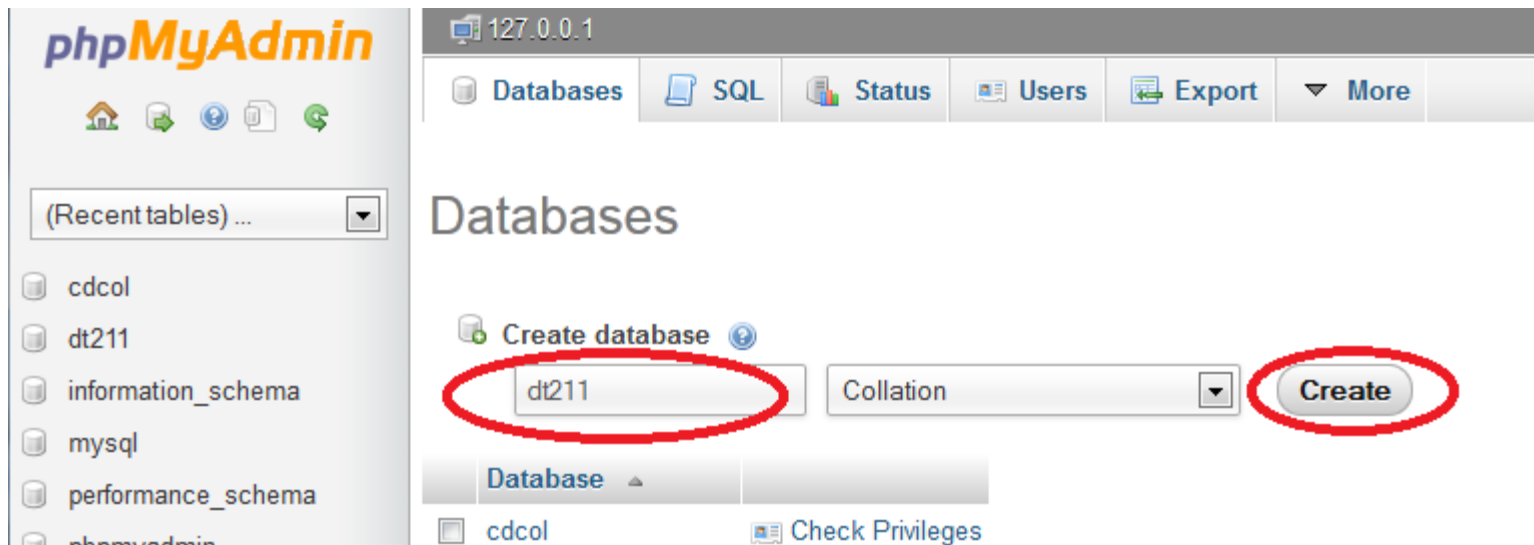
Creating a MySQL database using XAMPP

- Open your browser and enter `http://localhost/phpmyadmin`. This will bring you to



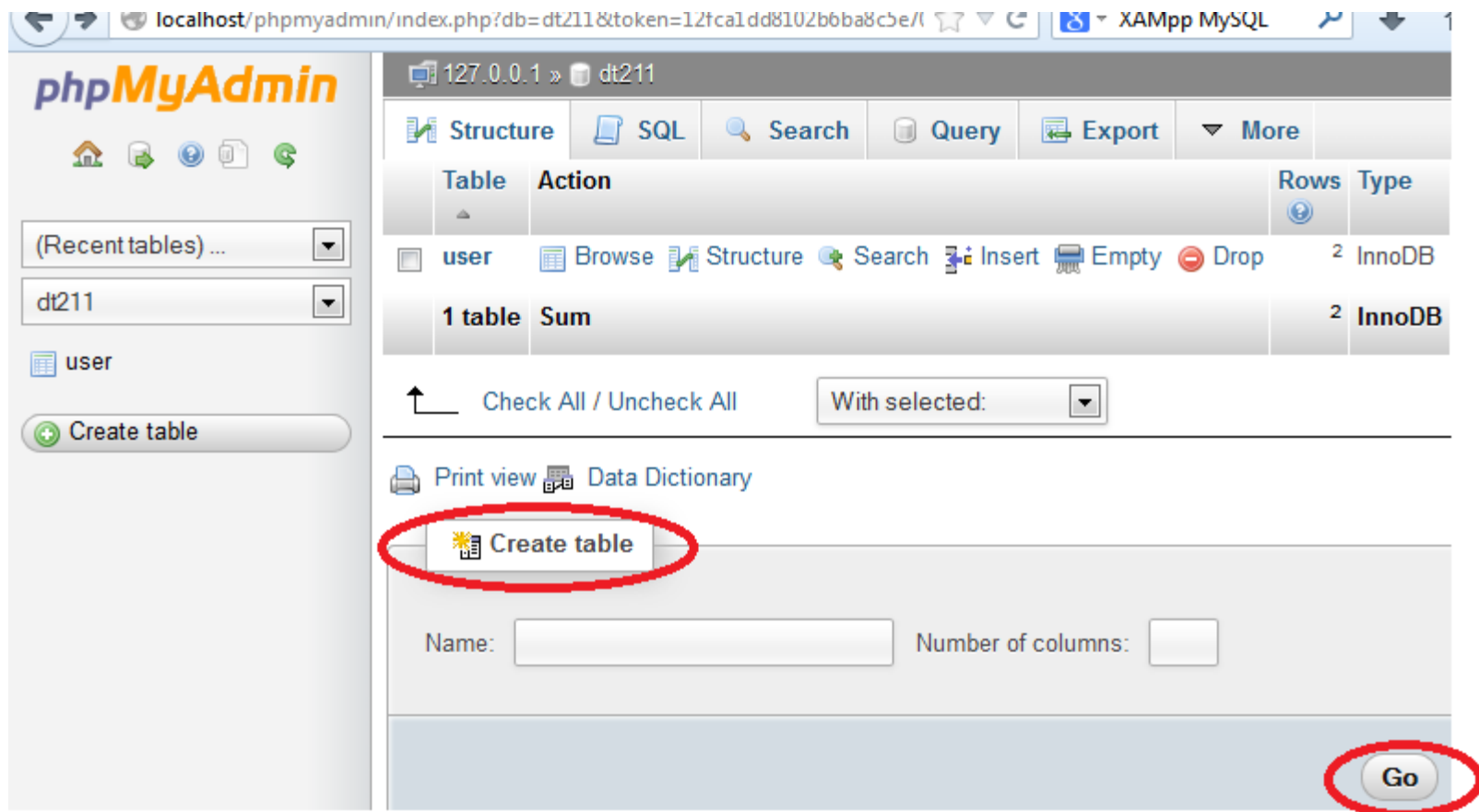
Creating a MySQL database using XAMPP

- Enter a name for the database, then click on the Create button. The name must be 64 characters or less and composed of letters, numbers and underscores. Avoid using numbers at the start of the name. underscores. Avoid using numbers at the start of the name.



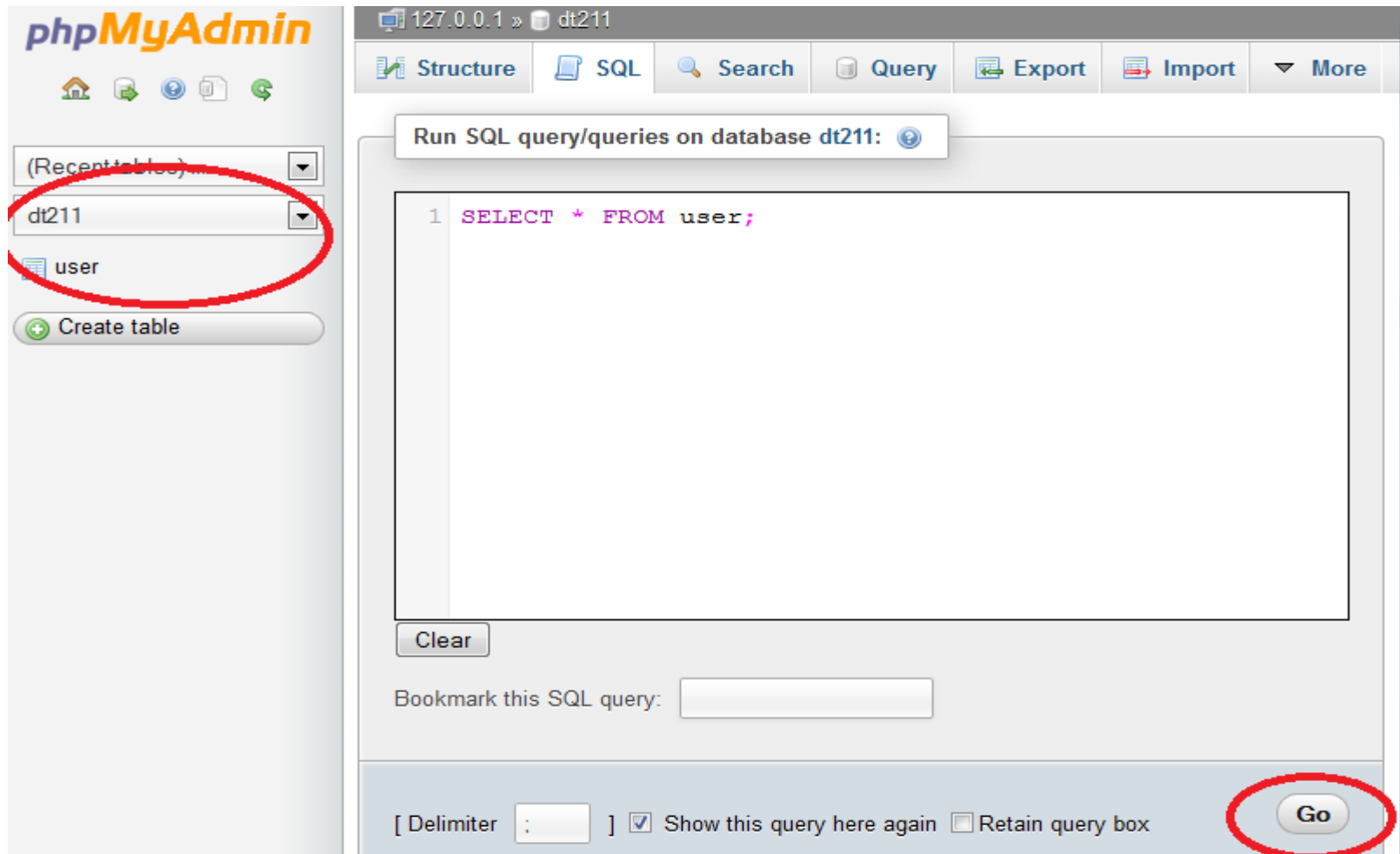
Creating a MySQL database using XAMPP

- Ensure the database was successfully created. Then you can start to create tables and relationships among tables.



Creating a MySQL database using XAMPP

- You can also run SQL query to create/alter tables and manipulate data in tables.



Creating a MySQL database using XAMPP

- YouTube is one of the greatest resources for tutorials and helps
- PHP MySQL Tutorial: Create a Database and Table in phpMyAdmin

<http://www.youtube.com/watch?v=n7c5zMk8cx4>

- PHP MySQL Tutorial: Connect to a MySQL Database

<http://www.youtube.com/watch?v=5sQ4scQz0qY>

Access MySQL Using PHP

- MySQL is Built Into PHP
- Before we can access data in a database, we must open a connection to the MySQL server.
- In PHP, this is done with the `mysqli_connect()` function.

Syntax:

```
mysqli_connect (host, username, password, dbname) ;
```

Access MySQL Using PHP

| Parameter | Description |
|-----------|---|
| host | Optional. Either a host name or an IP address |
| username | Optional. The MySQL user name |
| password | Optional. The password to log in with |
| dbname | Optional. The default database to be used when performing queries |

Note: There are more available parameters, but the ones listed above are the most important.

Access MySQL Using PHP

In the following example we store the connection in a variable (\$con) for later use in the script:

```
1  <?php
2  // Create connection
3  $con=mysqli_connect("localhost","root","","dt211");
4
5  // Check connection
6  if (mysqli_connect_errno($con))
7  {
8      echo "Failed to connect to MySQL: " . mysqli_connect_error();
9  }
10 $result = mysqli_query($con,"SELECT * FROM user");
11
12 while($row = mysqli_fetch_array($result))
13 {
14     echo $row['FirstName'] . " " . $row['LastName'];
15     echo "<br />";
16 }
17
18 mysqli_close($con);
19 ?>
```

Insert Data Into a Table

The INSERT INTO statement is used to add new records to a database table.

Syntax

It is possible to write the INSERT INTO statement in two forms. The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name  
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)  
VALUES (value1, value2, value3,...)
```

Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the “user” table.

Here is the HTML form:

```
1  <html>
2  <body>
3
4  <form action="insert.php" method="post">
5      User Name: <input type="text" name="username">
6      Password: <input type="text" name="password">
7      First Name: <input type="text" name="firstname">
8      Last Name: <input type="text" name="lastname">
9
10     <input type="submit">
11 </form>
12
13 </body>
14 </html>
```


Insert Data From a Form Into a Database

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".

Here is the "insert.php" page:

```
<?php
$con=mysqli_connect("localhost","root","","dt211");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$sql="INSERT INTO user (UserName, Password, FirstName, LastName)
VALUES
('$ _POST[username]', '$ _POST[password]', '$ _POST[firstname]', '$ _POST[lastname]')";

if (!mysqli_query($con,$sql))
{
    die('Error: ' . mysqli_error());
}
echo "1 record added";

mysqli_close($con);
?>
```