**PHP and MySQL**

**Session 5 Practical : OOPHP & MySQL**

**Object Oriented PHP**

**TASK 1**

Complete Steps 1-11 of this tutorial: <https://www.killerphp.com/tutorials/php-objects-page-1/>

Screenshot your final code and the browser output when you are finished.

Graphical user interface, text

Description automatically generated

**Using MySQL**

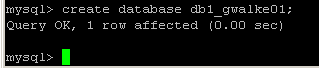
Our first step would be to create a database in which we will define our tables. In our case, we already have a database created for us.

If you were running MySQL via the command line (SSH), to create a database you would type the following command at the mysql> prompt:

Note – Almost all MySQL commands end with a semicolon.

**Create database db1\_gwalke01;**

Your display would then resemble below.



BUT…

We will be using phpMyAdmin instead of the command line which gives us a graphical user interface through which we can manipulate our MySQL database.

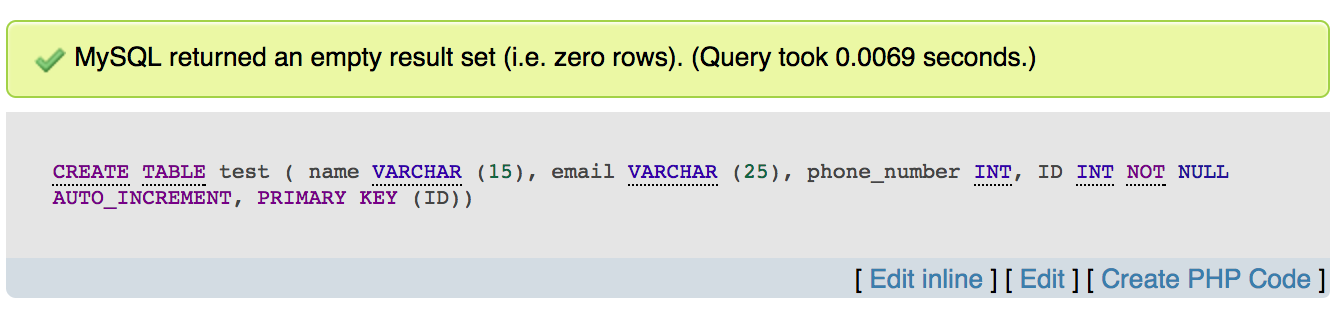
Accessible here: <https://intweb.bucks.ac.uk/sqladmin/>

**TASK 2**

Before we can execute commands on the database, we must first create a table in which data can be stored. Type the following SQL command under the “SQL” tab (after you have clicked on your database name in the left hand column). This will create a table called test with four fields.

CREATE TABLE test (  
name VARCHAR (15),  
email VARCHAR (25),  
phone\_number INT,  
ID INT NOT NULL AUTO\_INCREMENT,  
PRIMARY KEY (ID));

Your display should now look like that below



Graphical user interface, text, application

Description automatically generated

The first table in your database has now been created. *Note: no two tables can have the same name.*

**Column Characteristics:**

* A name may not be made up of strictly numbers.
* A name may start with a number.
* A name may be up to 64 characters.

**Other table options:**

The following options can be placed after any datatype, adding other characteristics and capabilities to them.

* Primary Key. Used to differentiate one record from another. No two records can have the same primary key. This is obviously useful when it is imperative that no two records are mistaken to be the other.
* Auto\_Increment. A column with this function is automatically incremented one value (previous + 1) when an insertion is made into the record. The datatype is automatically incremented when 'NULL' is inserted into the column.
* NOT NULL. Signifies that the column can never be assigned a NULL value.

Examples:

* soc\_sec\_number INT PRIMARY KEY;  
  No two soc\_sec\_number records can hold the same value.
* ID\_NUMBER INT AUTO\_INCREMENT;  
  Automatically increments in value, starting at '1', with every subsequent insertion.

**Table-Relevant Commands**

Via the command line, we would be able to execute a number of useful commands pertaining to the tables, such as the following:

**Show Tables**

mysql> **show tables;**

Result:  
This will list all tables currently existing within the database.

*(this is also achieve by clicking on the DATABASE name on the left hand side of the phpMyAdmin interface)*

**Show Columns**

mysql> **show columns from test;**

Result:  
This will return the columns and column information pertaining to the designated table.

*(this is also achieved by clicking on the TABLE name on the left hand side of the phpMyAdmin interface)*

Take a minute to execute each one of the above commands after you have created the test table. They will prove very helpful as your database increases in size and complexity.

You should now have a basic understanding of the creation of tables, one of the most important concepts of the MySQL server. You now know that tables are constructed using datatypes, which when grouped together form a record.

A database can be manipulated in four possible ways: addition, deletion, modification, and search. These topics will all be briefly covered in the following two sections. However, before we begin, I would like to highlight the fact that SQL, like many computer languages, is somewhat particular about command syntax. The slightest error in placement of a parentheses, comma, or semicolon will almost surely end in error. As a result, take care to be attentive of command syntax.

**Insertion of records**

Insertion of data into the table is accomplished, logically enough, using the INSERT command.

**TASK 3**

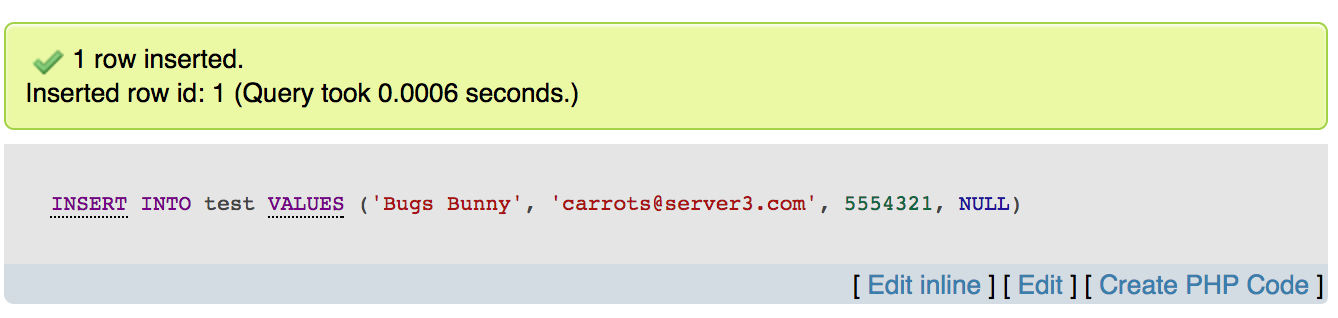
Execute the following SQL command within phpMyAdmin:

INSERT INTO test VALUES

('Bugs Bunny', 'carrots@server3.com',

5554321, NULL);

Result, assuming the command was correctly entered:



Graphical user interface, application

Description automatically generated

So what happened?

* Single quotations were placed around the data of type VARCHAR. All data of type STRING (i.e. char, varchar, text, blob, etc.) must be surrounded in single quotes, or an error will occur.
* There were no single quotes surrounding the phone number. Datatypes of type INT do not require single quotes. (Consider why it might be better to use a CHAR datatype for a phone number?)
* NULL ? A NULL allows any datatype with the characteristic AUTO\_INCREMENT to be automatically assigned a value. If it is the first record inserted into the database, it is assigned the value '1'. Otherwise, it is assigned the previously inserted value + 1 (i.e. if the previously inserted value was '2', then the next would be '3'). In addition, the insertion of NULL into a variable of type TIMESTAMP causes that variable to be given the value of the current date.

**Note:** It is of importance to remember that the same number of values must be inserted as datatypes are contained within a record. In the above example, if one attempted to insert only three values instead of four, the insertion would fail. The same result applies if one attempted to insert five values.

Example:

mysql> insert into test values('doggy');

ERROR 1058: Column count doesn't match value count

mysql>

**Note (2):** One of the advantageous aspects of MySQL is its ability to convert without trouble between datatypes. MySQL automatically converts between integers, strings, and dates without problems.

**Selection**

**TASK 4**

A database would not be much use if one was not able to search and extract data from it. In MySQL terms, this is accomplished through the SELECT statement. Execute the following command.

SELECT \* FROM test

WHERE (name = "Bugs Bunny");

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **Email** | **phone** | **ID** |
| Bugs Bunny | carrots@server3.com | 5554321 | 1 |

Graphical user interface, application

Description automatically generated

**TASK 5**

Add the following records to the test table.

INSERT INTO test VALUES

('Bugs Bunny', 'peppers@server3.com',

5554331, NULL);

INSERT INTO test VALUES

('Bugs Bunny', 'lettuce@server3.com',

5554341, NULL);

INSERT INTO test VALUES

('Bugs Bunny', 'celery@server3.com',

5554351, NULL);

Graphical user interface, text, application, table

Description automatically generated

**TASK 6**

Display all the bugs bunny records by typing the following command

SELECT \* FROM test

WHERE (name = "Bugs Bunny");

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **Email** | **phone** | **ID** |
| Bugs Bunny | carrots@server3.com | 5554321 | 1 |
| Bugs Bunny | peppers@server3.com | 5554331 | 2 |
| Bugs Bunny | lettuce@server3.com | 5554341 | 3 |
| Bugs Bunny | celery@server3.com | 5554351 | 4 |

Graphical user interface, application

Description automatically generated

**Deletion**

One can also delete records inserted into the table. This is accomplished through the DELETE command. Type the following command at the mysql> prompt.

**TASK 7**

DELETE FROM test

WHERE (phone\_number = 5554321);

Result: (Using the previously illustrated example)

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Bugs Bunny | peppers@server3.com | 5554331 | 2 |
| Bugs Bunny | lettuce@server3.com | 5554341 | 3 |
| Bugs Bunny | celery@server3.com | 5554351 | 4 |

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**Modification**

MySQL also has the capability of modifying data already entered into the table. This is accomplished through the UPDATE command.

**TASK 8**

UPDATE test SET name = 'Daffy Duck'

WHERE name = "Bugs Bunny";

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Daffy Duck | peppers@server3.com | 5554331 | 2 |
| Daffy Duck | lettuce@server3.com | 5554341 | 3 |
| Daffy Duck | celery@server3.com | 5554351 | 4 |

Graphical user interface, text, application

Description automatically generated

**SUMMARY**

In this session we have covered the core MySQL database manipulation functions, basic insertion, deletion, modification, and search.

**FURTHER READING**

A database may be considered as a complex structure but it is only made from a number of simple data structures. In MySQL, the structure is based upon **tables** where each row in the table forms a **record.** Each record will hold information about a single person or item and will have the same number of elements, known as **fields**, as all other **records** in the **table**.

Each field is based upon the one format or structure known as a **datatype**. Together, one or more of these datatypes form the record. A table holds the collection of records that make up part of the database. We can consider the hierarchy of a database to be that of the following:

Database < Table < Record < Datatype

Datatypes come in several forms and sizes, allowing the programmer to create tables suited for the scope of the project. The decisions made in choosing proper datatypes greatly influence the performance of a database, so it is wise to have a detailed understanding of these concepts.

**MySQL Datatypes**

MySQL is capable of many of the datatypes that even the novice programmer has probably already been exposed to. Some of the more commonly used include:

**CHAR (M)**  
CHAR's are used to represent fixed length strings. A CHAR string can range from 1-255 characters. In later table creation, an example CHAR datatype would be declared as follows:

ex.  
car\_model CHAR(10);

**VARCHAR (M)**  
VARCHAR is a more flexible form of the CHAR data type. It also represents data of type String, yet stores this data in variable length format. Again, VARCHAR can hold 1-255 characters. VARCHAR is usually a wiser choice than CHAR, due to it's variable length format characteristic. Although, keep in mind that CHAR is much faster than VARCHAR, sometimes up to 50%.  
(A CHAR stores the whole length of the declared variable, regardless of the size of the data contained within, whereas a VARCHAR only stores the length of the data, thus reducing size of the database file.)

ex.  
car\_model VARCHAR(10);

**INT (M) [Unsigned]**  
The INT datatype stores integers ranging from -2147483648 to 2147483647. An optional "unsigned" can be denoted with the declaration, modifying the range to be 0 to 4294967295

ex.  
light\_years INT;  
Valid integer: '-24567'.    Invalid integer: '3000000000'.

ex.  
light\_years INT unsigned;  
Valid integer: '3000000000'.    Invalid integer: '-24567'.

**FLOAT [(M,D)]**  
A FLOAT represents small decimal numbers, used when a somewhat more precise representation of a number is required.

ex.  
rainfall FLOAT (4,2);  
This could be used to represent rainfall average in centimeters per year, which could be a decimal value. More specifically, FLOAT (4,2) states the fact that rainfall can hold up to four characters and two decimal places. Thus,

42.35 is valid, accurately represented.  
324.45 is invalid, rounded to 324.5.  
2.2 is valid, accurately represented.  
34.542 is invalid, rounded to 34.54.

*Note: Due to the fact that FLOAT is rounded, those wishing to represent money values would find it wise to use* ***DECIMAL****, a datatype found within MySQL that does not round values. Consult the documentation for a complete explanation.*

**DATE**   
Stores date related information. The default format is 'YYYY-MM-DD', and ranges from '0000-00-00' to '9999-12-31'. MySQL provides a powerful set of date formatting and manipulation commands, too numerous to be covered within this article. However, one can find these functions covered in detail within the MySQL documentation.

the\_date DATE;

**TEXT / BLOB**  
The text and blob datatypes are used when a string of 255 - 65535 characters is required to be stored. This is useful when one would need to store an article such as the one you are reading. However, there is no end space truncation as with VARCHAR AND CHAR. The only difference between BLOB and TEXT is that TEXT is compared case insensitively, while BLOB is compared case sensitively.

**SET**  
A datatype of type string that allows one to choose from a designated set of values, be it one value or several values. One can designate up to 64 values.

ex.  
transport SET ("truck", "wagon") NOT NULL;

From the above declaration, the following values can be held by transport:

""  
"truck"  
"wagon"  
"truck,wagon"

**ENUM**  
A datatype of type string that has the same characteristics as the SET datatype, but only one set of allowed values may be chosen. Usually only takes up one byte of space, thus saving time and space within a table.

ex.  
transport ENUM ("truck", "wagon") NOT NULL;

From the above declaration, the following values can be held by transport:

""  
"truck"  
"wagon"

**Records**

Together, a group of declared datatypes form what is known as a record. A record can be as small as one data variable, or as many as deemed needed. One or more records form the structure of a table.