MUHABURA INTEGRATED POLYTECHNIC COLLEGE
Website: www.mipc.ac.rw
E-mail: info@mipc.ac.rw
P.O.Box: 26 Musanze – Rwanda
Tel: +250788687033
"Accredited by WDA Resolutions of the BoD 1094/17 on 29 June 2017"

MUHABURA INTEGRATED POLYTECHNIC COLLEGE (MIPC)

SECTOR: ICT

SUB-SECTOR: SOFTWARE DEVELOPMENT

REQF Level:4

MODULE TITLE: WEBSITE AND DATABASE INTEGRATION

Competence: Integrate website with database

Module code: SFDWD401

Credits:6

Learning hours:60

By: IMANISHIMWE Aline

ACADEMIC YEAR 2021-2022

TABLE OF CONTENTS

LEARNING UNIT 1: CONNECT TO THE DATABASE

Learning Outcome 1.1: Review the importance and strategies of website and database integration

Website and database integration (or data integration) refers to combining databases with your website so that web visitors can add, remove and update information in your database using a web browser.

1.1.1 Importance to integrate a website with a database (importance of website and database integration)

- ***Website and database integration,*** enables you to manage enterprise-wide data or information from a centralized location/storage/database. (it allows high performance in business).
- With website and database integration, it is easier to identify bottlenecks(slowness of data flow in the database).
- It helps to improve user experience and to reduce delivery time.
- It helps you to manage online data. (it provides the easy way to manage website online)
- It allows you to keep and handle large amounts of data
- It is faster to add new features like links and it is easier to maintain those features(=>it is easier to update the website).

- Web based databases help you to save money because it is cheaper depending on the number of computers you use and you don't need to invest in servers to store the data at your business.
- Using an online database program allows your business to be flexible(flexible use) and you only pay for the amount of storage that you use.

- It helps the developers to create dynamic websites/database websites/ database-driven site.

NB: There are 2 types of website (static website and dynamic website):

- **Static web pages** are pages that remain the same when you view their URL (unless they were edited by the page creator).
  - Everything we created so far via **HTML ,css , javascript** were static pages.
- **Dynamic web pages** are pages where some or all of the content is dependent on some conditions or user interaction.

Example: Google's search results are dynamic, as an example, since their content depends on the user search query and on the current state of the Google database(s).

Dynamic web pages are created using:

- **Client-side scripts** embedded in an HTML page for processing on the client (your computer)
- **Server-side programs** that are processed on a server computer
- A mix of **client-side scripts** and **server-side processing** make up dynamic web pages.
- **Database**: is a collection of related tables that contain data

Advantages

- *Less hassle to update/ it is not difficult to update a dynamic website/*It is easier and faster to update content.
- *More functionality and high scalability*
- *Dynamic (changing) content draws visitors back to site*
- *Reduce chances for errors*
- *Dynamic content helps in search engines*

Disadvantages

- *More expensive to host*
- *Relatively slow performance in comparison with a static site*

1. The alternative to a **database website** is called a "static **website**." This is a **website** whose pages permanently exist as HTML.

A **static website** is the website that contains web pages with fixed content. Each page is coded in HTML and displays the same information to every user. **Examples** of **static** web page include about us page with a corporate **website**, mission, vision etc.

Advantages

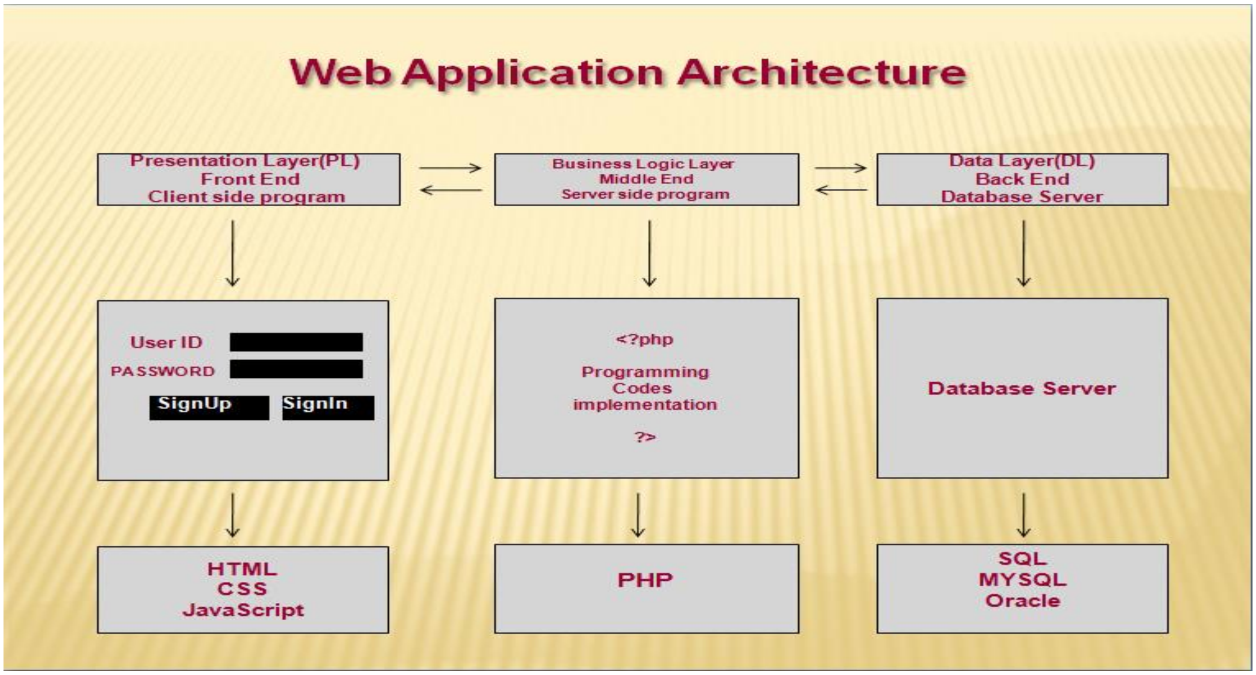
- *Quick and easy to develop*
- *Cheap to host and develop*

Disadvantages

- *Content can become stagnant/static/inactive/slow in accessibility*
- *Requires web development expertise to update*

3-tiers architecture as used in website and database integration

3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. A “tier” in this case can also be referred to as a “layer”.



Presentation layer

Presentation/client Tier(client software/ web browser): The presentation tier is the front end layer in the 3-tier system and consists of the user interface and do some data processing. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user.

Logic layer

The “business“logic/server tier plays the role of transferring information between the website and the data tier, including integration of the required decision logic or transformation of transferred data (calculations, aggregation of information from more data sources and the like). This might be written in C#, Java, C++, Python, Ruby, php, JavaScript etc.

Server tier (application server): Manages client connections and data processing.

Data access layer

The data storage/ database tier (database server) implements persistent data storage, with a relational database (RDBMS) or another type of database (NoSQL). A data access layer (DAL) in computer software, is a layer of a computer program which provides simplified access to data stored in persistent storage of some kind, such as an entityrelational database.

This tier is for data storage and low-level data manipulation

It’s also responsible for managing updates, allowing simultaneous (concurrent) access from web servers, providing security, ensuring the integrity of data, and providing support services such as data backup. Importantly, a good database tier must allow quick and flexible access to millions upon millions of facts. This could be MSSQL, MySQL, Oracle, or PostgreSQL, Mongo, etc.

Strategies to integrate website with database

what to be considered while connecting website and database?

For connecting website with database, you will need:

- Protocols
- Browser requests
- Web Server processes
- Web server responses

Protocols

set of rules that govern data communication over network.

There are various types of protocols that support a major and compassionate role in communicating with different devices across the network. These are:

1. Transmission Control Protocol (TCP)
2. Internet Protocol (IP)

3. User Datagram Protocol (UDP)
4. Post office Protocol (POP)
5. Simple mail transport Protocol (SMTP)
6. File Transfer Protocol (FTP)
7. Hyper Text Transfer Protocol (HTTP)
8. Hyper Text Transfer Protocol Secure (HTTPS)

1. **Transmission Control Protocol (TCP)**: TCP is a popular communication protocol which is used for communicating over a network. It divides any message into series of packets that are sent
from source to destination and there it gets reassembled at the destination.

2. Internet Protocol (IP): IP is designed explicitly as addressing protocol. It is mostly used with
TCP. The IP addresses in packets help in routing them through different nodes in a network until it reaches the destination system. TCP/IP is the most popular protocol connecting the networks.

3. User Datagram Protocol (UDP): UDP is a substitute communication protocol to Transmission
Control Protocol implemented primarily for creating loss-tolerating and low-latency linking between different applications.

4. Post office Protocol (POP): POP3 is designed for receiving incoming E-mails.

5. Simple mail transport Protocol (SMTP): SMTP is designed to send and distribute outgoing E-Mail.

6. File Transfer Protocol (FTP): FTP allows users to transfer files from one machine to another.
Types of files may include program files, multimedia files, text files, and documents, etc.

7. Hyper Text Transfer Protocol (HTTP): HTTP is designed for transferring a hypertext among two or more systems. HTML tags are used for creating links. These links may be in any form like
text or images. HTTP is designed on Client-server principles which allow a client system for establishing a connection with the server machine for making a request. The server acknowledges the request initiated by the client and responds accordingly.

8. Hyper Text Transfer Protocol Secure (HTTPS): HTTPS is abbreviated as Hyper Text Transfer
Protocol Secure is a standard protocol to secure the communication among two computers one using the browser and other fetching data from web server. HTTP is used for transferring data between the client browser (request) and the web server (response) in the hypertext format, same in case of HTTPS except that the transferring of data is done in an encrypted format. So, it can be said that https thwart hackers from interpretation or modification of data throughout the transfer of packets.

Browser requests/HTTP Request

The browser sends an HTTP request message to the server, asking it to send a copy of the website to the client. This message, and all other data sent between the client and the server, is sent across your internet connection using TCP/IP.

your browser sends a request for a specific file, often an HTML file.

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers. HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server.

- **HTTP request Contains:**
    - the method to be applied to the data resource
    - the identifier of the resource
    - the protocol version in use
- **Most commonly used methods:**
    - GET – Fetch/ receive a document
    - HEAD - Fetch just the header of the document
    - POST - Execute the document, using the data in body/ used to send form data
    - PUT - Store a new document on the server
    - DELETE - Remove a document from the server

When to use GET?

- Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL).
- GET also has limits on the amount of information to send.
- The limitation is about 2000 characters.
- However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- **Note:** GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

- Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and
- has **no limits** on the amount of information to send.
- Moreover, POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
- **Developers prefer POST for sending form data.**

Web Server processes

In general, a web server can contain one or more websites. A web server processes incoming network requests over HTTP and several other related protocols.

Web server responses / HTTP Response

HTTP response is a message generated by a server after receiving and interpreting the HTTP request.

- **Responses contain:**

Status line with the protocol version, a status code, and a “reason phrase”

- Response-Header (containing information about the server)
- Entity Header (meta-information)
- Entity Body (data)

Learning Outcome1.2: Select connection tools and platforms according to the application specifications.

Software specifications/acquirement include:

- **DBMSs to be used**. Eg: MySQL which is RDBMS
- **Editors**: text editors are used to write and edit source codes. Eg: Notepad, notepad++, Dreamweaver, sublime etc
- **Browsers**: are used to access and display/view the contents from the server (it used to access and to view websites). Eg: Common web browsers include Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, and Apple Safari.
- **Web servers** typically host websites that are accessible on the **Internet** and they respond to all requests from the web browsers.

Examples of web servers include: IIS (Internet Information Server), Apache HTTP Server etc**.**

Web server architecture:

- LAMP: is fully open source, Linux for operating system, Apache for web server, MySQL for database and PHP for server-side scripting
- WAMP: Uses Windows for operating system, with Apache, MySQL, and PHP
- XAMPP (X-platform Apache MySQL PHP and Perl)
- WISA: Full Microsoft package, Windows, Internet Information Server (IIS), SQL Server (enterprise) or Access (small-scale) and ASP or ASP.NET

Identification of connection tools and platforms

- server-side script language (JavaScript-, PHP, SQL)
- Middleware (…….?????)
- Client-side scripting languages(HTML, CSS)
- Platforms or OS(Operating System) include: windows, linux, unix, mac os etc

HTML: is used to define the content of web pages.
CSS: is used to specify the layout of web pages/ the language for styling web pages
JavaScript: is used to program the behaviour of web pages/the language for programming web pages/ It is a language for creating dynamic contents
SQL: a language for accessing databases
Python and Java: are the programming languages
PHP: A web server programming language for creating dynamic website.
JQuery: A JavaScript library for developing web pages

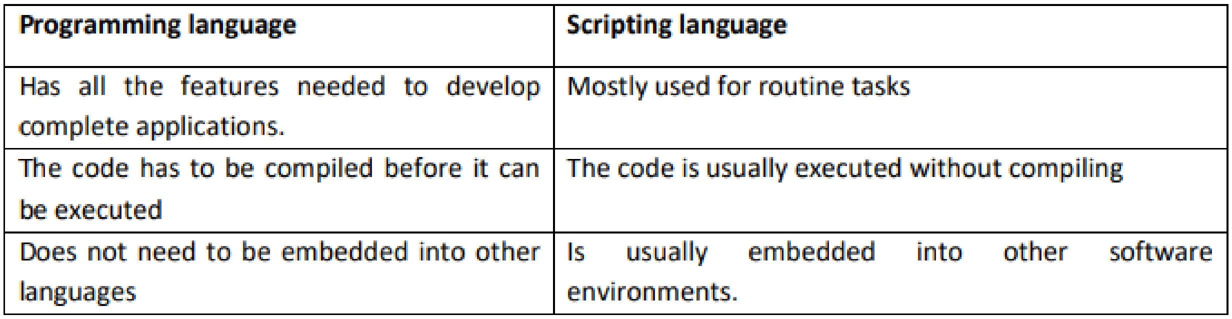
What is a Scripting Language?

A script is a set of programming instructions that is interpreted at runtime.
A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments. The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.

Server-side scripts are interpreted on the server while client side scripts are interpreted by the
client application.

PHP is a server-side script that is interpreted on the server while
JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be
embedded into HTML pages.

Programming Language Vs Scripting Language



Programming language	Scripting language
Has all the features needed to develop complete applications.	Mostly used for routine tasks
The code has to be compiled before it can be executed	The code is usually executed without compiling
Does not need to be embedded into other languages	Is usually embedded into other software environments.

Learning Outcome 1.3: Connect database to a web application using server-side script language (PHP).

1.3.1 Introduction to PHP

PHP is an open source, server-side, scripting language used for the development of web applications. By scripting language, we mean a program that is script-based (lines of code) written for the automation of tasks.

The full of **PHP** is **PHP Hypertext Preprocessor**. The P in the PHP being the acronym that is used to identify the whole thing.

The PHP Hypertext Preprocessor (PHP) is a scripting language that allows web developers to create **dynamic** contents that interact with databases. PHP is basically used for developing web-based software applications.

It is called the Preprocessor because the PHP codes are processed before sending them to the browser.

PHP offers a web developer more freedom and choices with higher level of detail.

It is open source and free which means that users can download from any kind of PHP downloading websites with the required licenses or permits and get going. Moreover, no kind of renewal charges are taken after downloading the PHP application.

<u>Basic PHP Syntax</u>

- A PHP scripting block always starts with **<?php** and ends with **?>**.

```
<?php                              ?>
```

- It can be placed anywhere in the document such as html file.

<u>PHP FILES</u>

A **file** with the <u>**.PHP**file extension</u> is a **PHP** Source Code **file** that contains Hypertext Preprocessor code. They are often used as web page **files** that usually generate HTML from a **PHP** engine running on a web server. The HTML content that the **PHP** engine creates from the code is what's seen in the web browser.

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have ".php" as extension, eg: Index.php

Here is how to create a PHP file by using Notepad:

1. Open Notepad or any other interpreter
2. Enter your PHP program into Notepad.
3. Choose Save As from the File menu.
4. Enter the file name as **your_file.php** being sure to include the **.php** extension.
5. Set the Save As Type to All Files.
6. Finally, click the Save button.

Embedding PHP in XHTML (Combine PHP with XHTML)

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>

</html>
```

- Each code must end with a semicolon (;) with php language.

- There are two basic statements to output text with PHP: **echo** and **print**.

PHP Case Sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

In the example below, all three echo statements below are legal (and equal):

```
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

- But; all variable names are case-sensitive in PHP language.

In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

```
<?php
```

```
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP variables declaration

- A variable is used to store information.
- **Variables** are used to store data/information like string of text, numbers, arrays etc.
- **PHP** automatically converts the **variable** to the correct data type, depending on its value. After declaring a **variable** it can be reused throughout the code. The assignment operator (=) used to assign value to a **variable**.
- Once a variable is declared, it can be used over and over again in the script.
- All variables in PHP start with a $ sign symbol.
- The correct way of declaring a variable in PHP:

Example: $var_address = value;

- A variable can contain a string, and a variable containing a number:

  -
    ```
    <?php
    $s="Hello World!";
    $y=16;
    ?>
    ```

Note that: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

When you assign a text value to a variable, put quotes around the value.

PHP is a Loosely Typed Language means that:

- No need to declare a variable before adding a value to it.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In PHP, the variable is declared automatically when you use it.

Naming Rules for Variables:

  - A variable name must start with a letter or an underscore “_”

- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name cannot start with a number
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_var), or with capitalization ($myVar)
- A string variable is used to store and manipulate text.

String Variables in PHP:

- String variables are used for values that contain characters.
- A string can be used directly in a function or it can be stored in a variable.

Example:

```
<?php
$s="Hello World";
echo $s;
?>
```

- **The Concatenation Operator** is the **only string operator** in PHP.
  - The concatenation operator (.) is used to put two string values together.
  - To concatenate two string variables together, use the concatenation operator:

Example:

```
<?php
$s1="Hello World!";
$s2="What a nice day!";
echo $s . " " . $s;
?>
```

PHP Forms and User Input

- If you know a little HTML, then you know that the FORM tags can be used to interact with your users. Things that can be added to a form are the likes of text boxes, radio buttons, and check boxes, drop down lists, text areas, and submit buttons. A basic HTML form with a textbox and a Submit button looks like this:
- **<!DOCTYPE html>**
- **<html>**
- **<head>**

- **<title>A basic HTML form</title>**
- **</head>**
- **<body>**
- **<FORM NAME ="form1" METHOD ="POST" ACTION = "">**
- **<table border=0 >**
- **<tr><td>Firstname:</td><td><INPUT TYPE = "TEXT" placeholder ="Fname" name="Fname"></td></tr>**
- **<tr><td>LastName:</td><td><INPUT TYPE = "TEXT" placeholder ="Lname" name="Lname"></td></tr>**
- **<tr><td><INPUT TYPE = "Radio" name="gender">Male</td><td><INPUT TYPE = "Radio" name="gender">Female</tr>**
- **<tr align="center"><td><INPUT TYPE = "Submit" Name = "Submit1" value = "submit"></td></tr>**
- **</table>**
- **</FORM>body>**
- **</html>**

we will discuss the **METHOD**, **ACTION, Name** and **SUBMIT** attributes in the form above, because they are important.

The Method Attribute of HTML Forms

The **Method** attribute is used to tell the browser how the form information should be sent. The two most popular methods you can use are GET and POST. However, our METHOD is blank. So change it to this:

- <FORM NAME ="form1" **METHOD ="GET"** ACTION = "">
- To see what effect using GET has, save your work again and then click the Submit button on your form. You should see this:
- The thing to notice here is the address bar. After **basicForm.php**, we have the following:

? Submit1=Login. This is a consequence of using the GET method. The data from the form ends up in the address bar

.

The POST Attribute of HTML Forms

- Using POST means that the form data will not be appended to the address in the address bar for all to see. if the data is not sensitive then use GET, otherwise use POST.

The ACTION Attribute of HTML Forms

- The Action attribute is crucial. It means, "Where do you want the form sent?" If you miss it out, your form will not be sent anywhere.

Getting values from a Text Box with PHP

We are going to use these to process text that a user has entered into a text box. The METHOD attribute tells you **how** form data is being sent, and the ACTION attribute tells you **where** it is being sent.

To get at the text that a user entered into a text box, the text box needs a NAME attribute. You then tell PHP the NAME of the textbox you want to work with.

The Submit Button of a HTML FORM

- The HTML Submit button is used to submit form data to the script mentioned in the ACTION attribute.
- <Form Name ="form1" Method ="POST" **ACTION = "basicForm.php"**>
- Therefore, the page mentioned in the ACTION attribute is basicForm.php. To submit this script, you just need a HTML Submit button:
- **<INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">**

The Submit Button of a HTML FORM

- The HTML Submit button is used to submit form data to the script mentioned in the ACTION attribute.
- <Form Name ="form1" Method ="POST" **ACTION = "basicForm.php"**>
- Therefore, the page mentioned in the ACTION attribute is basicForm.php. To submit this script, you just need a HTML Submit button:
- **<INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">**

- ➢ The isset () function is used to **check whether a variable is set or not**.
- ➢ $_POST is a variable used to **grab/take data sent through a web form**. ($_POST is used to retrieve/bring/get values passed to your page via a POST request.

1.3.2 Make a database structure

Introduction to database (PHP MySQL Database - review/revision)

With PHP, you can connect to and manipulate databases.

What is MySQL and why it is used?

MySQL is an open-source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). (MySQL is the most popular database system used with PHP).

- MySQL is a RDBMS used on the web/ internet.
- MySQL is a RDBMS that runs on a server
- MySQL is used for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL as a programming language
- MySQL compiles/ can be used on a number of platforms or different OSs.
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: called **My.**

MySQL is most used because of: its quick processing, its proven reliability, its ease and flexibility of use.

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

1. **What is SQL?**

SQL is the most popular language for creating, adding, accessing and managing content in a database.

SQL is generally a standard language used in the development of relational databases. (It is a **programming language** designed for querying data from a database).

What are the two major components of standard SQL and what function do they serve?

a) **A Data Definition Language (DDL)** for defining the database structure and controlling access to the data.

b) **A Data Manipulation Language (DML)** for retrieving and updating data.

What is the difference between a database and DBMS (Database Management System)?

- A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. (4 types of database: text database, hierarchical database, relational database and network database).

- A **database management system** (**DBMS**) is **system** software for creating and **managing databases**. The **DBMS** provides users and programmers with a systematic way to create, retrieve, update and manage **data**. A **DBMS** makes it possible for end users to create, read, update and delete **data** in a **database**.

Database Queries

A query is a question or a request or a command.

A **query** is also defined as a request for data or information from a **database** table or combination of tables. (What is a query in database with an example?)

We can query/ request a database for specific information and have a record returned.

Look at the following query (using standard SQL):

- ⇨ SELECT LastName FROM Employee

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security

Different key terms and functions used in database connection

a. **We use mysqli_connect()** or mysql_connect() to connect to the mysql database

Eg: $conn = mysqli_connect($servername, $username, $password);

b. **Servername:** it is optional and specifies the server to connect to. The default value is "**localhost:3306**"

c. **Username:** it is optional and specifies the username to login with to the server. The default value is the name of the user/server that owns the server process. "**root**".

d. **Password:** it is optional and specifies the password to login with to the server and the default value is empty " ".

e. **mysqli_connect_error()**or **mysql_error()**function**, connect_error( )** functionand **PDO::ERRMODE_SILENT()** function is used to identify the exact errors that have happened.

f. The connection is closed automatically when the script ends. But, it is possible to close the connection before by using**close(), mysqli_close() / mysql_close()** and **null**.

NB:Application of connection object and method

Object= use of classes
Method/procedural= use of functions

PHP 5 and later can work with a ***MySQL*** database by using:

- MySQLi extension (the "i" stands for improved) = MySQL improved extension
- PDO (***PHP*** Data ***Objects***)

Earlier versions of PHP used the MySQL extension.

Application database access methods

1. Mysqli_query

The **mysqli_query( )**function is used to execute some queries.
It is used to simplify the act of performing a query against the database represented by the link parameter.

2. Mysqli_select

The **SQL SELECT command** is used to fetch/collect/get data from MySQL database.
The **mysqli_select_db( ) function** is used to change the default database for the connection (it sets the active MySQL database, and it returns **true** on success or **false** on failure).

3. Mysqli_num_rows

The **mysqli_num_rows( )** function is an inbuilt function in php which is used to return the number of rows present in the result set.
It is generally used to check if data is present in the database or not.
To use this function, it is mandatory to first set up the connection with the mysql database.

4. Mysqli_fetch_array

The **mysqli_fetch_array**() **function** fetches/retrieves a result row as an associative array, a numeric array, or both.
It returnsan array that corresponds to the fetched row or FALSE if there are no more rows for the database connection represented by the link parameter.

mysqli_fetch_array() is an extended version of the mysqli_fetch_row() function.

5. Mysqli_close
Mysqli_close( ) function is used to terminate or close the connection to database.
We use single line comment (// or #)and multiple line comment(/* */) in php to add comments in php scripts.

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

EXERCISE 1

```
<?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";

// create database

/* $a="create database levels";

if (mysqli_query($conn, $a)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}*/

mysqli_select_db($conn,"levels"); // for selecting database where to create tables

$b="create table level5(number int(6) auto_increment primary key, names varchar(50), age
int(10))";

//To execute the query

//mysqli_query($b,$conn);  this gives error because it does not follow the syntax

if (mysqli_query($conn, $b)) {
    echo "Table level5 created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);    // is used to close the connection from the database.
?>
```

L.U.2. IMPLEMENT CRUD OPERATIONS

PHP CREATE Database

1. **Using Object-oriented methods**

Create a MySQL Database Using MySQLi

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Note: When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password).

Tip: If you have to use a specific port, add an empty string for the database-name argument, like this: new mysqli("localhost", "username", "password", "", port)

PHP CREATE TABLES

Create a MySQL Table Using MySQLi

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```sql
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

using Object oriented method

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
```

L.O.1.1. Insert data into database using structured query language (SQL) in reference to standard queries

Insert Data Into MySQL Using MySQLi

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

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

In the previous chapter we created an empty table named "MyGuests" with five columns: "id", "firstname", "lastname", "email" and "reg_date". Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP with default update of current_timesamp (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

Example (MySQLi Procedural)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

- ✓ Preparing a query string
    - Specify values

Example

```
<!DOCTYPE html>
<html>
  <body>
		<form method="post" action="process.php">
			First name:<br>
			<input type="text" name="first_name">
```

```
                    <br>
                    Last name:<br>
                    <input type="text" name="last_name">
                    <br>
                    City name:<br>
                    <input type="text" name="city_name">
                    <br>
                    Email Id:<br>
                    <input type="email" name="email">
                    <br><br>
                    <input type="submit" name="save" value="submit">
          </form>
  </body>
</html>
<?php
$conn= mysqli_connect("localhost","root","","dbname") or die("unable to connect")
if(isset($_POST['save']))
{
          $first_name = $_POST['first_name'];
          $last_name = $_POST['last_name'];
          $city_name = $_POST['city_name'];
          $email = $_POST['email'];
          $sql = "INSERT INTO employee (first_name,last_name,city_name,email)
          VALUES ('$first_name','$last_name','$city_name','$email')";
          if (mysqli_query($conn, $sql)) {
                    echo "New record created successfully !";
          } else {
                    echo "Error: " . $sql . "
" . mysqli_error($conn);
          }
          mysqli_close($conn);
}
?>
```

- ✓ Executing a query
  - Check if records are inserted

L.O.1.2. Retrieve and display data from database in the most appropriate control according to the information requirements

- ✓ Preparing a query string
  - Adding conditions
  - Using different SQL Clauses
  - Using functions and operators
- ✓ Executing a query
  - Check if records are found
- ✓ Creating an array of results

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:
SELECT column_name(s) FROM table_name
or we can use the * character to select ALL columns from a table:
SELECT * FROM table_name

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Procedural)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
<?php

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();

?>
```

Code lines to explain from the example above:

First, we set up an SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

L.O.1.3. Update data with user-supplied input according to the information changes

Update Data In a MySQL Table Using MySQLi

The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . $conn->error;
}

$conn->close();

?>
```

- ✓ Preparing a query string
  - Adding conditions
  - Using different SQL Clauses
  - Using functions and operators
- ✓ Executing a query
  - Check if records are updated

L.O.1.4. Delete data from database according to the information requirements

Delete Data From a MySQL Table Using MySQLi

The DELETE statement is used to delete records from a table:

DELETE FROM table_name
WHERE some_column = some_value

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (MySQLi Object-oriented)

```
 <?php
 $servername = "localhost";
 $username = "username";
 $password = "password";
 $dbname = "myDB";

 // Create connection
 $conn = new mysqli($servername, $username, $password, $dbname);
 // Check connection
 if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
 }

 // sql to delete a record
 $sql = "DELETE FROM MyGuests WHERE id=3";

 if ($conn->query($sql) === TRUE) {
   echo "Record deleted successfully";
 } else {
   echo "Error deleting record: " . $conn->error;
 }

 $conn->close();

?>
```

- ✓ Preparing a query string
  - Adding conditions
  - Using different SQL Clauses
  - Using functions and operators
- ✓ Executing a query
  - Check if records are deleted

L.U.3. CREATE REPORTS TO PRESENT SUMMARY INFORMATION

L.O.3.1. Display general data from database according to the information requirements

Loops in Arrays

- While loop
- Foreach loop
- Memory release

✓ Data presentation in HTML tables

Pagination and paragraphing

```
<!DOCTYPE html>
<html>
<head>
	<title>Pagination</title>
	<link rel="stylesheet" type="text/css" href="bootstrap-4.0.0-
dist/css/bootstrap.css"></style>

</head>
<body>
<?php
$conn= new mysqli ("localhost", "root","","admin") or die("unable to connect "
.$conn->connect_error());

$limit = 10;
if (isset($_GET["page"])) {
	$page  = $_GET["page"];
	}
	else{
	$page=1;
	};
$start_from = ($page-1) * $limit;
$select = "SELECT * from student ORDER BY ID ASC LIMIT $start_from, $limit
";
$result = mysqli_query($conn,$select);
?>
<table><th>ID</th><th>First Name</th><th>Last Name</th><th>Roll
No</th><th>City</th>
<?php
while ($row = mysqli_fetch_array($result)) {
	?>

	<tr>
		<td><?php echo $row['ID']?></td>
		<td><?php echo $row['FirstName']?></td>
		<td><?php echo $row['LastName']?></td>
		<td><?php echo $row['RollNo']?></td>
		<td><?php echo $row['City']?></td>
	</tr>
```

```
        <?php
}
?>
</table>
<?php

$result_db = mysqli_query($conn,"SELECT COUNT(ID) FROM student");
$row_db = mysqli_fetch_row($result_db);
$total_records = $row_db[0];
$total_pages = ceil($total_records / $limit);
/* echo  $total_pages; */
$pagLink = "<ul class='pagination'>";
for ($i=1; $i<=$total_pages; $i++) {
         $pagLink .= "<li class='page-item'><a class='page-link'
href='pagination2.php?page=".$i."'>".$i."</a></li>";
}
echo $pagLink . "</ul>";
?>
</body>
</html>
```

- Data in form elements

L.O.3.2. Create specific report based on user-supplied input data

- User input data handling
- Input processing using form methods
  - GET method
  - POST method
- Information format and display

L.O.3.3. Prepare customized and periodic according to information requirements

- Custom parameters
  - Start parameter
  - End parameter
- Methods for sending Data
  - GET method
  - POST method
- Form attributes
  - Action
  - Method
  - Enctype
- Report format according to the application specifications

EXERCISES

1. What is Web database integration?

2. What is a webserver?

3. In order to integrate database with website we need some Software specifications. What are those software? Give 2 examples for each.

4. List five functions used in mysql and their use.

5. What stand for CRUD in term of web development?

6. One way to utilize a large quantity of data is to integrate your website with a database. List 5 Database Web Applications.

7. What is a comment in programming? Show how a comment can be introduced in scripting code

8. Give three image file formats and two sound file formats.

9. Respond True or False

a) The ALTER table statement is often used to modify an existing table's structure.

b) The DROP TABLE command is equivalent to the DELETE FROM <table name> command.

c) The delete command can be used to delete column(s).

d) Drop command can be used to delete row(s).

e) Insert command can be used when you are changing data.

10. Respond with True or False for the follow while creating a Variable in PHP.

a) A variable starts with the $ sign, followed by the name of the variable

b) A variable name must begin with a letter or the underscore character

c) A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

d) A variable name should contain spaces

e) Variable names are not case sensitive ($y and $Y are same variables)

11. (a) Domain name **localhost** is mapped to which IP address?

(b) What is the default **port number** of a web server?

12. What are the two main attributes of **HTML's form** element and what are their functions?

13. Explain the function of the following concept.

a) FTP

b) SMTP

c) DNS

d) POP3

e) HTTP

14. While developing website and database integration, how can you:

a) Open and close “PHP”?

b) Open and close “HTML”?

c) Open and close “Form” in PHP?

d) Connect to database using PHP?

e) Create button “Send” using HTML?

15. Explain four conditional statements (tests) used in programming. Give the syntax for each.

16. Create database in PHP named STUDENTDB, and insert into your database the following table called address where the StudentRegno it Auto number:

StudentRegno	Firstname	Lastname	Location	Sex	phone
1	Faustin	HABUMUGABE	GISAGARA	M	0788344543
2	Francois	MUGISHA	GASAKA	M	0722345365
3	Francine	UWERA	CYANIKA	F	0757565879

17.(a) Give and explain any eight (8) HTML tags

REFERENCES

1. Learn PHP, MYSQL, JavaScript-with jQuery, CSS & HTML5, 4 edition, Robin Nixon

2. HTML, XHTML, & CSS ALL-IN-ONE for dummies, 2nd edition, Andy Harris

3. How to Do Everything with PHP and MySQL (McGraw-Hill, 2005, 0-07-146654-1), Vikran

Wasani

4. https://www.formget.com/update-data-in-database-using-php/

5. https://www.tutorialspoint.com/php/php_validation_example.htm

6. https://www.w3schools.com/php/php_mysql_prepared_statements.asp