APUNTES DE CURSO DE PHP CON MySQL Y P.O.O.

Para poder realizar este curso y ejersitarlo es necesario en primer lugar que tenga instalado en su computadora el lenguaje PHP , el servidor Apache , el administrador de bases de datos phpmyadmin, la base dedatos MySQL y un editor de código textual como podria ser el Notepath , SublimText o Atom .
Para esto puede ser muy practico instalar el paquete WAMP si usa sistema operativo Windows o XAMP si usa sistema operativo Linux , dado que dichos paquetes contienen la instalación de todo el softword necesario ,con excepcion del editor de código.

El código de PHP , requiere en primer lugar ,para ser reconocido como tal , que el archivo sea nombrado con la extensión .php y en 2º lugar dentro del archivo el código PHP inicia con la etiqueta <?php y termina con ?>

Argumentos constantes:

No cambian a lo largo del programa pero se puede acceder a ellos o consultarlos ya sea desde el ámbito local como desde el ámbito global. Se definen a través de la instrucción define(nombre, valor, false/true) el tercer argumento por omisión es false y se puede obviar a menos que se quiera cambiar su estado.
Otro asunto muy importante es que las constantes no llevan el signo $, el nombre de una constante solo puede contener letras mayúsculas o minúsculas y guion bajo pero ningún otro signo y ademas si se usan letras mayúsculas al referirse a ellas con minúsculas se interpretara como otra palabra. Y al tratar de utilizarla dará error por intentar usar una variable que no esta definida pero si cambiamos el estado del 3º argumento que por omisión es false y lo ponemos en true lo interpretará tanto en minúscula como en mayúscula.

Comentarios:

Los comentarios de una sola linea en PHP se marcan con un # a la izquierda y los comentarios de varias lineas se inician con /* y se terminan con */.

Diferencias entre simples comillas y dobles comillas en PHP:

Si definimos una variable y luego intentamos imprimirla con un echo entre comillas simples , solo imprimirá el nombre de la variable como un texto .
Pero si en cambio hacemos un echo con el nombre de la variable entre comillas dobles hacemos lo que se llama interpolación de variable , e incluso si dentro de las comillas la juntamos con otro texto cualquiera , no variable , lo que hará es concatenar el texto con el contenido de la variable e imprimirlo como un solo string.

Tipos de variables:

Hay 8 tipos de variables en PHP pero las mas comunes son 5
Los nombres de variables se identifican antemponiendo un signo $ y no es necesario definir que tipo es una variable antes de usarla , el tipo se asigna dinámicmente según lo que se introduzca en ella.
Los 5 tipos de variables mas comunes son:

_ I**nteger o Int** : numeros enteros positivos o negativos
_ **Float** o sea decimales de punto decimal flotante + o -
_ **String o str (cualquier texto con letras números o signos)**
_ **Bolean** o binario (0 o 1 , equivalente a verdadero o falso “true” o “false”)
_ **Null**

con la instrucion **var_dump**(nombre de variable); podemos imprimir en pantalla que tipo de variable de forma abreviada es y que valor tiene en ese momento y con la instrucion
var-type(nombre) devuelve el tipo de variable completo pero es necesario anteponerle una instrucion **print** o **echo** para que muestre en pantalla o se puede guardar el resultado en otra variable para un uso posterior.

Una variable cualquiera con un dato se puede redefinir como variable null simplemente reescribiendola con un dato null o NULL y se convertirá en variable tipo null o sea nula , sin valor.

ADICIÓN Y SUSTRACCIÓN:

Si después de asignar un valor numérico $x por ejemplo , luego escribimos $x += 9 el resultado sera la suma del valor de $x + 9

El mismo concepto funciona con la sustracción $x -= 9 y el resultado será la resta.
Y también funciona con *= y con /= .

Los operadores usados en comparaciones son :

== igual ,
= = = significa idéntico (sin espacios)
!= distinto
> mayor
< menor

La diferencia entre igual e idéntico es que para que sea idéntico tiene que no solo contener eel mismo valor sino ademas tiene que ser el mismo tipo de dato. Por ejemplo un String "10" tiene el mismo valor numérico que un integer 10 pero son diferentes tipos de datos , entonces son iguales poro no son idénticos.

Pre incremento/decremento y post incremento/decremento

Al escribir ++$x esto devuelve el valor de $x incrementado en una unidad, si $x valía 10 retornará 11 en ese acto, es decir que lo sumará antes de mostrarlo o consultarlo es lo que se llama
pre incremento.
Al escribir $x++ devolverá en ese acto el valor que tenía $x y luego lo sumara , es decir que la siguiente vez que se consulte mostrará el valor incrementado pero en esa instancia mostrará el mismo valor , eso se llama post incremento.
El mismo concepto vale para sustracción o resta --$x es un pre decremento de una unidad y $x-- es un post decremento de una unidad.

Instrucciones condicionales:

if , **elseif**, **else** , **switch**

la sintaxis es :

```
if ( condición) {
        lista de instrucciones
        condición verdadera
        }
elseif ( otra condición )
        {
        lista de instrucciones ;
        en caso que la 1º condición sea falsa ;
        y la otra condición sea verdadera
        }
else {
    lista de instrucciones ;
    en caso que la1º condición ;
    y la otra sean falsas ;
    }

switch ( variable a consultar ) {
    case ¨valor-variable-x¨ :
            lista de instrucciones ;
            si x es verdadero ;
        breack;
    case “valor-variable-y”:
            lista de instrucciones ;
            si y es verdadero ;
        breack;
    case “valor-variable-z”:
            lista de instrucciones ;
            si z es verdadero ;
        breack;
    default:
            lista de instrucciones si no se da;
            ninguno de los 3 casos;
    }
```

Bucle WHILE:

1º_ Inicialización

Es una instrucción previa al bucle que consiste en cargar una variable local de cualquier tipo con un valor para que sea consultada durante la ejecución del bucle de modo que cuando el valor cambie en algún punto de la ejecución dará por terminado el bucle

bucle :

while (condición) {
lista de instrucciones;
a ejecutar siempre que la condición;
sea verdadera;
incremento o sustracción o cambio de estado de la variable que permita alcanzar una instancia en que se de por finalizado el bucle;
}

Bucle For:

for($x=valor ; $x< o > o == valor ; $x+ o – valor incremento) {
inicial límite o dec. por ciclo

lista de instrucciones ……. ;
a ejecutar ……... ;
dentro del bucle for ……... ;
}

no se requiere inicializar la variable que se usara de referencia por que en la 1º linea en que se define el **for** ya se esta inicializando con el valor inicial.

ARREGLOS:

Hay 3 tipos de arreglos:

- **Indexados**
- **Asociativos**
- **Multi-dimensionales**

Los arreglos indexados son arreglos en los que se pueden almacenar valores referidos a un índice numérico que empieza en 0 y se puede acceder a dichos valores llamándolos por el numero de índice entre corchetes.

ejemplo de arreglo indexado:

```
$usuarios = array( ¨Juan¨, ¨ Pedro¨, ¨Jose¨, ¨Mario¨);
echo $usuarios[0];
```

y mostrará Juan y sucesivamente con [1] o [nº]

También se pueden reemplazar, en array los paréntesis por corchetes y es el mismo resultado.
La instrucción count(nombre de arreglo) permite saber el número total de elementos de un arreglo.

Los arreglos asociativos son arreglos que almacenan valores ordenados de acuerdo a una clave y se puede acceder a ellos invocando a la clave asociada al valor.

Por ejemplo :

```
$edad=[´Juan´ => 33, ´Pedro´ => 22, ´Anibal´  => 32];

echo $edad[´Juan´ ];

y devolverá  33
```

Podemos listar el contenido del arreglo con la instrucción foreach y sumarle strings para presentarlo como en el siguiente ejemplo:

```
foreach ($edad as $key => $value) {
        echo “nombre : ” . $key . “edad :  “ .$value;
        echo <br>;
}
```

y devolverá:

```
nombre : Juan edad : 33
nombre : Pedro edad: 22
nombre : Anibal edad : 32
```

Los arreglos multi-dimensionales son básicamente arreglos de arreglos.

Por ejemplo definiremos un arreglo multi-dimensional llamado $multi

```
$multi = array(
                array(‘php’, ‘java’, ’javascript’, ’python’),
                array(´laravel´, ´sinfony´, ´nodej´, ‘fesk’),
                array(´mysql´, ´mongodb´, ´couchdb´, ‘pandas’)
                );
```

Una vez definido accederemos al contenido de cada elemento por su posición invocando el número de orden del arreglo dentro de $multi (o sea de fila) y el numero de orden del elemento dentro de cada arreglo (o sea de columna).
Así si por ejemplo:

```
    echo $multi[0][0];
devuelve:   php

    echo $multi[1][1];
devuelve:   sinfony;

   echo $multi[2][2];
devuelve:   couchdb
```

FUNCIONES:

Las funciones son una instrucción o conjunto de instrucciones que se ejecutarán repetidamente a lo largo del programa y que mediante la definición de la función se llamará a la función cuantas veces se requiera durante el desarrollo del programa sin tener que reescribir las mismas instrucciones repetidamente. Esto entra dentro de los métodos conocidos como DRY (do not repeat your self)
Para definir la función la sintaxis es:

```
function nombrefunción() {
                                    lista de ;
                                    instrucciones;
                                    a ejecutar;
                                    por la función;
                                    }
```

El nombre de la función puede iniciar con un guion bajo o una letra , dentro del nombre puede haber letras o números pero no pueden haber puntos ni espacios y las letras pueden ser mayúsculas o minúsculas y esto no tendrá efecto al invocar la función , es decir que si la función la llamamos por ejemplo miFunción() y luego la invocamos como mifunción() se ejecutará de todos modos.
Para ejecutar la función la instrucción es simplemente:

nombrefunción();

Funciones con argumentos:

Los argumentos de función son básicamente variables que se pasan a la función a través de los paréntesis que siguen al nombre de función:

```
funciónmultiplica($x, $y) {
        # por ejemplo
        echo $x * $y;
                        }
```

y mostrará el producto de $x por $y .

FUNCIONES SUPER GLOBALES:

Son un conjunto de funciones pre-definidas como variables especiales dentro de PHP a las que se puede acceder tanto en el ámbito local como en el ámbito global, de modo que son variables a las que se puede acceder como si fueran constantes.
Existen cientos de variables globales pero vamos a ver las mas importantes:

- **server**
- **request**
- **post**
- **get**
- **sessions**
- **cokies**

Reseña de la función de cada una:

- **server** verifica el request que puede ser por un **post** o un **get** y también verifica IP como la clase de explorador con que el usuario esta trabajando , su sistema operativo y algunas configuraciones del servidor.

- **request** tiene una función similar a server en lo que respecta a la verificación de post y get.

- **post envia datos desde un formulario a un programa especificnando la URL de destino y los recibe y procesa en el programa de destino** como datos que esta buscando o que esta despachando a Instagram o Faceboock. Permite enviar grandes vol'umenes de informacion de forma textual o binaria como imagenes, etc . Y la informacion y destinatario no son visibles en el proceso , por lo que es posible enviar satos encriptados, reservados o sensibles.

- **get** hace lo mismo que post pero con el limite que URL del destinatario y la información son visibles en el proceso y el volumen de informacón por cada dato es de un máximo de 2000 caracteres y solo se pueden enviar caracteres textuales.

- **sessions** almacena algunos datos que serán usados a lo largo de todo el website.

- **cokies** almacena cierta información en el explorador. Es similar a session pero cokies esta instalado en el explorador y sessions esta instalado en el código del programa o website que construimos.

POST:

Ejemplo de código usando el método post :

El siguiente formulario HTML llamado “ejemploPost.html” captura el dato “nombre” y al dar Enter en submit/ lo despacha por el método post al programa PHP llamado “ejemploPost.php”

este programa php esta en el mismo servidor y carpeta que el formulario pero en el parámetro action se podria incluir la URL de otro servidor y asi se puede transferir información de un servidor a otro.

```
<!DOCTYPE html>
<html>
   <head>
       <title>ejemploPost.html</title>
   </head>
  <body>
   <h3>FORMULARIO HIPERTEXTO DE CAPTURA Y ENVÍO DE INFORMACIÓN</h3>
   <form name="formulario" method="post" action="ejemploPost.php">
   Nombre: <input type="text" name="nombre"value="">
   <input type="submit"/>
   </form>
  </body>
</html>
```

El siguiente programa llamado “ejemploPost.php” es el que recibe la información “nombre”enviada por el formulario de arriba .

```
<html>
   <head><title>ejemploPost.php</title></head>
  <body>
   <h3>PROGRAMA P.H.P. RECIBE INFORMACIÓN</h3>
   <form>Tu nombre es : <?php isset($_POST['nombre']) ? print $_POST['nombre']:"";?>
   </form>
   <br />
  </body>
</html>
```

La información enviada por el método POST puede contener caracteres de texto,números o datos binarios. Puede contener todo un archivo con imágenes , etc de gran tamaño y la información , su origen y destino permanecen ocultos.

GET:

Ejemplo usando el método GET:

El siguiente formulario HTML llamado “ejemploGet.html” captura el dato “nombre” y al dar Enter en submit/ lo despacha por el método Get al programa PHP llamado “ejemploGet.php”

este programa php esta en el mismo servidor y carpeta que el formulario pero en el parámetro action se podría incluir la URL de otro servidor y así se puede transferir información de un servidor a otro.

```
<!DOCTYPE html>
<html>
   <head>
       <title>ejemploGet.html</title>
   </head>
  <body>
   <form name="formulario" method="get" action="ejemploGet.php">
   Nombre: <input type="text" name="nombre"value="">
   <input type="submit"/>
   </form>
  </body>
</html>
```

El siguiente programa llamado “ejemploGet.php” es el que recibe la información “nombre”enviada por el formulario de arriba .

```
<?php
/* EJEMPLO GET.PHP */
$nombre = $_GET['nombre'];
echo $nombre;
?>
```

La información enviada por el método GET puede contener únicamente caracteres de texto y esta limitada 2000 caracteres , ademas la información de destino puede ser vista en el explorador y los datos contenidos pueden llegar a ser interceptados .Por eso es que no se considera un medio recomendable para transferir información sensible o reservada.

El método GET también es muy usado para transferir datos dentro del mismo programa cuando se ingresan datos desde un código HTML dentro de un programa PHP para tomar alguna acción en base a esos datos.

REQUEST:

La variable superglobal REQUEST sirve para recuperar en un programa PHP uno o varios datos que han sido enviados a ese programa mediante una variable GET o POST.

Asi , si en el ejemplo de programa ejemploPost.PHP se podría haber utilizado las instrucciones :

```
$nombre = $_REQUEST[‘nombre’];
print $nombre;
```

en lugar de `<?php isset($_POST['nombre']) ? print $_POST['nombre']:"";?>`

y el resultado hubiera sido el mismo ,o también en el programa ejemploGet.php , se podría haber usado la misma instrucción

```
$nombre = $_REQUEST[‘nombre’];
```

en lugar de `$nombre = $_GET[‘nombre’];`

con el mismo resultado.

SESSIONS:

La variable sessions es una forma de almacenar información que podemos utilizar posteriormente a través demultiples a sitios web.
Cuando ejecutamos una aplicación en nuestra computadora ,ella sabe quienes somos y que estamos haciendo pero cuando accedemos a un website o una aplicación web no sabe quioenes somos oque hacemos ,entonces puede acudir a la información que esta almacenada , que consiste básicamente en una cantidad de caracteres y números almacenados en la computadora que sessions puede identificar .

Ejemplo de uso de la variable superglobal SESSIONS:

```
/*session1.php*/
<?php
/* ESTE PROGRAMA CARGA ALGUNOS DAOS DE SESION EN LA VARIABLE  SUPERGLOBAL
SESSION PARA SER USUDA LUEGO POR OTROS PROGRAMAS , SOLO A TÍTULO DE EJEMPLO */
session_start();
$_SESSION['nombreusuario'] = "pepe-capo";
$_SESSION['edad'] = 64;
$_SESSION['colorfavorito'] = "azul";
?>

/* session2.php */
<?php
/*ESTE PROGRAMA CAPTURA LA INFORMACION ALMACENADA EN SESSIONS POR EL
PROGRAMA session1.php CUANDO SE EJECUTO Y LUEGO BORRA ESTA INFORMACIÓN
PARA QUE NO SEA CAPTURADA POR OTRO INTRUSO , ES DECIR "CIERRA LA SESIÓN"
*/

session_start();
?>
<html>
  <head></head>
  <body>
    <h3><?php
        echo $_SESSION['nombreusuario'] . " su edad es " . $_SESSION['edad'];
        ?>
    </body>
 </html>

/* session3.php */
<?php
/* ESTE PROGRAMA BORRA TODOS LOS DATOS DE SESION CARGADOS EN ELPROGRAMA session1, ES DECIR , EJECUTA EL CIERRA DE SESION PARA QUE NINGUN INTRUSO PUEDA LEER
LOS DATOS DE LA SESION */

session_start();
session_unset();
session_destroy();
echo " LA SESION SE HA CERRADO ";
echo "<br>";
?>
```

TIME:

Para obtener la zona horaria se usa la función: **date_default_timezone_get();**
Con esto se obtiene la zona horaria del servidor , no de la computadora donde se esta realizando la consulta.

Para establecer la zona horaria de donde se esta ejecutando el programa se hace con la instrucción
date_default_timezone_set('America/Argentina/Buenos_Aires');
En el ejemplo se tomo la zona horaria de la lista de zonas permitidas desde www.php.net/manual/en/timezones.php con la salvedad que se debe escribir sin acento y sin espacios, no como aparece escrita en la lista.

Para obtener la hora se usa la funcion date(‘H’) para obtener en el formato de 0 a 24
o date(‘h’) para el formato 00 a 12 (tarde o mañana).

date(‘h:i’) muestra horas:minutos
date("H:i,a :s"); muestra Horas(0a12):minutos, am o pm :segundos
date('Y-m-d H:i,a :s') muestra año-mes-dia hora:minutos, am o pm :segundos
date('d-m-Y H:i,a :s'); muestra día-mes-año hora:minutos, am o pm :segundos

SERVER:

El siguiente código muestra algunas de las variadas informaciones que se pueden obtener de la variable superglobal SERVER:

```
<?php
/* EJEMPLO SERVER.PHP */
   $path = $_SERVER['PHP_SELF'];
   $gateway = $_SERVER['GATEWAY_INTERFACE'];
   $ip = $_SERVER['SERVER_ADDR'];
   $sname = $_SERVER['SERVER_NAME'];
   $ssoft = $_SERVER['SERVER_SOFTWARE'];
   $sprot = $_SERVER['SERVER_PROTOCOL'];
   $reqmet = $_SERVER['EQUEST_METHOD'];
   $browser = $_SERVER['HTTP_USER_AGENT'];
   $RIP = $_SERVER['REMOTE_ADDR'];
   $RHost = $_SERVER['REMOTE_HOST'];
   $RUS = $_SERVER['REMOTE_USER'];
   $RPort = $_SERVER['REMOTE_PORT'];
   echo " La ubicación de este programa es : $path";
   echo '<br>';
   echo " El Gateway Interface o Número de revisión de la especificación
CGI del          servidor es: $gateway";
   echo '<br>';
   echo " La dirección IP local del servidor es : $ip";
   echo '<br>';
   echo " El nombre del servidor es : $sname";
   echo '<br>';
   echo "El programa utilizado por el servidor es : $ssoft";
   echo '<br>';
   echo "El protocolo y version utilizado por el servidor  es: $sprot";
   echo '<br>';
   echo "El método utilizado para acceder a esta página es: $reqmet";
   echo '<br>';
    echo "El la cadena de browsers intermediarios usados para Browser
aceder a esta   página  (destino - origen) es : ";
   echo '<br>';
   echo $browser;
   echo '<br>';
     echo "La dirección IP de su dispositivo (desde el cual Ud.esata
observando esta          página) es : $RIP";
   echo '<br>';
    echO "El nombre del servidor desde el cual Ud. esta accediendo a esta
página es :      $RHost";
   echo '<br>';
   echO "Su nombre de usuario en su dispositivo es : .$RUS.";
   echo '<br>';
    echO "En n° de puerto con que su dispositivo se conecta a Internet para
acceder a        esta página es : .$RPort.";
   echo '<br>';
 ?>
```

MySQL

Crear y Borrar Bases de datos:

Para comenzar debemos ,con el browser dirigirnos a localhost/phpmyadmin
Nos dirigimos a la solapa deDatabases y en la casilla debajo de Create Database le asignamos un nombre a una nueva base de datos y en este punto podriamos crearla base de datos simplemente digitando el boton Create pero para aprender las sentencias SQL nos dirigimos a la solapa de SQL y en dicho cuadro de texto digitamos las siguientes instrucciones:
(normalmente las palabras reservadas de SQL se escriben en mayúsculas y se veran de color rojo)

CREATE DATABASE testdb; (le damos al boton GO y se ejecuta la sentencia y podemos obserbar que en la lista de bases de datos almargen izquierdo aparece la nueva base de datos)

USE testdb; (boton GO y en ese momento estamos usando la base de datos)

DROP DATABASE testdb; (le damos GO y observaremos que la basede datos desaparece de la lista, la hemos borrado o dado de baja)

Crear tablas:

Para crear una tabla seleccionamos una base de datos de entre la lista que aparece al margen izquierdo de phpMyAdmin (en este caso usaremos CURSOPHP, por ejemplo) y aparecerá la solapa de Estructura. Si la base de datos todavía no tiene ninguna tabla los avisara en el margen superior que dice “No tables found in database.” y veremos en la sección superior izquierda el título Create table con el casillero debajo invitando a escribir un nombre para la tabla.
A fin de familiarizarnos con el lenguaje SQL nos dirigimos a la solapa SQL y escribimos las siguientes sentencias:

```
1 CREATE TABLE users (
2
3       id INT(2) NOT NULL,  # crea el campo "id" o índice con números enteros de 2 dígitos que no puede ser nulo
4       username VARCHAR(20), # crea el campo username tipo textual ,longitud variable máximo 20 caracteres
5       email VARCHAR(20),
6       password VARCHAR(30)
7
8       );
```

Luego le damos a GO o Continua y veremos que al margen izquierdo aparece bajo CURSOPHP un árbol donde se puede ver una ramificación NEW y debajo el símbolo de tabla con nombre users y si le damos sobre el ícono de esa tabla se abrirá la pantalla de estructura mostrando los campos y todas sus características.

Dar de baja una tabla:

De forma similar que para crearla , seleccionamos la base de datos , nos dirigimos a la solapa SQL y escribimos la instrucción:

```
1 DROP TABLE users
```

Luego le damos a GO o Continuar y nos anunciara que la tabla ha sido borrada pero sigue apareciendo al margen izquierdo bajo la base de datos. Entonces refrescamos el índice de base de datos con el botón verde a la derecha del engranaje en la parte superior del margen izquierdo y veremos que al refrescarse el índice desaparece la tabla.

Insertar registros en una tabla:

Seleccionamos la base de datos , nos dirigimos a la solapa SQL y escribimos la instrucción:

```
1 INSERT INTO users (id, username, email, password)
        VALUES (1, "usuario1", "usuario1@gmail.com", "usuario1111");
```

Luego le damos a GO o Continuar y si seleccionamos la tabla en la pestaña Examinar veremos que aparece debajo del encabezado con los nombres de los campos los respectivos datos que se han ingresado en cada campo.

También se pueden insertar varios registros en una sola instrucción INSERT

```
1 INSERT INTO users (id, username, email, password)
        VALUES (1, "usuario2", "usuario1@gmail.com", "usuario2345"),
        (2, "usuario3", "usuario3@gmail.com", "usuario678"),
        (3, "usuario4", "usuario4@gmail.com", "usuario910"),
        (4, "usuario5", "usuario5@gmail.com", "usuario112");
```

Le damos a GO o Continuar y veremos que se han insertado 4 registros más.

Agregar y borrar registros en una tabla:

Si escribimos el siguiente script SQL:

```
1 UPDATE users SET username = “user number 4”;
```

Lo que hará es reemplazar el contenido del campo username de todos los registros de la tabla users. Por eso es muy importante especificar en que numero de registro se debe operar el cambio. Eso se establece con el parámetro WHERE.
Por ejemplo:

```
1 UPDATE users SET username = ‘usuario nº 4’ WHERE id = 4;
```

(nota : se debe usar siempre comillas simples y rectas para envolver los textos)

Pero el id 4 no existe enla tabla por que cuando insertamos los registros 2 al 5 les dejamos el id = 1 en los registros 3, 4 y 5.
Para corregir esto escribiremos las siguientes instruciones SQL:

```
1 UPDATE users SET  id = 3  WHERE username = ‘usuario3’;
2 UPDATE users SET  id = 4  WHERE username = ‘usuario4’;
3 UPDATE users SET  id = 5  WHERE username = ‘usuario5’;
```

A continuación si podremos ejecutar el cambio del ‘usuario4’ por ‘usuario n.º 4 en el registro con id = 4

También es posible cambiar el contenido de varios campos del mismo registro a la vez con una sola instrucción como por ejemplo:

```
1 UPDATE users SET username = ‘usuario nº 4’, email = ‘user4@yahoo.com.ar’
WHERE id = 4;
```

Para borrar un registro se hace con la instrccion DELETE FROM , como por ejemplo:

```
1 DELETE FROM users where id= 5;
```

Ademas del campo id usado en el ejemplo , se puede usar cualquier otro campo y condición como referencia para encontrar el registro a borrar.

Tipos de datos SQL :

los mas importantes son:

int => números enteros ±

decimal => números con punto decimal flotante.

datetime => contiene fecha y hora.

date => contiene la fecha.

time => contiene la hora.

varchar => contiene string de caracteres letras, números o signos en una cantidad limitada a la longitud definida para cada variable varchar.

text => contiene string de caracteres de gran cantidad de caracteres, puede contener varias páginas de texto , letras, números y símbolos, es usado para albergar el contenido de blogs , por ejemplo.

null => es una variable de contenido nulo.

Operadores SQL :

Un operador es una palabra reservada usada previamente o dentro de una clase. Los 3 principales operadores son:

Operadores aritméticos: +, -, / , * ,%

Operadores comparativos: =, !=, >, <

Operadores lógicos: [and or],

CLAVES:

Claves primarias:

Es una constante que identifica cierta columna o campo de la tabla como un único valor. Muchas veces es necesario hacer una columna de la tabla sea única e identificable a efectos de evitar confusiones por datos que se pueden repetir en otros campos o columnas, esa es una de las funciones de la clave primaria.

En 1° lugar el contenido de la clave primaria no puede ser nulo y en 2° lugar debe ser único y no repetido y lo mas importante: solo se puede haber una única columna o campo en la tabla que cumpla la función de clave primaria. En el 99% de los casos se creará la clave primaria como ID, normalmente no la usara como nombre de usuario ni nada de eso.

Ejemplo de script SQL para crear una tabla con una clave primaria:

```
1 | CREATE TABLE blogs (
2 |   id INT(2) NOT NULL,
3 |   title VARCHAR(20),
4 |   body TEXT,
5 |   PRIMARY KEY(id)
6 | );
```

Claves foráneas:

Es una clave en su tabla que apunta a alguna clave primaria en otra tabla.

Esto es útil para , por ejemplo , en las dos tablas creadas como Users y Blogs necesitaremos recoger algunos usuarios y sus blogs. Todo lo que necesitamos hacer es relacionar las dos tablas. Para eso es la clave foránea.
Entonces en este ejemplo. Primeramente nos dirigimos en phpmyadmin a la estructura de la tabla users y establecemos el campo ID de dicha tabla como clave primaria. Para ello en estructura nos dirigimos a la linea correspondiente al campo ID y simplemente hacemos click en el ícono correspondiente a una llave con la leyenda “primaria” y veremos que junto al nombre del campo id aparece el ícono de la llave indicando que ya esta activo como clave primaria.

Una vez echo esto nos dirigimos a la otra tabla a relacionar con esta , en este caso la tabla blogs, y abrimos la zolapa de estructura y seleccionamos agregar una columna (en la parte inferior) y le damos continuar.
El el casillero correspondiente al nombre de la columna le ponemos un nombre representativo , por ejemplo users_id (haciendo referencia a que se trata de la columna id de la tabla users) y le asignamos el mismo tipo y longitud , en este caso INT(2) y le damos a Guardar .
Veremos un anuncio indicando que los cambios se han realizado y si observamos la estructura de blogs , veremos que se ha agregado la nueva columna o campo, (normalmente lo agrega como último campo o columna ,a menos que especifiquemos lo contrario).

A continuación nos dirigimos ala solapa de SQL con el propósito de relacionar la clave foránea.
Y escribimos el siguiente script SQL:

```
ALTER TABLE blogs ADD FOREIGN KEY (users_id) REFERENCES users(id)
```

Luego volvemos a la vista de estructura y veremos que aparece junto al nombre del campo una llave de color gris indicando que esta asignado como una clave secundaria.
Por último para verificar que la relación este realmente establecida nos dirigimos al margen izquierdo y seleccionamos la base de datos a la que pertenecen ambas tablas. Una vez seleccionadas nos dirigimos a la barra superior y al extremo derecho , donde dice Mas ,desplegamos la opciones y seleccionamos diseñador y entonces debe aparecer un gráfico donde aparecen las 2 tablas y una linea verde vinculando el campo id de la tabla users con el campo users_id de la tabla blogs. Esto demuestra que el vínculo esta realmente establecido. Desde allí se puede anular en cualquier momento o también se puede establecer en forma gráfica mediante la herramienta que aparece como una linea de vínculo en la barra de herramientas al margen izquierdo del gráfico y a la derecha de la columna de bases de datos, al extremo izquierdo de la pantalla.

FUNCIONES SQL:

Las funciones son para hacer ciertas cosas en las bases de datos o solo para extraer alguna pieza de información. Como por ejemplo el mínimo o promedio de una columna o campo o contar el número de filas o registros de este campo. Obviamente tenemos muchas funciones, vamos a analizar solo algunas funciones básicas.
Uno de las instrucciones es SELECT, este ya lo hemos usado unas 3 veces anteriormente.
Por ejemplo: calculemos el promedio de valores del campo id de la tabla users.

```
1| SELECT AVG(id) FROM users
```

y al darle en Continuar nos arrojará un resultado de 2.5000 bajo el rótulo **AVG(id)**

Otro ejemplo: sumaremos la cantidad de registros o filas en el campo email de la tabla users.

```
1| select count(email) from users
```

y nos arrojará un total de 4 bajo el rótulo **count(email)**.
(como podemos ver , la función también funciona si la escribimos en minúsculas , las mayúsculas son solo una convención).

algunas de las funciones mas comunes son SUM , AVG, COUNT, MIN, MAX

Naturalmente las funciones puramente aritméticas como SUM y AVG funcionarán solo con campos que contengan datos numéricos INT o DECIMAL mientras que COUNT , MIN O MAX funcionan con cualquier tipo de datos pues cuanta o compara elementos.

La instrucción select se usa para seleccionar algunos datos de nuestra base de datos desde nuestras tablas y por supuesto este tiene un montón de opciones y se pueden hacer una búsqueda mas fácil o mas compleja y esto puede hacer tu aplicación mas rápida o mas lenta.
Podemos escribir select * y esto seleccionará todos los datos de su tabla y podemos seleccionar ciertas columnas o podemos seleccionar solo una columna y vamos a ver ejemplos con operadores aritméticos y comparativos y vamos a usar operadores lógicos.

Aclararemos que dentro del script SQL los 2 guiones medios como - - sirven para hacer comentarios de una linea. Para los comentarios de múltipleslineas se usa /* ……..*/ igual que en el script de PHP.

Si escribimos por ejemplo : SELECT password FROM users
listará todo el contenido de la columna o campo password de la tabla users.
Y si escribimos : SELECT password, id FROM users
listará todo el contenido de la columna o campo password junto a todo el contenido de la columna o campo id de la tabla users.

Si escribimos por ejemplo : SELECT username FROM users WHERE id = 2
listará : usuario3 , que es el nombre del usuario contenido en el registro con id = 2
Y si escribimos : SELECT * from users WHERE id = 2

listará el id, username, email y password del registro con id = 2 , es decir todo el contenido de ese registro.
Si ahora escribimos : SELECT * FROM users WHERE id != 2
listará todo el contenido de los registros con id 1, 3 y 4 , es decir , todos los id distintos a 2.
Lo correspondiente sucederá si usáramos un operador comparativo en la búsqueda.

Si escribieramos : SELECT * FROM users WHERE id = 2 AND username = “usuario3”
mostrará el contenido de todo el registro con id 2 y username “usuario3” por que cumple con ambas condiciones.

ORDER BY:

Primero vamos a grabar algunos registros en la tabla blogs con las ue vamos a trabajar:

```
INSERT INTO blogs (id, title, body, users_id)
        VALUES (1, ‘title1’, ‘body1’, 1),
                (2, ‘title2’, ‘body2’, 2),
                (3, ‘title3’, ‘body3’, 2),
                (4, ‘title4’, ‘body4’, 1);
```

Ahora vamos a pedirle que liste todos los campos y registros de blogs ordenados por id

```
SELECT * FROM ‘blogs’ ORDER BY id
```

(no especificaremos si queremos orden ascendente o descendente por lo que tomara por defecto ascendente)

Ahora vamos a pedirle lo mismo en forma descendiente.

```
SELECT * FROM `blogs` ORDER BY id DESC
```

Ahora haremos lo mismo con la tabla users

```
SELECT * FROM `users` ORDER BY id DESC
```

Si queremos que liste solo 2 de los campos , podemos especificar :

```
SELECT username, email FROM `users` ORDER BY id DESC
```

Y listara solo esos 2 campos en el orden especificado.

O también podemos pedirle que lo ordene ascendente-mente por nombre de usuario.

```
SELECT username, email FROM `users` ORDER BY username ASC
```

Nota: En la sintaxis de los query es muy sensible el uso de ciertos tipos de comillas simples , en el caso de los valores de campos textuales se deben usar comillas verticales (‘ ’) pero en el caso de los nombres de las tables se deben usar comillas hacia atrás (` `). De lo contrario da error.

GROPUPS:

Primero crearemos una tabla que nos servirá de ejemplo:

```
CREATE TABLE  cats (
        id int(2) PRIMARY KEY AUTO_INCREMENT,
        name varchar(20)
        );
```

En este caso el valor de id no necesitaremos ingresarlo por que al estar definido como autoincrementable se incrementara a medida que se ingresen nuevos registros.

Ahora ingresaremos algunos datos en la tabla cats:

```
INSERT INTO cats (name)
                VALUES (‘php’), (‘js’), (‘ruby’);
```

A continuación le agregaremos manualmente , desde phpmyadmin una nueva columna o campo a la tabla blogs a continuación del último campo que llamaremos cat_name y definiremos como VARCHAR(20).
Este campo funcionara en forma similar a una clave foranea pero no lo será , si no que sera solo un campo relacional.
Y a continuación cargaremos en ese campo algunos datos que ya emos introducido en el campo name de la tabla cats .

A continuación escribiremos el siguiente query con la función GROUP.

```
SET sql_mode = ''; /* esto hace posible que se puedan agrupar
            diferentes tipos de datos , de lo contrario el siguiente
            query daría error por que sin esta instrucción solo
            puede agrupar por la misma variable que selecciona */
```

```
SELECT title, body, cat_name, COUNT(*) FROM blogs GROUP BY cat_name;
```

Y el resultado que arrojará con los datos disponibles será:

title	body	cat_name	COUNT(*)
title2	body2	js	2
title1	body1	php	1
title4	body4	ruby	1

Como se ve este query o consulta contó las ocurrencias de los datos del campo cat_name y exhibió otros datos del registro donde halló la 1° ocurrencia (si tuviera mas de una).

ALTER:

Es un comando para modificar,agregar o borrar elementos de la tabla. A continuacio veremos como se usan para modificar las columnas o campos.

Por ejemplo vamos a agregarle a la tabla blogs un campo llamado created_at de tipo timestamp como para almacenar (las fechas de creación de cada blog):

ALTER TABLE blogs ADD COLUMN created_at timestamp

Luego de ejecutarlo al revisar la estructura podremos ver que se ha agregado el campo al final de los registros y al examinar el contenido veremos:

+ Opciones

		id	title	body	users_id	cat_name	created_at
■	Editar Copiar Borrar	1	title1	body1	1	php	2021-07-31 12:18:25
■	Editar Copiar Borrar	2	title2	body2	2	js	2021-07-31 12:18:25
■	Editar Copiar Borrar	3	title3	body3	2	js	2021-07-31 12:18:25
■	Editar Copiar Borrar	4	title4	body4	1	ruby	2021-07-31 12:18:25

Como se puede ver en todos los registros del campo created_at guardo por omisión la fecha y hora del momento en que fue creado.

Y si ahora quisieramos borrar esa columna , ejecutaremosla instrucción :

ALTER TABLE blogs DROP COLUMN created_at

Y al revisar nuevamente la estructura veremos que el campo efectivamente ha desaparecido.

ALIASES:

Los alias son simplemente falsos nombres o nombres sustitutos que se pueden usar temporalmente para llamar a las columnas o campos según sea necesario.
Por ejemplo si tiene un largo y complejo nombre y queremos usar un nombre mas corto y/o simple o para mostrarlo en la cabecera de una consulta en una forma mas coloquial o en otro idioma.

Por ejemplo :

SELECT title AS Título FROM blogs

Y a continuacion listará el contenido de la tabla blogs de la siguiente forma :

Título
title1
title2
title3
title4

SELECT id AS N°, title AS Título, users_id AS N°_de_usuario FROM `blogs`

Y el resultado será:

N°	Título	N°_de_usuario
1	title1	1
2	title2	2
3	title3	2
4	title4	1