**EJERCICIO SQL**  
  
Se poseen las siguientes tablas:

|  |  |  |  |
| --- | --- | --- | --- |
| **USUARIOS** | **FAVORITOS** | **USUARIOSPAGOS** | **PAGOS** |
| codigousuario usuario clave edad | codigousuario codigousuariofavorito | codigopago codigousuario | codigopago importe fecha |

Aclaración: la tabla "favoritos" une usuarios con otros usuarios a traves de los campos codigousuario y codigousuariofavorito. Un ejemplo seria que si el usuario con código 1 tiene de favoritos a los usuarios de código 5,6 y 8 en la tabla figurara:

|  |  |
| --- | --- |
| **Codigousuario** | **codigousuariofavorito** |
| 1  1  1 | 5  6  8 |

**A)** Encuentre el error en la siguiente consulta, si la intención era devolver todos los nombres de usuario (campo "usuario") y clave de los usuarios que tengan 18 años, ordenados alfabéticamente por el nombre de usuario

SELECT USUARIO, CLAVE FROM USUARIOS ORDER BY CODIGOUSUARIO

**Solución**, en rojo el lo que faltaba, no hay error de sintaxis, sólo de lógica, no hay una clausula **where** para filtrar por edad y se ordenaba por código de usuario y no por nombre**:**

***SELECT USUARIO, CLAVE FROM USUARIOS WHERE EDAD = 18 ORDER BY USUARIO***

**B)** Encuentre el error en la consulta, que debería devolver todos los usuarios con sus favoritos, únicamente mostrando nombres de usuario: en la primera columna que se muestre el nombre de usuario del dueño del favorito, y en la segunda que aparezca el nombre de usuario del favorito. Si le parece necesario, reescriba la consulta

SELECT USUARIOS.USUARIO, FAVORITOS.CODIGOUSUARIO

FROM USUARIOS

INNER JOIN FAVORITOS ON FAVORITOS.CODIGOUSUARIO = USUARIOS.CODIGOUSUARIO

**Errores:**

* No se listan las columnas correctas.
* El join no es suficiente, se necesita otro join, como está, no obtendrá los registros corrctos correspondientes.

**Solución:**

*SELECT u.usuario as "Nombre de Usuario", fu.usuario as "Favorito"*

*FROM usuarios u*

*INNER JOIN favoritos f ON ( u.codigousuario = f.codigousuario )*

*INNER JOIN usuarios fu ON ( f.codigousuariofavoritos = fu.codigousuario )*

**C)** Indique que índices (sean clave primaria o no) debería tener cada tabla, y por qué.

* Un indice primario para cada tabla (porque la primera normalización lo debe tener, además es necesario para el manejo relacional).
* La tabla favoritos debe tener dos claves foraneas que juntas funcionan como clave primaria compuesta para evitar duplicación y facilitar el indexado.
* En la tabla usuarios puedo suponer que el campo usuario será de indice único, aunque no hay nada que lo indique en los enunciados, más allá de la experiencia con tablas de usuarios.

**D)** ¿Existe algo de la estructura de la base de datos que cree se debería modificar? ¿Por qué?

|  |  |  |  |
| --- | --- | --- | --- |
| **USUARIOS** | **FAVORITOS** | **USUARIO\_PAGO** | **PAGOS** |
| ID usuario clave fecha\_nacimiento | ID\_USUARIO ID\_USUARIO\_FAVORITO | ID\_PAGO ID\_USUARIO | ID importe fecha |

Aquí ralmente estoy es abstrayendo un poco más los nombres de las columnas para que sean más asociativos con los nombres de sus tablas, es decir, al hacer usuarios.id, se muy bien que me refiero a la clave única de la tabla usuarios, claro esta, siempre y cuando no sea obligatorio usar la palabra “codigo” para indentificar dicho campo debido a que es un codigo complejo, alfanúmerico, etc....

En Usuarios cambio edad por fecha, para que la fecha se cálcule y no tenga que actualizarse constantemente.

El nombre de la tabla muchos a muchos, en realidad es más un estandar mio que una generalización, me parece más comodo que se coloque el nombre de cada tabla involucrada en plural, ya que cada fila tendrá una relación única, pero esto depende más de los estándares internos manejados por las empresas o desarrolladores.

**EJERCICIO ALGORITMICO: ORDENAR**

Encuentre el o los errores en la función, si la idea es ordenar un vector de N dimensiones, por cualquiera de ellas, en forma ascendente. Los parámetros deben ser primero el vector, y luego la dimensión ordinal por el cual ordenar. La función debe devolver un nuevo vector ya ordenado.

Ejemplo:

v[0] = array(“nombre” => “Juan”, “apellido” => “Perez”, “edad” => 26);

v[1] = array(“nombre” => “Carlos”, “apellido” => “Gomez”, “edad” => 30);

v[2] = array(“nombre” => “Ernesto”, “apellido” => “Ramirez”, “edad” => 22);

Llamando a la función con los parametros $v y “edad”, la misma debería devolver el listado ordenado por la edad (en forma ascendente), es decir:

w[0] = array(“nombre” => “Ernesto”, “apellido” => “Ramirez”, “edad” => 22);

w[1] = array(“nombre” => “Juan”, “apellido” => “Perez”, “edad” => 26);

w[2] = array(“nombre” => “Carlos”, “apellido” => “Gomez”, “edad” => 30);

FUNCION:

function orderArray($array, $dimension){

$length = count($array);

for ($outer = 0; $outer < $length; $outer++) {

  for ($inner = 0; $inner < $length; $inner++) {

    if ($array[$outer] < $array[$inner]) {

    $tmp = $array[$outer];

    $array[$outer] = $array[$inner];

    $array[$inner] = $tmp;

    }

  }

  }

}

**Código funcionando de dos formas:**

**Por referencia:**

**Error:**

* *No existe el operador referencial en el parametro $array, por tanto el array no es modificado.*
* *No se accede a la clave $dimension en el array al momento de comparar.*

**Solución:**

**function orderArray(&$array, $dimension){**

**$length = count($array);**

**for ($outer = 0; $outer < $length; $outer++) {**

**for ($inner = 0; $inner < $length; $inner++) {**

**if ($array[$outer][$dimension] < $array[$inner][$dimension]) {**

**$tmp = $array[$outer];**

**$array[$outer] = $array[$inner];**

**$array[$inner] = $tmp;**

**}**

**}**

**}**

**}**

**Por valor:**

**Error:**

* *No se devuelve -return- el array una vez ha sido modificado.*
* *No se accede a la clave $dimension en el array al momento de comparar.*

**Solución:**

**function orderArray($array, $dimension){**

**$length = count($array);**

**for ($outer = 0; $outer < $length; $outer++) {**

**for ($inner = 0; $inner < $length; $inner++) {**

**if ($array[$outer][$dimension] < $array[$inner][$dimension]) {**

**$tmp = $array[$outer];**

**$array[$outer] = $array[$inner];**

**$array[$inner] = $tmp;**

**}**

**}**

**}**

**return $array;**

**}**

**EJERCICIO ALGORITMICO: RECURSIVIDAD**

Encuentre el o los errores en la función, que debería recorrer todos los archivos existentes en el directorio “/files/” recursivamente, incluyendo los archivos de todos los subdirectorios. Una vez concluido el recorrido, la función debería devolver un vector que contenga todos los archivos (indicando la ruta completa). Ejemplo:

Vec[0] = “/files/config.inc”

Vec[1] = “/files/connection.php”

Vec[2] = “/files/images/logo.gif”

Vec[3] = “/files/images/ads/160x600.gif”

FUNCION:

$main = "/files/";

function readDirs($main){

$dirHandle = opendir($main);

while($file = readdir($dirHandle){

if(is\_dir($main . $file) && $file != '.' && $file != '..'){

$vec = readDirs($file);

} else{

$vec[] = $main.$file;

}

}

}

**Error:**

* *La verificación de archivo, no se hace usando la barra de separación de directorios, esto evita que se puedan recorrer todos los directorios, además de que se tomaran rutas de dos directorios unidas en uno solo, como un directorio (uno que no existe).*
* *La verificación de ‘.’ (Directorio actual) y ‘..’ (Directorio superior) no está en el lugar adecuado.*
* *Cuando se usa recursivamente la función, al retornar el array directamente sobre el array existente, sobreescribe todo lo que ha sido escrito en él anteriormente.*
* *Al pasar el path/ruta al directorio a la funcion de manera recursiva, se está pasando solo el nombre de lo que se supone es un directorio, pero el final y no la ruta completa, esto evita que se recorra dicho directorio, porque se necesita la ruta completa.*

**Solución (Con algunas mejoras):**

function readDirs ( $rootDir = DIRECTORY\_SEPARATOR ){

$dirHandle = opendir( $rootDir );

$vec = [];

while( $file = readdir ( $dirHandle ) ){

if ( $file != '.' && $file != '..' ) {

$path = ( $rootDir != DIRECTORY\_SEPARATOR ? $rootDir . DIRECTORY\_SEPARATOR . $file : $rootDir . $file );

if ( is\_dir ($path) )

$vec = array\_merge($vec, readDirs($path));

else

$vec[] = str\_replace('./', '.', $path);

}

}

return $vec;

}