PHP Perfect SQL v1.0 (SQL perfectas en PHP)

Muchas veces cuando programamos para web es muy fácil cometer errores en la construcción sintáctica de consultas SQL, por ejemplo cuando tenemos que realizar un INSERT de varios campos que provienen de un formulario y tenemos que realizar concatenaciones de cláusulas y variables y comillas simples etc... lo que supone una gran pérdida de tiempo y eficacia a la hora de programar.

Se acompaña un perf_sql.php con las funciones que se detallan a continuación.

El código puede ser cambiado, mejorado, distribuido libremente. Ha sido programado totalmente por Jose Carlos García (yo) de Distintiva Solutions (www.distintiva.com). Experto en desarrollos web, windows, linux de grandes dimensiones ...

SELECT

get_select($table, $columns, $where='', $order='')

ejemplos:

get_select ('clientes', '*')

SELECT * FROM clientes

get_select ('clientes', '*', 'nombre='jose')

SELECT * FROM clientes WHERE nombre='jose'

get_select ('clientes', '*', 'nombre=jose', 'apellido')

SELECT * FROM clientes WHERE nombre=jose ORDER BY apellido

Antes de ver las UPDATE es necesario profundizar en la siguiente función, muy util.

get_mult_set($a_cols, $a_vals, $simb=',', $sign='=', $comillas=true)

devuelve asignaciones múltiples comúnmente utilizadas en sentencias SQL. Mejor lo vemos con ejemplos.

```
$array_cols= array ( 'nombre',  'url');
$arrat_vals= array( 'Jose',  'www.distintiva.com');
get_mult_set ( $array_cols , $array_vals )  me devolverá
```

nombre='Jose', url='www.distintiva.com'

Muy útil por ejemplo para realizar una UPDATE de una tabla y lo sencillo que es incluir más campos y valores con sólo añadir elementos a los arrays.
Automáticamente lo separa por comas y encierra los valores entre comillas simples.

La función tiene más parámetros. Usando los dos arrays del ejemplo anterior.

get_mult_set ($array_cols , $array_vals, 'OR') me devolverá

nombre='Jose' OR url='www.distintiva.com'

que puede ser muy útil en cláusulas WHERE.

El siguiente parámetro cambia el signo de asignación que por defecto era el '='

get_mult_set ($array_cols , $array_vals, 'AND' , '>') obtendremos

nombre > 'Jose' OR url >'www.distintiva.com'

Y si el último parámetro hace que los valores no estén entre comillas simples, muy útil cuando hay que hacer comparaciones con números en vez de strings.

get_mult_set ($array_cols , $array_vals, 'AND' , '>', false)

nombre > Jose OR url >www.distintiva.com

¿ Y que pasa si estoy mezclando valores string con numericos o simplemente quiero introducir una función propia de MySQL tipo NOW( )? Si el último parámtro quita o pone comillas a todos los valores.
Esto tiene solución, con anteponer !! al valor, veamos un ejemplo

$cols= array('nombre', 'edad', 'fecha', 'url');
$vals= array('Jose', '!!30', '!!now( )', 'www.distintiva.com');
get_mult_set ($ cols , $ vals) me deviolverá

nombre='Jose', edad=30, fecha=now( ), url='www.distintiva.com'

Esta es una función que usaremos mucho en conjunto con otras que veremos y se basa en otra más sencilla que es.

get_simp_set($col, $val, $sign='=', $comillas=true)

que realiza asignaciones sólo de un campo con un valor en vez de muchos campos con muchos valores. Hace el mismo efecto que la función get_mulg_set si los arrays sólo tuvieran un solo valor.

get_simp_set ('edad', '30'); nos devolverá

edad='30'

y como antes, podemos cambiar los parámetros por defecto

get_simp_set ('edad', '30', '<');

edad < ’30’

y quitar las comillas simples

get_simp_set (‘edad’, ‘30’, ‘<’, false);

edad < 30

Como la intención de estas funciones es despreocuparnos del SQL y tener la certeza que nuestras consultas estarán bien realizadas al 100% para emplear más tiempo programando la lógica de nuestras aplicaciones, vamos a volver a realizar los ejemplos de SELECT mezclando estas funciones.

El primer ejemplo que hemos visto

get_select (‘clientes’, ‘*’, ‘nombre=’jose’)

Se puede intentar hacerlo más eficaz para evitar errores de la siguiente forma.:

```
$where = get_simp_set( ‘nombre’, ‘jose’ );
get_select (‘clientes’, ‘*’, $where);
```

produciendo el mismo resultado

SELECT * FROM clientes WHERE nombre=’jose’

Si nuestra consulta necesita recuperar varias columnas en vez de todas tendremos que hacer ...

```
$where = get_simp_set( ‘nombre’, ‘jose’ );
get_select (‘clientes’, ‘nombre, apellidos, telefono, url, direccion’, $where);
```

Pero como el objetivo es evitar cualquier posible error tenemos la siguiente función de apollo.

get_commas( )

Es una función con argumentos variables, le podemos pasar todos los que queramos y nos devoverá separados por comas cada unos de los parámetros. OJO, sólo hay que tener en cuenta que el primer parámetro deberá ser un booleano que indique si queremos que encierre los parámetros entre comillas simples. Veamos.:

get_commas (false, ‘nombre’, ‘apellidos’, ‘telefono’, ‘url’, ‘direccion’) nos devuelve

nombre, apellidos, telefono, url, dirección

si queremos que nos los devuelva entre comillas simples haremos

get_commas (true, ‘nombre’, ‘apellidos’, ‘telefono’, ‘url’, ‘direccion’)

'nombre', 'apellidos', 'telefono', 'url', 'dirección' que nos será muy útil cuando hagamos alguna INSERT.

También tenemos una versión mejorada de esta función

function get_commasA($arr_in, $comillas=true)

que realiza las mismas funciones pero si tenemos los datos almacenados un array

INSERT

Las insert siempre son un coñazo (perdón por la expresión) a la hora de montar una sentencia SQL y poner las asignaciones, comillas etc... y que funcione a la primera. Con la siguiente función podremos modificar nuestro código fuente cuanto queramos, añadir valores a insertar, quitar etc... teniendo la seguridad de que nunca nos dará error.

get_insert($table, $columns, $values)

Lo vemos con ejemplos:

$cols = get_commas(false, 'nombre', 'empresa'); //- false para que no meta comillas
$vals = get_commas(true, 'jose', 'Distintiva Solutions')
get_insert('clientes', $cols, $vals) nos devolverá:

INSERT INTO clientes (nombre, empresa) VALUES ('jose', 'Distintiva Solutions');

Y como hemos visto antes si uno de los valores no tiene que estar entre comillas podemos usar :

$cols = get_commas(false, 'nombre', 'fecha_alta');
$vals = get_commas(true, 'jose', '!!now( )')
get_insert('clientes', $cols, $vals) nos devolverá:

INSERT INTO clientes (nombre, fecha_alta) VALUES ('jose', now( ));

Y ya nos pueden decir que aumentemos el formulario de alta en todos los campos que quieran porque a nosotros no nos costará nada modificar los arrays sabiendo que todo saldrá a la perfección.

Si los datos a insertar vienen en un array recordar que podremos usar la función get_commasA($mi_array, true);

UPDATE

Otra facilidad es la de realizar un update sin equivocarnos en un solo SET...
Vamos a ver unos ejemplos que es la forma más fácil de comprenderlo...
function get_update($table, $values, $where)

La función es muy simple y hace uso de las anteriormente citadas.

```
$a_c=array(‘nombre’, ‘empresa’);
$a_v=array('Jose', ‘Distintiva’ );

$sql=get_update(‘clientes’,  get_mult_set($a_c, $a_v) , get_simp_set('id', 125));
```

El resultado será:

```
UPDATE clientes SET nombre=’Jose’, empresa=’Distintiva’ WHERE id=’125’
```

Y cualquier modificación sólo afectaría a los arrays de columnas y valores que se componen mediante el get_mult_set y la condición con el get_simp_set.

Con las explicaciones que he realizado anteriormente del get_mul_set etc... también son aplicables en este caso.

Y seguimos facilitando las cosas, ahora con una función mucho más sencilla...

DELETE

get_delete($table, $where='')

Es muy simple, sólo hay que pasarle la tabla y la condición de borrado con un get_simp_set

Para más complicaciones... a veces tenemos que updatear una tabla con valores de otra.

Pues a no preocuparse porque también está disponible pero sólo funciona con MySQL v 4.xx o superior

MAS

get_update_join($table_a, $table_b, $id_a, $id_b, $values, $where='')

Supongamos que tenemos 2 tablas con una relación mediante ID, si queremos actualizar los datos de la primera con valores de la segunda que cumplan tal condición...

```
$a_c=array("a.precios");
$a_v=array("!!b.coste " );

function get_update_join(‘productos’, ‘divisas’, ‘id’, ‘id_prod’, get_mult_set($a_c, $a_v),  get_simp_set(‘a.id’, ‘25’))
```

De esto resulta...

```
UPDATE productos a, divisas b  SET a.precios=b.coste WHERE a.id=b.id_prod AND ( a.id=’25’)
```

Esta función updatea productos de la tabla productos con valores que pueden cambiar de la tabla divisas y que cada producto se relaciona con la divisa por el id y quiero updatear sólo los que su ID sea=25

Y por último una select entre 2 tablas relacionadas por un mismo índice se puede sacar su sentencia SQL mediante la función

function get_select_join($table_a, $table_b, $id_a, $id_b, $columns, $where='', $order='')

que sin más explicación funciona como la anterior...