*Una sentencia preparada (Prepared Statements ) o sentencia parametrizada es una plantilla de un query o consulta que queremos ejecutar una gran cantidad de veces.*

* *Ahorra tiempo de procesamiento ya que la consulta se analiza una sola vez y después solo se envían los parámetros.*
* *Minimizan el riesgo de ataques de inyección SQL*

<?php

$conexion = **new** mysqli('localhost', 'root', '', 'dwes');

$sentencia = $conexion->stmt\_init();

$sentencia->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');

$cod\_producto = "nuevo";

$nombre\_producto = "nuevo PC";

$sentencia->bind\_param('ss', $cod\_producto, $nombre\_producto);

$sentencia->execute();

$sentencia->close();

$conexion->close();

?>

Inicializa una sentencia y devuelve un objeto para usarlo con mysqli\_stmt\_prepare:

$sentencia = $conexion->stmt\_init();

Las sentencias preparadas admiten parámetros. Para preparar una sentencia con parámetros, en lugar de poner los valores debes indicar con un signo de interrogación su posición dentro de la sentencia SQL.

$sentencia->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');

Y antes de ejecutar la consulta tienes que utilizar el método **bind\_param**  para sustituir cada parámetro por su valor. El primer parámetro del método bind\_param es una cadena de texto en la que cada carácter indica el tipo de un parámetro, según la siguiente tabla.

| Caracteres indicativos del tipo de los parámetros en una consulta preparada. | |
| --- | --- |
| **Carácter.** | **Tipo del parámetro.** |
| **I.** | Número entero. |
| **D.** | Número real (doble precisión). |
| **S.** | Cadena de texto. |
| **B.** | Contenido en formato binario (BLOB). |

$sentencia->bind\_param('ss', $cod\_producto, $nombre\_producto);

Cuando uses bind\_param para enlazar los parámetros de una consulta preparada con sus respectivos valores, deberás usar siempre variables como en el ejemplo anterior. Si intentas utilizar literales, por ejemplo:

$sentencia->bind\_param('ss', 'nuevo', 'nuevo PC'); // Genera un error

Obtendrás un error. El motivo es que los parámetros del método bind\_param se pasan por referencia.

El método bind\_param permite tener una consulta preparada en el servidor MySQL y ejecutarla tantas veces como quieras cambiando ciertos valores cada vez.

Ejemplo de una consulta:

<?php

$conexion = **new** mysqli('localhost', 'root', '', 'dwes');

$sentencia = $conexion->stmt\_init();

$sentencia->prepare('SELECT producto, unidades FROM stock WHERE unidades<2');

$sentencia->execute();

$sentencia->bind\_result($producto, $unidades);

**while**($sentencia->fetch()) {

**print** "<p>Producto **$producto**: **$unidades** unidades.</p>";

}

$sentencia->close();

$conexion->close();

?>

Para asignar a variables los campos que se obtienen tras la ejecución:

$sentencia->bind\_result($producto, $unidades);

Utilizando el método **fetch**  se recorren los registros devueltos.

Ejemplo de consulta con parámetros:

<?php

$conexion = **new** mysqli('localhost', 'root', '', 'dwes');

$sentencia = $conexion->stmt\_init();

$sentencia->prepare('SELECT producto, unidades FROM stock WHERE unidades<?');

$cantidad=8;

$sentencia->bind\_param('i', $cantidad);

$sentencia->execute();

$sentencia->bind\_result($producto, $unidades);

**while**($sentencia->fetch()) {

**print** "<p>Producto **$producto**: **$unidades** unidades.</p>";

}

$sentencia->close();

$conexion->close();

?>

Ejercicio.

A partir de la página web obtenida en el ejercicio anterior, añade la opción de modificar el número de unidades del producto en cada una de las tiendas. Utiliza una consulta preparada para la actualización de registros en la tabla stock. No es necesario tener en cuenta las tareas de inserción (no existían unidades anteriormente) y borrado (si el número final de unidades es cero).