Programación de Servicios y Procesos

Bases de Datos y PHP. MySQL.

IES Nervión
Miguel A. Casado Alías

Introducción

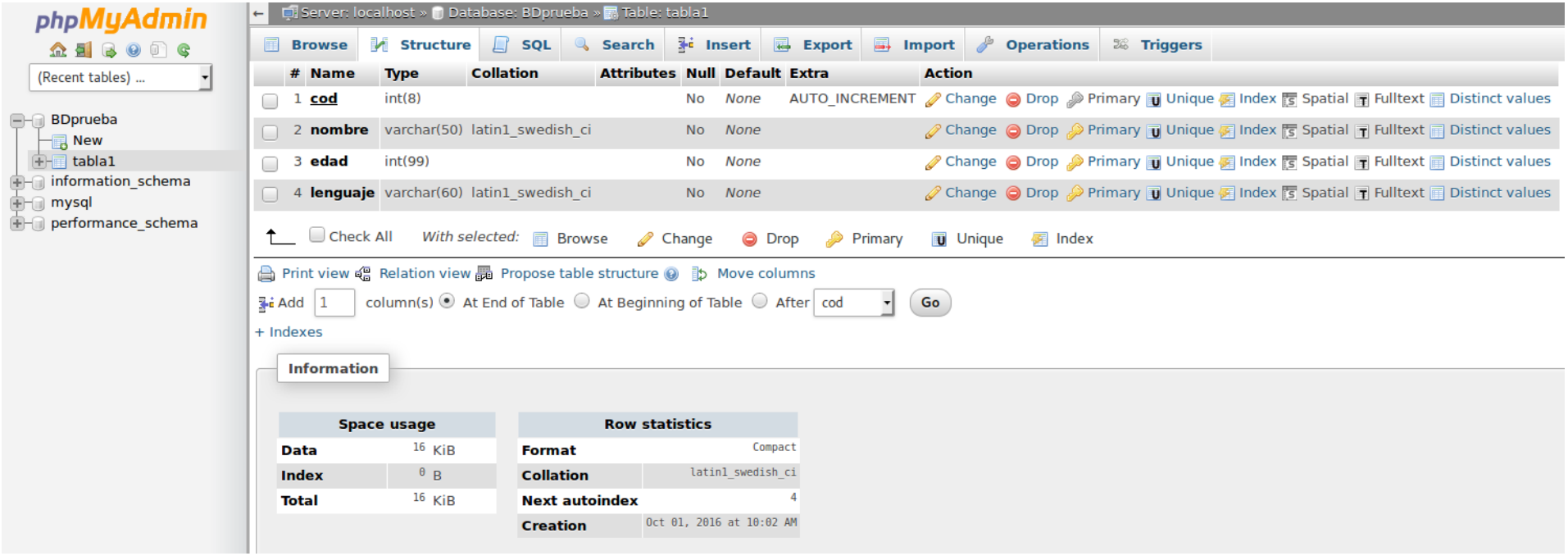
- PHP soporta múltiples SGBD (Oracle, SQLite, SQL Server, PostgreSQL…)
- Podemos usar extensiones de PHP específicas para cada SGBD o bien usar una extensión genérica: PDO.
- PDO nos proporciona uniformidad (usamos la misma sintáxis independientemente del SGBD)
- Las extensiones específicas nos ofrecen mayor potencia y rendimiento
- Nos centraremos en **MySQL** con la extensión **MySQLi**

MySQL

- Uno de los SGBD más populares del mundo
- Adquirido por Sun Microsystems en 2008, la cual fue comprada por Oracle en 2010
- Licencia dual: GPL / Comercial
- En 2009 se creó un fork: MariaDB, compatible casi al 100% con MySQL
- Soporta integridad referencial (Motor InnoDB)

phpMyAdmin

- Herramienta para administrar BB.DD. MySQL y MariaDB
- Proporciona una interfaz gráfica de fácil uso
- Escrita en PHP
- Software Libre (Licencia GPL)



MySQLi: Conexión con la BD

- Se debe crear una instancia de la clase mysqli, cuyo constructor puede recibir varios parámetros:
  - IP o nombre del servidor MySQL
  - Nombre de usuario
  - Contraseña
  - Nombre de la BD
  - Puerto
  - Socket

```
$conexion = new mysqli('localhost',‘juan','mipassword567','BDejemplo');
```

MySQLi: Control de error de conexión

- Si hay error al conectar, se verá reflejado en las siguientes propiedades de mysqli:
  - connect_errno: código de error
  - connect_error: mensaje de error
- Si no hay error, ambas propiedades valdrán null

```
$conexion = new mysqli('localhost', 'juan', 'mipassword000', 'Bdejemplo');

if ($conexion->connect_error) {
    trigger_error("Failed to connect to MySQL: " . $conexion->connect_error,
                  E_USER_ERROR);
}
```

MySQLi: Sentencias SQL

- Usaremos el método **query**
- Si ejecutamos una sentencia que no devuelve datos (p.ej: INSERT, UPDATE, DELETE…), el método query devolverá true si se ejecuta correctamente, y false si no.
- La propiedad affected_rows devuelve el n.º de filas afectadas
- La propiedad error devuelve el último error

```
$sql = "DELETE FROM MyGuests WHERE id=3";

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

MySQLi: Consultas

- Usaremos el método **query** para obtener un conjunto de resultados
- Sobre el conjunto de resultados, podemos aplicar:
  - fetch_array(): Por defecto, obtiene un registro y lo almacena en un array con claves numéricas y claves asociativas. Si le pasamos parámetros podemos cambiar cómo es el array:
    - MYSQLI_NUM: array con claves numéricas
    - MYSQLI_ASSOC: array asociativo
    - MYSQLI_BOTH: array asociativo y de clave numérica (default)
  - fetch_assoc(): Igual que fetch_array (MYSQLI_ASSOC)
  - fetch_row(): Igual que fetch_array (MYSQLI_NUM)
  - fetch_object(): Devuelve objeto en lugar de array. Los campos del registro serán propiedades del objeto devuelto.

MySQLi: Ejemplo de consulta

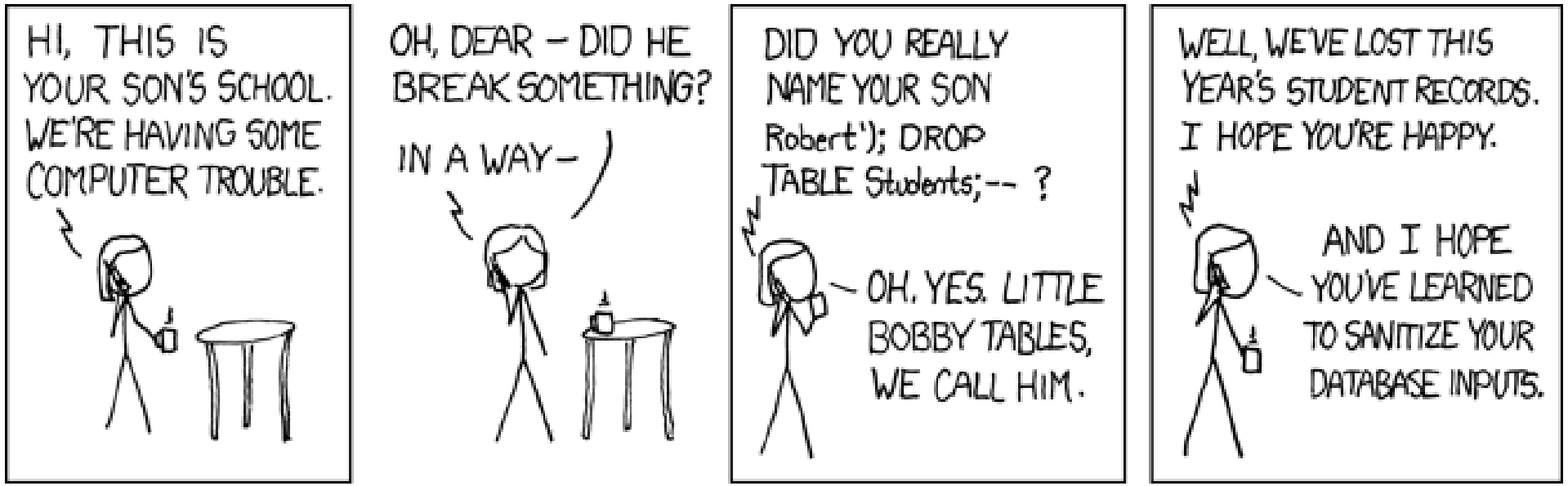
```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

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";
}
```

MySQLi: Cerrar conexiones

- Las conexiones se destruyen automáticamente al finalizar el script en el que se usan
- De todas formas, es conveniente cerrarlas expresamente cuando ya no sean necesarias para así liberar recursos y quizás mejorar el rendimiento
- Se usa el método **close**

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

SQL Injection



http://bobby-tables.com/about.html

https://diego.com.es/ataques-sql-injection-en-php

MySQLi: Sentencias preparadas

- También llamadas sentencias parametrizadas
- Permiten ejecutar la misma sentencia (o similar) varias veces con gran eficiencia
- Se lleva a cabo en dos etapas:
  - Preparación: Se envía una plantilla de sentencia al SGBD, que puede contener parámetros
  - Ejecución: Se asignan los valores de los parámetros y se envían al SGBD para que se ejecute la sentencia
- Utilizan menos recursos, ya que la sentencia solo se analiza una vez (en la fase de preparación)
- Son seguras frente a la inyección SQL

MySQLi: Ejemplo inserción con sentencia preparada

```php
// Preparamos la sentencia
$stmt = $db->prepare("INSERT INTO Clientes (nombre, ciudad, contacto) VALUES (?, ?, ?)");
// Vinculamos parámetros a variables
$stmt->bind_param('ssi', $nombre, $ciudad, $contacto);
// Establecemos los parámetros y ejecutamos
$nombre = "Impresiones Ulloa SL";
$ciudad = "Valencia";
$contacto = 3342;
$stmt->execute();
// Establecemos los parámetros y ejecutamos
$nombre = "PHP Transportes SA";
$ciudad = "Cádiz";
$contacto = 7617;
$stmt->execute();
```

Carácter	Tipo de Dato
i	Entero
d	Doble
s	Cadena
b	Blob

Primer parámetro de bind_param

MySQLi: Ejemplo consulta con sentencia preparada

```php
$stmt = $dbConnection->prepare('SELECT * FROM
                usuarios WHERE nombre = ?');
$stmt->bind_param('s', $nombre);
$stmt->execute();
$result = $stmt->get_result();
while ($row = $result->fetch_assoc()){
  // Hacer algo con $row
}
```

MySQLi: Ejemplo bind_result y fetch

```php
if ($stmt = $conn->prepare("SELECT nombre,
                            ciudad FROM Clientes")) {
    $stmt->execute();
    // Vinculamos variables a columnas
    $stmt->bind_result($nombre, $ciudad);
    // Obtenemos los valores
    while ($stmt->fetch()) {
        printf("%s %s\n", $nombre, $ciudad);
    }
    // Cerramos la sentencia preparada
    $stmt->close();
}
```

MySQLi: bind_result() vs get_result()

- bind_result ()
  - Es más fácil de usar
  - No funciona con sentencias SQL que usan * (SELECT *…)
- get_result ()
  - Es más enrevesada de usar (fetch_assoc(), row[‘col1’]…)
  - Funciona con todo tipo de sentencias SQL
  - Necesita el driver mysqlnd en el servidor
- bind_result() vs get_result() at stackoverflow.com