Ejecutar Sentencias

Para ejecutar sentencias de definición y/o manipulación de datos en el gestor MySQL se puede usar las funciones o métodos:

mysqli_query(orden) **mysqli::query**(orden)	Es la más usada. Ejecuta la orden, recibida como parámetro, y almacena el conjunto de resultados en un buffer (lo que permite al servidor liberar recursos asociados con los resultados de la sentencia tan pronto como sea posible) .

mysqli_real_query(orden) **mysqli::real_query**(orden)	Ejecuta la orden, recibida como parámetro, pero no almacena el conjunto de resultados en un buffer

```php
<?php
 $Id = new mysqli("127.0.0.1", "u1", "key1", "mibd");
 if ($Id  ->connect_errno) {
   echo "Falló conexión a MySQL: ("  .  $Id ->connect_errno . ") " . $Id ->connect_error;
   }

 if (  ! $Id ->query( "DROP TABLE IF EXISTS test")   ||    ! $Id ->query("CREATE TABLE test(n INT)") ||
      ! $Id ->query(  "INSERT INTO test  VALUES (1), (2), (3)")  )  {
           echo "Falló la creación de la tabla: (" . $Id->errno . ") " . $Id->error;
    }
// Ejecuta sentencia  almacenando resultados en buffer
  $result = $Id->query( "SELECT  n  FROM  test  ORDER BY  n  DESC" );

  echo "Muestra conjunto de resultados...<br>";
  $result->data_seek(0);
  while ($fila = $result->fetch_assoc()) {
      echo " n = " . $fila['n'] . "<br>";
   }
// Ejecuta sentencia  NO almacenando resultados en buffer
  $Id->real_query("SELECT n  FROM  test  ORDER BY  n  ASC");
  $result = $Id->store_result();

  echo "Muestra datos en orden  ascendente...<br>";
  while ($fila = $result->fetch_assoc()) {
      echo " n = " . $fila['n'] . "<br>";
    }

?>
```

Sentencias multiples

MySQL admite el envio de varias ordenes para su ejecución, con ello se reduce los viajes de ida y vuelta entre cliente y servidor y por tanto el tiempo de la aplicación.

Las sentencias múltiples o multiconsultas se ejecutan con la función mysqli_multi_query() o con el método mysqli::multi_query()

En la cadena de ordenes que se le pasa como argumento, cada una de ellas debe terminar en punto y coma.

La cadena de ordenes puede incluir unas que generen conjuntos de resultados con otras que no lo generen.

Principales Métodos y Funciones para tratar sentencias multiples

Métodos	Funciones	Funcionalidad
mysqli::**multi_query**(orden)	**mysqli_multi_query**(orden)	Ejecuta una consulta multiple en la BD
mysqli::**store_result**( )	**mysqli_store_result()**	Obtiene un cjto de resultados de la consulta
mysqli::**more_results**( )	**mysqli_more_results()**	Comprueba sí hay más resultados en multi-query
mysqli::**next_result**( )	**mysqli_next_result()**	Apunta a próximo resultado en multi-query

```php
<?php
 $Id = new mysqli("127.0.0.1", "u1", "key1", "mibd");
 if ($Id ->connect_errno) {
   echo "Falló conexión a MySQL: ("  .  $Id ->connect_errno . ") " . $Id ->connect_error;
   }

 if (  ! $Id ->query( "DROP TABLE IF EXISTS test")   ||     ! $Id ->query("CREATE TABLE test(n INT)") ||
      ! $Id ->query(  "INSERT INTO test  VALUES (1), (2), (3)")  )   {
           echo "Falló la creación de la tabla: (" . $Id->errno . ") " . $Id->error;
   }
// Guarda en $orden las  sentencias a ejecutar
  $orden = " SELECT COUNT(*) as Tfilas FROM test ;  INSERT INTO test  VALUES (5) ; " ;
  $orden =  $orden   . " SELECT COUNT(*)  as Tfilas FROM test ; ";
  if ( !$Id->multi_query( $orden ) ) {                  //  Ejecuta  el conjunto de ordenes  de $orden
     echo "Falló la multiconsulta: (" . $Id->errno . ") " . $Id->error;
    }
  do {                                                  // repetir
      if ( $result =  $Id->store_result() ) {           //  Obtener resultados de una orden y guardalos  en $result
        while ( $fila = $result->fetch_assoc() ) {      //   Extraer fila de $result y la guarda en $fila
            echo " Total = " . $fila['Tfilas'] . "<br>";  }     //  Mostrar contenido de fila extraida
        $result->free();                                //  liberar  recursos
       }
    }  while ($Id->more_results() && $Id->next_result() );  // mientras haya resultados
  $Id->close();                                         // cierra conexión
?>
```

El resultado del ejemplo sería: Total = 3
Total = 4

mysqli_query() solo admite una orden SQL para ordenar su ejecución; sí se envía varias ordenes su ejecución da error

```php
$Id   =  new mysqli("127.0.0.1", "u1", "key1", "mibd");
$result = $Id->query("SELECT 1;  DROP TABLE  tarifas;");
if (!$result) {
   echo "Error : (" . $mysqli->errno . ") " . $mysqli->error;
}
```

Sentencias parametrizadas (o preparadas)

Una sentencia parametrizada se usa para ejecutar una orden repetidamente; Se realiza una sola vez el análisis de la sentencia y no cada vez que se ejecuta.

La ejecución de este tipo de ordenes se hace en dos etapas:

etapa de preparación: se envía una plantilla de la sentencia al servidor de BD que analiza su sintaxis e inicializa los recursos del servidor para su uso posterior.

etapa de ejecución: se vincula los valores de los parámetros con valores de variables y se envía al servidor que crea una orden desde la plantilla de la sentencia y los valores vinculados para ejecutarla usando los recursos internos previamente creados.

Los parámetros se denotan con el símbolo **?**.

Principales Métodos y Funciones para tratar sentencias parametrizadas

Métodos	Funciones	Funcionalidad
mysqli::**prepare**(orden)	**mysqli_prepare**(orden)	Envia plantilla de orden al server para su análisis
mysqli_stmt::**bind_param**( )	**mysqli_stmt_bind_param()**	Vincula parametros con valores
mysqli_stmt::**execute**( )	**mysqli_stmt_execute()**	Ejecuta la sentencia
mysqli_stmt::**bind_result**( )	**mysqli_stmt_bind_result()**	Vincula resultados a variables
mysqli_stmt::**get_result**( )	**mysqli_stmt_get_result()**	Obtiene resultados de una orden

La vinculación de parametros y valores se hace por posición y es necesario indicar tipo de dato de cada parametro y valor . Los tipos se indican en una cadena que tendrá tantos caracteres como parametros haya en la sentencia preparada. Los caracteres que pueden formar parte de esta cadena son:

Caracter	Tipo de valor
i	Entero
d	Doble
s	Cadena
b	Blob

mysqli_stmt::bind_param (string $tipos , mixed &$v1 [,mixed &$v2 ...])
mysqli_stmt_bind_param ($orden , string $tipos , mixed &$var1 [,mixed &$var2 ...])

```php
<?php
 $Id = new mysqli("127.0.0.1", "u1", "key1", "mibd");
 if ($Id ->connect_errno) {  echo "Falló conexión a MySQL: " . $Id ->connect_errno . " - " . $Id ->connect_error ; exit;  }

 if ( ! $Id ->query( "DROP TABLE IF EXISTS T1") || ! $Id ->query("CREATE TABLE T1(n INT, ciudad varchar(50))") )
      {  echo "Falló creación de la tabla: " . $Id->errno . " - " . $Id->error ;  }
//  Sentencia parametrizada:  etapa 1→ preparación
 if ( !( $orden = $Id->prepare( "INSERT INTO T1 VALUES ( ? , ? )" ) ) )
    {  echo "Falló la preparación: " . $Id>errno . " - " . $Id->error;     }
//  Sentencia parametrizada:  etapa 2→ vinculación y ejecución
 $y = array( 'Roma' , 'Paris', 'Berlin', 'Soria', 'Lugo');
  $x = 0;  $z= $y[$x];
 if ( ! $orden->bind_param( "is", $x , $z) )          //  Vincula parámetros con variables
  { echo "Falló la vinculación de parámetros: " . $orden->errno . "- " . $orden->error;   }
 if ( ! $orden->execute() ) {                          //  Ejecuta la orden
  { echo "Falló la ejecución: " . $orden->errno . "- " . $orden->error;   }
//  ejecución repetida de la orden, solo se transfiere datos de cliente a servidor
  for ($x = 1; $x < 5; $x++) {
    if ( ! $orden->execute() ) {  echo "Falló la ejecución: " . $orden->errno . "- " . $orden->error;    }
   }
 $orden->close();            //  libera recursos de sentencia parametrizada
 $Id->close();  // cierra conexión
?>
```

También pueden recuperarse los resultados de sentencias preparadas en variables de salida. Estas variables deben ser vinculadas después de ejecutar la sentencia y ha de haber una variable por cada columna del select

```php
<?php
 $Id = new mysqli("127.0.0.1", "u1", "key1", "mibd");
 if ($Id ->connect_errno) {    echo "Falló conexión a MySQL: "  .  $Id ->connect_errno . " -  " . $Id ->connect_error ; exit;    }

//  Sentencia parametrizada:
 if  (  !( $orden = $Id->prepare( "SELECT  ciudad  FROM T1  WHERE  n >  ? " ) ) )   // etapa 1→ preparación
   { echo "Falló la preparación: " . $Id>errno . "- " . $Id->error;      }

 $x = 2;

 $orden->bind_param( "i", $x ) ;

 $orden->execute() ) ;                          //  Ejecuta la orden

  $city       = NULL;

 $orden->bind_result($city));                 //  Establece vinculo de valores de filas con variables

  while ( $orden->fetch()) {
     printf("%s \n", $city  );                 //  Muestra  valores obtenidos
  }
  $orden->close();                             //  libera recursos de sentencia parametrizada


 if  (  !( $orden = $Id->prepare( "SELECT  *   FROM T1  WHERE  ciudad >  ? " ) ) )   // etapa 1→ preparación
   { echo "Falló la preparación: " . $Id>errno . "- " . $Id->error;      }

  $x = 'Avila';

 $orden->bind_param( "s", $x ) ;

 $orden->execute() ) ;                          //  Ejecuta la orden

  $result  = $orden->get_result() ;       //  Obtiene resultados

  for  ($n = ($result->num_rows - 1);  $n >= 0;  $n-- )   {
          $result->data_seek($n);
           var_dump($result->fetch_assoc() );
        }
   $result->close();                           //  libera recursos usados para almacenar resultados
  $orden->close();                             //  libera recursos de sentencia parametrizada

 $Id->close();  // cierra conexión


?>
```

Control de transacciones

```php
<?php
  $Id = new mysqli("127.0.0.1", "u1", "key1", "mibd");
  if ($Id  ->connect_errno) {
        echo "Falló conexión a MySQL: "  .  $Id ->connect_errno . " -  " . $Id ->connect_error ;
        exit;
    }

  $Id->autocommit(false);

   $Id->query("INSERT INTO test(n) VALUES (1)");


  $Id ->rollback();

   $Id ->query("INSERT INTO test(id) VALUES (2)");


   $Id ->commit();
?>
```