Подготавливаемые запросы

СУБД MySQL поддерживает подготавливаемые запросы. Подготавливаемые (или параметризованные) запросы используются для повышения эффективности, когда один запрос выполняется многократно, и безопасности кода.

Принцип работы

Выполнение подготавливаемого запроса проводится в два этапа: подготовка и исполнение. На этапе подготовки на сервер посылается шаблон запроса. Сервер выполняет синтаксическую проверку этого шаблона, строит план выполнения запроса и выделяет под него ресурсы.

MySQL сервер поддерживает неименованные, или позиционные, псевдопеременные *?*.

Каждый подготавливаемый запрос использует ресурсы сервера. Если запрос больше не нужен, его необходимо сразу закрыть. Если не сделать этого явно, запрос закроется сам, но только когда PHP освободит его дескриптор, как правило это происходит при выходе запроса из области видимости или при завершении работы скрипта.

Пример #1 Объектно-ориентированный стиль

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* проверка соединения */
if (mysqli_connect_errno()) {
    printf("Не удалось подключиться: %s\n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* создаем подготавливаемый запрос */
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {

    /* связываем параметры с метками */
    $stmt->bind_param("s", $city);

    /* запускаем запрос */
    $stmt->execute();

    /* связываем переменные с результатами запроса */
    $stmt->bind_result($district);

    /* получаем значения */
    $stmt->fetch();

    printf("%s находится в округе %s\n", $city, $district);

    /* закрываем запрос */
    $stmt->close();
}

/* закрываем соединение */
$mysqli->close();
?>
```

Пример #2 Процедурный стиль

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* проверка соединения */
if (mysqli_connect_errno()) {
    printf("Не удалось подключиться: %s\n", mysqli_connect_error());
    exit();
}
$city = "Amersfoort";
```

```
/* создаем подготавливаемый запрос */
if ($stmt = mysqli_prepare($link, "SELECT District FROM City WHERE Name=?")) {

    /* связываем параметры с метками */
    mysqli_stmt_bind_param($stmt, "s", $city);

    /* запускаем запрос */
    mysqli_stmt_execute($stmt);

    /* связываем переменные с результатами запроса */
    mysqli_stmt_bind_result($stmt, $district);

    /* получаем значения */
    mysqli_stmt_fetch($stmt);

    printf("%s находится в округе %s\n", $city, $district);

    /* закрываем запрос */
    mysqli_stmt_close($stmt);
}

/* закрываем соединение */
mysqli_close($link);
?>
```

Типы данных значений в результирующей таблице

Пример #3 Исходные типы данных

```
if (!$mysqli->query("CREATE TABLE test(id INT, label CHAR(1))") ||
    !$mysqli->query("INSERT INTO test(id, label) VALUES (1, 'a')")) {
    echo "Не удалось создать таблицу: (" . $mysqli->errno . ") " . $mysqli->error;
}

$stmt = $mysqli->prepare("SELECT id, label FROM test WHERE id = 1");
$stmt->execute();
$res = $stmt->get_result();
$row = $res->fetch_assoc();

printf("id = %s (%s)\n", $row['id'], gettype($row['id']));
printf("label = %s (%s)\n", $row['label'], gettype($row['label']));
```

Результат выполнения данного примера:

```
id = 1 (integer)
label = a (string)
```

Получение результатов запроса с привязкой переменных

Результаты из подготовленного запроса можно получить либо привязав выходные переменные, либо запросив объект mysqli_result.

Выходные параметры нужно привязывать после выполнения запроса. Каждому столбцу результирующей таблицы должна соответствовать ровно одна переменная.

Пример #4 Привязка переменных к результату запроса

```
if (!$mysqli->query("CREATE TABLE test(id INT, label CHAR(1))") ||
    !$mysqli->query("INSERT INTO test(id, label) VALUES (1, 'a')")) {
    echo "Не удалось создать таблицу: (" . $mysqli->errno . ") " . $mysqli->error;
}

if (!($stmt = $mysqli->prepare("SELECT id, label FROM test"))) {
    echo "Не удалось подготовить запрос: (" . $mysqli->errno . ") " . $mysqli-
>error;
}


if (!$stmt->execute()) {
    echo "Не удалось выполнить запрос: (" . $mysqli->errno . ") " . $mysqli-
>error;
```

```
}

$out_id    = NULL;
$out_label = NULL;
if (!$stmt->bind_result($out_id, $out_label)) {
    echo "Не удалось привязать выходные параметры: (" . $stmt-
>errno . ") " . $stmt->error;
}

while ($stmt->fetch()) {
    printf("id = %s (%s), label = %s (%s)\n", $out_id, gettype($out_id), $out_labe
l, gettype($out_label));
}
```

Результат выполнения данного примера:

```
id = 1 (integer), label = a (string)
```

Извлечение результатов запроса посредством mysqli_result интерфейса

Вместо использования привязки переменных к результатам запроса, результирующие таблицы можно извлекать средствами интерфейса mysqli_result. Функция mysqli_stmt_get_result() возвращает буферизованный результирующий набор строк.

Пример #5 Использование mysqli_result для выборки результатов запроса

```
if (!$mysqli->query("CREATE TABLE test(id INT, label CHAR(1))") ||
    !$mysqli->query("INSERT INTO test(id, label) VALUES (1, 'a')")) {
    echo "Не удалось создать таблицу: (" . $mysqli->errno . ") " . $mysqli->error;
}

if (!($stmt = $mysqli->prepare("SELECT id, label FROM test ORDER BY id ASC"))) {
    echo "Не удалось подготовить запрос: (" . $mysqli->errno . ") " . $mysqli-
>error;
}

if (!$stmt->execute()) {
     echo "Не удалось выполнить запрос: (" . $stmt->errno . ") " . $stmt->error;
}

if (!($res = $stmt->get_result())) {
    echo "Не удалось получить результат: (" . $stmt->errno . ") " . $stmt->error;
}

var_dump($res->fetch_all());
```

Результат выполнения данного примера:

```
array(1) {
  [0]=>
  array(2) {
    [0]=>
    int(1)
    [1]=>
    string(1) "a"
  }
}
```

Использование mysqli_result interface имеет дополнительное преимущество в том, что буферизация результирующих таблиц на клиенте предлагает гибкую систему навигации по этим таблицам.

Пример #6 Буферизация результирующего набора для удобства чтения данных

```
if (!$mysqli->query("CREATE TABLE test(id INT, label CHAR(1))") ||
    !$mysqli->query("INSERT INTO test(id, label)
VALUES (1, 'a'), (2, 'b'), (3, 'c')")) {
```

```
    echo "Не удалось создать таблицу: (" . $mysqli->errno . ") " . $mysqli->error;
}

if (!($stmt = $mysqli->prepare("SELECT id, label FROM test"))) {
    echo "Не удалось подготовить запрос: (" . $mysqli->errno . ") " . $mysqli-
>error;
}

if (!$stmt->execute()) {
     echo "Не удалось выполнить запрос: (" . $stmt->errno . ") " . $stmt->error;
}

if (!($res = $stmt->get_result())) {
    echo "Не удалось получить результат: (" . $stmt->errno . ") " . $stmt->error;
}

for ($row_no = ($res->num_rows - 1); $row_no >= 0; $row_no--) {
    $res->data_seek($row_no);
    var_dump($res->fetch_assoc());
}
$res->close();
```

Результат выполнения данного примера:

```
array(2) {
  ["id"]=>
  int(3)
  ["label"]=>
  string(1) "c"
}
array(2) {
  ["id"]=>
  int(2)
  ["label"]=>
  string(1) "b"
}
array(2) {
  ["id"]=>
  int(1)
  ["label"]=>
  string(1) "a"
}
```