

PHP Создать базу данных MySQL

Создадим с помощью блокнота на нашем сервере файл index.php

[index.php](#)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Create a MariaDB Database</title>
  </head>
  <body>

  </body>
</html>
```

Создание новой базы данных

Оператор **CREATE DATABASE** используется для создания базы данных в MySQL.

В следующих примерах создается база данных с именем «my_DB»: В тело нашей страницы между тегами <body> и </body> вставим следующий php код

[index.php](#)

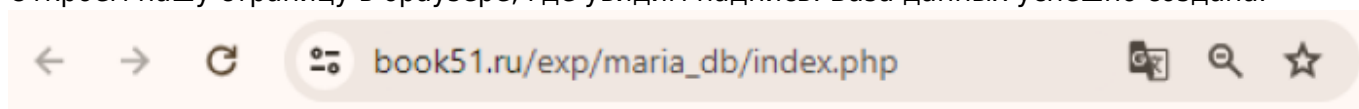
```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";

// Создаём соединение
$conn = new mysqli($servername, $username, $password);
// Проверим подключение
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Создадим базу данных my_DB
$sql = "CREATE DATABASE my_DB";
if ($conn->query($sql) === TRUE) {
    echo "База данных успешно создана";
} else {
```

```
echo "Ошибка создания базы данных: " . $conn->error;
}
// Закроем соединение
$conn->close();
?>
```

- **Примечание.** При создании новой базы данных необходимо указать только первые три аргумента объекта `mysqli` (имя сервера, имя пользователя и пароль).
- **Совет:** Если вам нужно использовать определенный порт, добавьте пустую строку в качестве аргумента имени базы данных, например: `new mysqli(«localhost», «username», «password», «», port)`

Откроем нашу страницу в браузере, где увидим надпись: База данных успешно создана.



База данных успешно создана

Проверим создание базы данных `my_DB` в MariaDB с помощью PhpMyAdmin

Сервер: MariaDB 10

Базы данных | SQL | Состояние | Учётные записи

Базы данных

[Создать базу данных](#)

Имя базы данных: utf8mb3_general_ci

☐ Отметить все [Удалить](#)

База данных	Сравнение	Действие
☐ forum	utf8mb3_general_ci	Проверить привилегии
☐ information_schema	utf8mb3_general_ci	Проверить привилегии
☐ mysql	utf8mb3_general_ci	Проверить привилегии
☐ my_DB	utf8mb3_general_ci	Проверить привилегии
☐ performance_schema	utf8mb3_general_ci	Проверить привилегии
☐ sys	utf8mb3_general_ci	Проверить привилегии

Всего: 6

Создание таблицы

Оператор CREATE TABLE используется для создания таблицы в MySQL.

Мы создадим таблицу с именем «MyGuests» с пятью столбцами: «id», «имя», «фамилия», «электронная почта» и «reg_date»: В тело нашей страницы между тегами <body> и </body> вставим следующий php код

[index.php](#)

```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";
$dbname = "my_DB";
// Создаём соединение
$conn = new mysqli($servername, $username, $password, $dbname);
// Проверим подключение
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Создадим таблицу базы данных
$sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";
if ($conn->query($sql) === TRUE) {
    echo "Таблица MyGuests успешно создана";
} else {
    echo "Ошибка создания таблицы: " . $conn->error;
}
// Закроем соединение
$conn->close();
?>
```

Примечания к таблице выше:

Тип данных указывает, какой тип данных может содержать столбец. Полную информацию обо всех доступных типах данных можно найти в нашем справочнике по типам данных .

После типа данных вы можете указать другие необязательные атрибуты для каждого столбца:

- NOT NULL — каждая строка должна содержать значение для этого столбца, значения NULL не допускаются.
- Значение DEFAULT — установите значение по умолчанию, которое добавляется, когда не

передается другое значение.

- UNSIGNED — используется для числовых типов, ограничивает сохраняемые данные положительными числами и нулем.
- AUTOУвеличение — MySQL автоматически увеличивает значение поля на 1 каждый раз, когда добавляется новая запись.
- ПЕРВИЧНЫЙ КЛЮЧ — используется для уникальной идентификации строк в таблице. Столбец с настройкой PRIMARY KEY часто представляет собой идентификационный номер и часто используется с AUTO_INCREMENT.

Каждая таблица должна иметь столбец первичного ключа (в данном случае столбец «id»). Его значение должно быть уникальным для каждой записи в таблице.

В следующих примерах показано, как создать таблицу в PHP:
Откроем нашу страницу в браузере, где увидим надпись: Таблица MyGuests успешно создана.

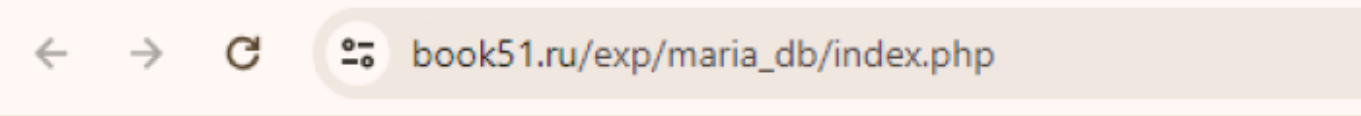


Таблица MyGuests успешно создана

Проверим создание таблицы MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin

phpMyAdmin

НедавнееИзбранное

Создать БД

forum

information_schema

mysql

my_DB

- Новая
- MyGuests
 - Индексы
 - Новый
 - PRIMARY
 - Столбцы
 - Новое
 - email (varchar, NULL, ...)
 - firstname (varchar)
 - id (PRI, int)
 - lastname (varchar)
 - reg_date (timestamp, ...)

performance_schema

sys

Сервер: MariaDB 10

Базы данныхSQLСостояниеУчётные записи

Базы данных

Создать базу данных

Имя базы данных

utf8mb3_general_ci

Отметить все

Удалить

База данных	Сравнение	Действие
☐ forum	utf8mb3_general_ci	Проверить привилегии
☐ information_schema	utf8mb3_general_ci	Проверить привилегии
☐ mysql	utf8mb3_general_ci	Проверить привилегии
☐ my_DB	utf8mb3_general_ci	Проверить привилегии
☐ performance_schema	utf8mb3_general_ci	Проверить привилегии
☐ sys	utf8mb3_general_ci	Проверить привилегии

Всего: 6

Вставка данных

После того, как база данных и таблица созданы, мы можем начать добавлять в них данные.

Вот несколько правил синтаксиса, которым следует следовать:

- SQL-запрос должен быть заключен в кавычки в PHP.
- Строковые значения внутри запроса SQL должны быть заключены в кавычки.
- Числовые значения не должны заключаться в кавычки
- Слово NULL не должно заключаться в кавычки

Оператор INSERT INTO используется для добавления новых записей в таблицу MySQL:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Чтобы узнать больше о SQL, посетите наш учебник по SQL .

В предыдущей главе мы создали пустую таблицу с именем «MyGuests» с пятью столбцами: «id», «имя», «фамилия», «электронная почта» и «reg_date». Теперь заполним таблицу данными.

Примечание. Если столбец имеет значение AUTO_INCREMENT (например, столбец «id») или TIMESTAMP с обновлением по умолчанию current_timestamp (например, столбец «reg_date»), его нет необходимости указывать в SQL-запросе; MySQL автоматически добавит это значение.

В следующих примерах в таблицу «MyGuests» добавляется новая запись. В тело нашей страницы между тегами <body> и </body> вставим следующий php код

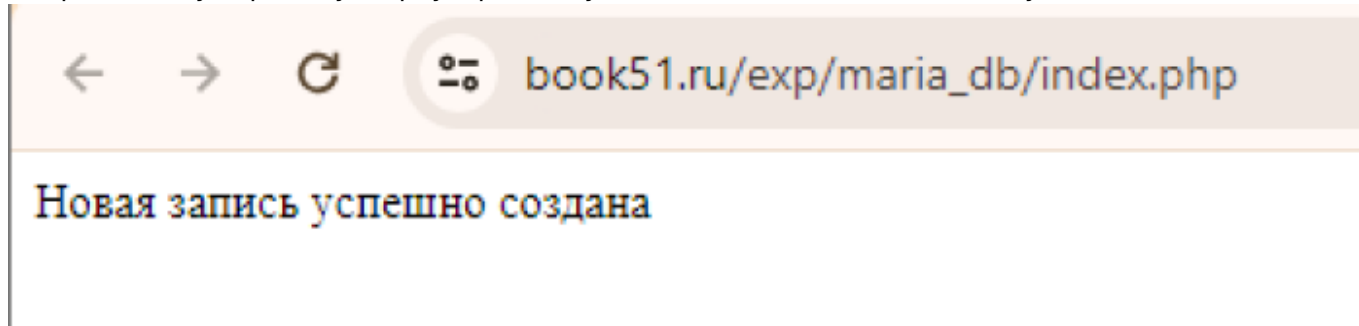
[index.php](#)

```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";
$dbname = "my_DB";
// Создаём соединение
$conn = new mysqli($servername, $username, $password, $dbname);
// Проверим подключение
if ($conn->connect_error) {
    die("Ошибка соединения: " . $conn->connect_error);
}
// Добавим новую запись
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

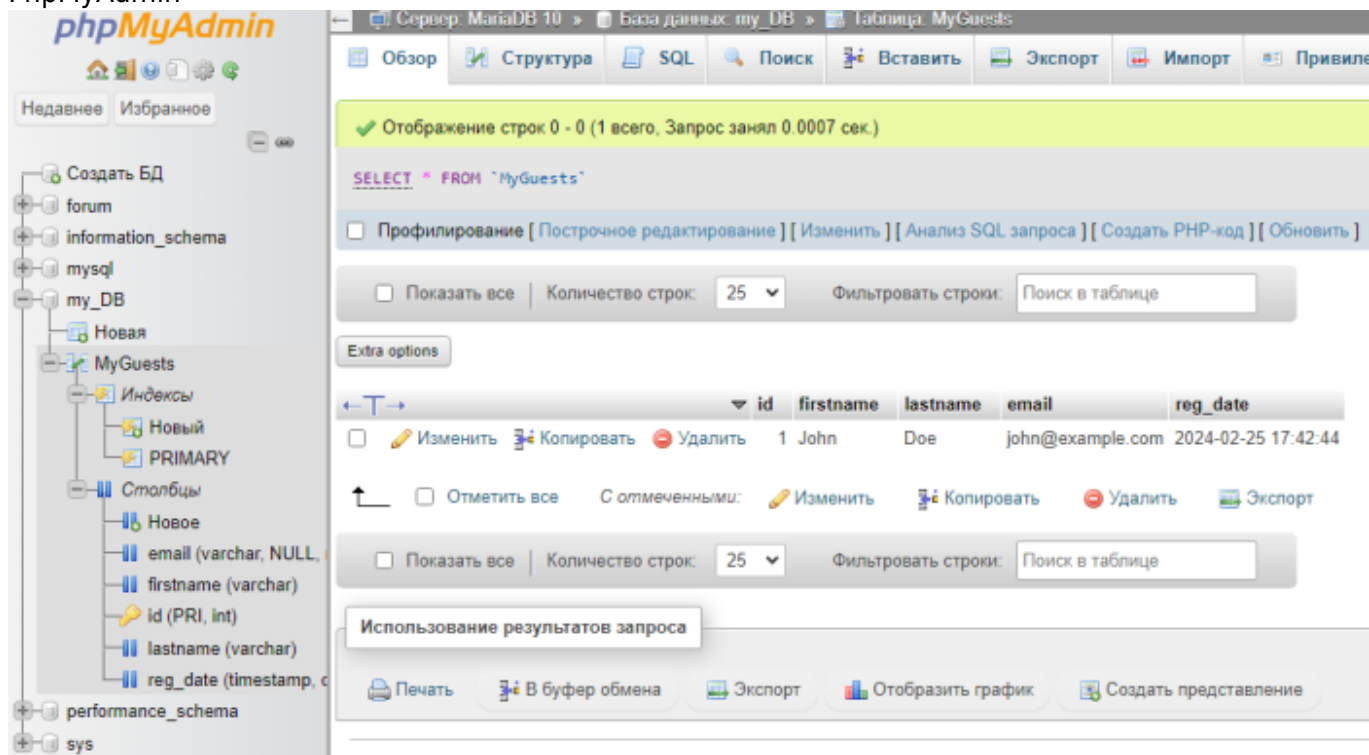
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

```
// Закроем соединение
$conn->close();
?>
```

Откроем нашу страницу в браузере, где увидим надпись: Новая запись успешно создана.



Проверим создание новой записи в таблице MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin



Получаем идентификатор последней записи

Если мы выполним INSERT или UPDATE для таблицы с полем AUTO_INCREMENT, мы сможем немедленно получить идентификатор последней вставленной/обновленной записи.

В таблице «MyGuests» столбец «id» представляет собой поле AUTO_INCREMENT:

```
CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
```

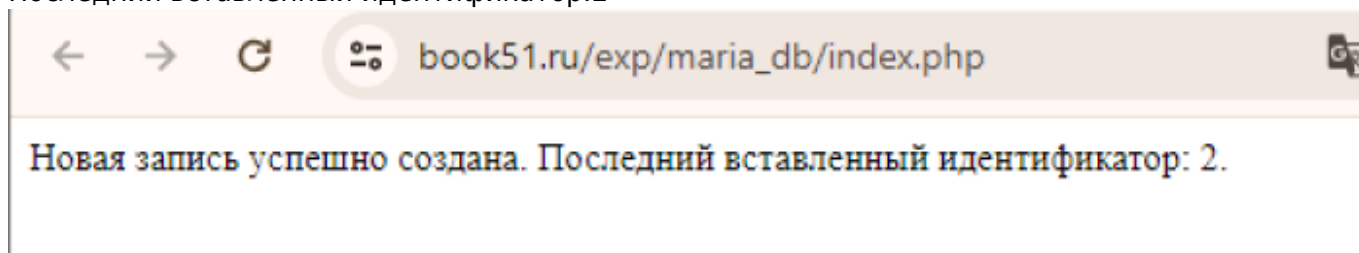
```
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
)
```

Следующий пример аналогичен предыдущему примеру (Вставка данных PHP в MySQL), за исключением того, что мы добавили одну строку кода для получения идентификатора последней вставленной записи. Мы также отображаем последний вставленный идентификатор: В тело нашей страницы между тегами <body> и </body> вставим следующий php код

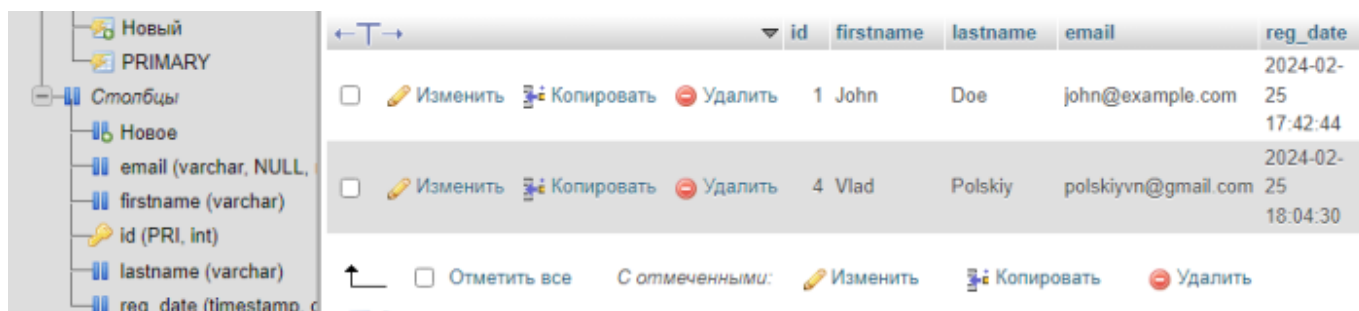
[index.php](#)

```
<?php  
$servername = "localhost";  
$username = "root";  
$password = "*****";  
$dbname = "my_DB";  
// Создаём соединение  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Проверим подключение  
if ($conn->connect_error) {  
    die("Ошибка соединения: " . $conn->connect_error);  
}  
// Добавим новую запись  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if ($conn->query($sql) === TRUE) {  
    echo "New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
// Закроем соединение  
$conn->close();  
?>
```

Откроем нашу страницу в браузере, где увидим надпись: Новая запись успешно создана. Последний вставленный идентификатор:2



Проверим создание новой записи в таблице MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin



The screenshot shows a database management interface. On the left, a tree view displays the database structure: 'Новый' (New) with a 'PRIMARY' key, and a list of columns: 'email (varchar, NULL)', 'firstname (varchar)', 'id (PRI, int)', 'lastname (varchar)', and 'reg_date (timestamp, c)'. The main area shows a table with columns: 'id', 'firstname', 'lastname', 'email', and 'reg_date'. The table contains two rows of data. The first row has id 1, firstname 'John', lastname 'Doe', email 'john@example.com', and reg_date '2024-02-25 17:42:44'. The second row has id 4, firstname 'Vlad', lastname 'Polskiy', email 'polskiyvn@gmail.com', and reg_date '2024-02-25 18:04:30'. Below the table, there are buttons for 'Изменить' (Edit), 'Копировать' (Copy), and 'Удалить' (Delete) for each row, and a 'Отметить все' (Mark all) button at the bottom.

	id	firstname	lastname	email	reg_date
☐ Изменить ☐ Копировать ☐ Удалить	1	John	Doe	john@example.com	2024-02-25 17:42:44
☐ Изменить ☐ Копировать ☐ Удалить	4	Vlad	Polskiy	polskiyvn@gmail.com	2024-02-25 18:04:30

↑ ☐ Отметить все С отмеченными: ☐ Изменить ☐ Копировать ☐ Удалить

Вставка нескольких записей

С помощью функции необходимо выполнить несколько операторов SQL `mysqli_multi_query()`.

В следующих примерах в таблицу «MyGuests» добавляются три новые записи: В тело нашей страницы между тегами `<body>` и `</body>` вставим следующий php код

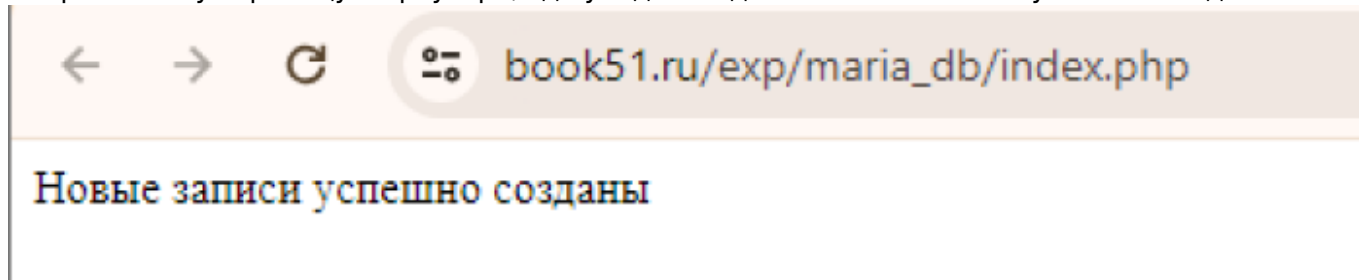
[index.php](#)

```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";
$dbname = "my_DB";
// Создаём соединение
$conn = new mysqli($servername, $username, $password, $dbname);
// Проверим подключение
if ($conn->connect_error) {
    die("Ошибка соединения: " . $conn->connect_error);
}
// Добавим несколько новую запись
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

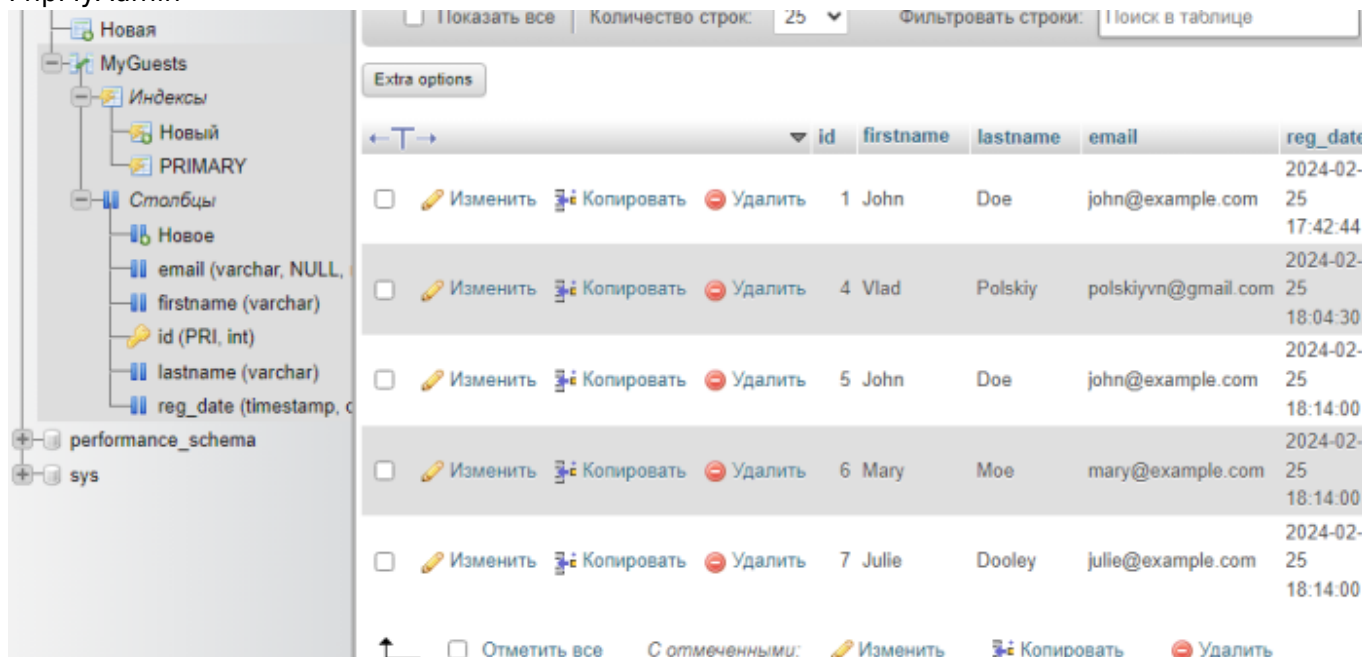
if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
// Закроем соединение
$conn->close();
?>
```

Обратите внимание, что каждый оператор SQL должен быть разделен точкой с запятой.

Откроем нашу страницу в браузере, где увидим надпись: Новые записи успешно созданы



Проверим создание новой записи в таблице MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin



Подготовленные операторы и связанные параметры

Подготовленный оператор — это функция, используемая для многократного выполнения одних и тех же (или аналогичных) операторов SQL с высокой эффективностью.

Подготовленные операторы в основном работают следующим образом:

1. Подготовка. Шаблон инструкции SQL создается и отправляется в базу данных. Определенные значения остаются неуказанными и называются параметрами (помечены знаком «?»). Пример: INSERT INTO MyGuests VALUES(?, ?, ?)
2. База данных анализирует, компилирует и выполняет оптимизацию запросов по шаблону инструкции SQL и сохраняет результат, не выполняя его.
3. Выполнение: позже приложение привязывает значения к параметрам, и база данных выполняет оператор. Приложение может выполнить оператор столько раз, сколько захочет, с разными значениями.

По сравнению с непосредственным выполнением операторов SQL подготовленные операторы имеют три основных преимущества:

- Подготовленные операторы сокращают время анализа, поскольку подготовка запроса выполняется только один раз (хотя оператор выполняется несколько раз).
- Связанные параметры минимизируют пропускную способность сервера, поскольку вам

нужно каждый раз отправлять только параметры, а не весь запрос.

- Подготовленные операторы очень полезны против SQL-инъекций, поскольку значения параметров, которые передаются позже с использованием другого протокола, не требуют правильного экранирования. Если исходный шаблон инструкции не получен из внешних входных данных, SQL-инъекция не может произойти.

В следующем примере используются подготовленные операторы и связанные параметры в MySQLi: В тело нашей страницы между тегами <body> и </body> вставим следующий php код

[index.php](#)

```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";
$dbname = "my_DB";
// Создаём соединение
$conn = new mysqli($servername, $username, $password, $dbname);
// Проверим подключение
if ($conn->connect_error) {
    die("Ошибка соединения: " . $conn->connect_error);
}
// подготовить и связать
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// задайте параметры и выполните
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
// Закроем соединение
$conn->close();
?>
```

Строки кода для пояснения из примера выше:

```
"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
```

В нашем SQL мы вставляем вопросительный знак (?) туда, где хотим заменить целое, строковое, двойное или BLOB-значение.

Затем взгляните на функцию `bind_param()`:

```
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

Эта функция привязывает параметры к SQL-запросу и сообщает базе данных, что это за параметры. Аргумент «sss» перечисляет типы данных, которыми являются параметры. Символ s сообщает MySQL, что параметр является строкой.

Аргумент может быть одного из четырех типов:

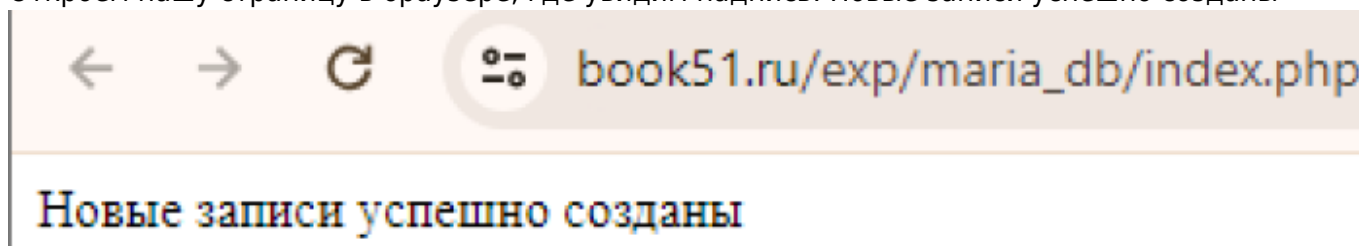
- i - целое число
- d - двойной
- s - строка
- b - БЛОБ

У нас должно быть по одному из них для каждого параметра.

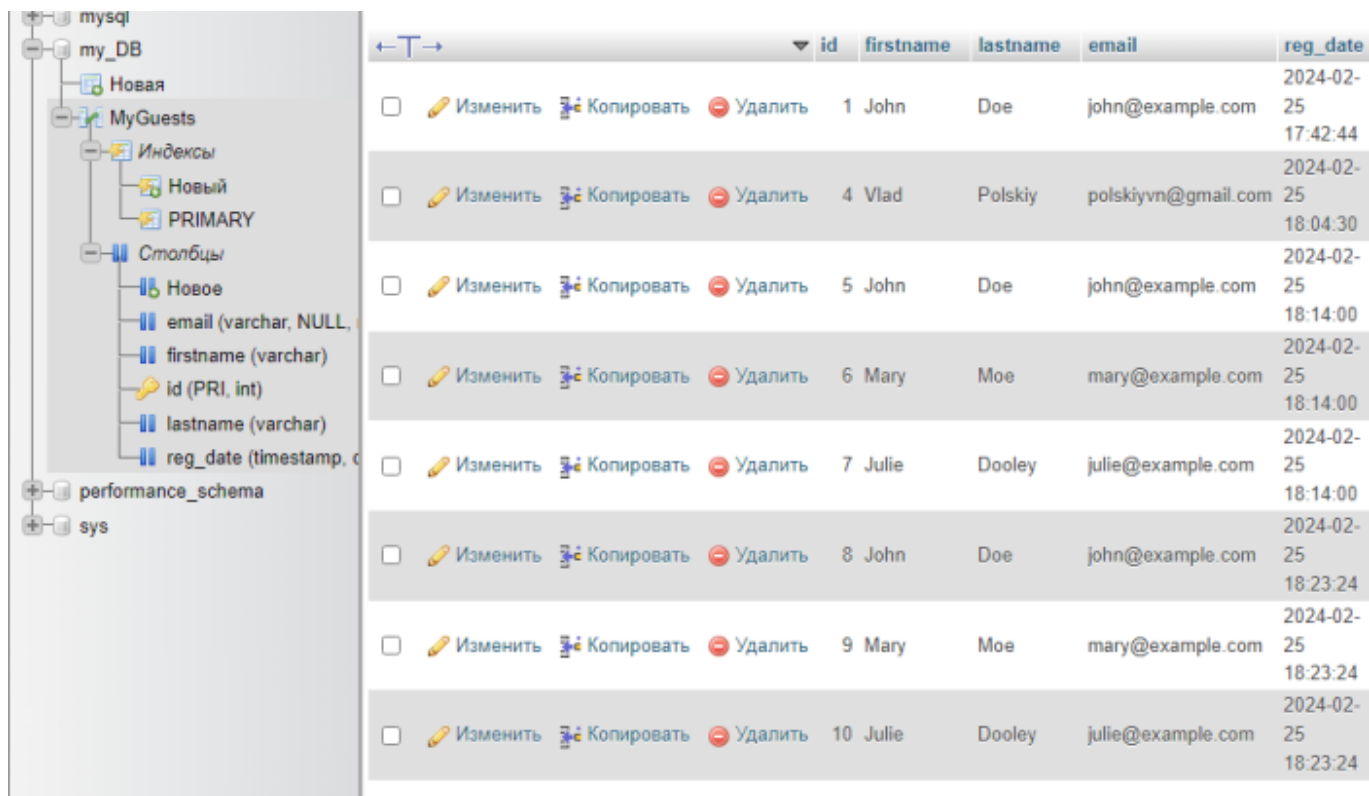
Сообщая mysql, какой тип данных следует ожидать, мы минимизируем риск SQL-инъекций.

Примечание. Если мы хотим вставить какие-либо данные из внешних источников (например, пользовательский ввод), очень важно, чтобы данные были очищены и проверены.

Откроем нашу страницу в браузере, где увидим надпись: Новые записи успешно созданы



Проверим создание новой записи в таблице MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin



The screenshot shows a database management interface. On the left, a tree view displays the database structure: 'mysql' contains 'my_DB', which contains 'Новая' (New), 'MyGuests', 'Индексы' (Indexes), 'Новый' (New), 'PRIMARY', 'Столбцы' (Columns), 'Новое' (New), 'email (varchar, NULL)', 'firstname (varchar)', 'id (PRI, int)', 'lastname (varchar)', and 'reg_date (timestamp, c)'. Below this, 'performance_schema' and 'sys' are also listed. On the right, a table view shows the data for the 'MyGuests' table. The table has columns: 'id', 'firstname', 'lastname', 'email', and 'reg_date'. The data is as follows:

	id	firstname	lastname	email	reg_date
☐	1	John	Doe	john@example.com	2024-02-25 17:42:44
☐	4	Vlad	Polskiy	polskiyvn@gmail.com	2024-02-25 18:04:30
☐	5	John	Doe	john@example.com	2024-02-25 18:14:00
☐	6	Mary	Moe	mary@example.com	2024-02-25 18:14:00
☐	7	Julie	Dooley	julie@example.com	2024-02-25 18:14:00
☐	8	John	Doe	john@example.com	2024-02-25 18:23:24
☐	9	Mary	Moe	mary@example.com	2024-02-25 18:23:24
☐	10	Julie	Dooley	julie@example.com	2024-02-25 18:23:24

Подготовленные операторы и связанные параметры

Подготовленный оператор — это функция, используемая для многократного выполнения одних и тех же (или аналогичных) операторов SQL с высокой эффективностью.

Подготовленные операторы в основном работают следующим образом:

1. Подготовка. Шаблон инструкции SQL создается и отправляется в базу данных. Определенные значения остаются неуказанными и называются параметрами (помечены знаком «?»). Пример: `INSERT INTO MyGuests VALUES(?, ?, ?)`
2. База данных анализирует, компилирует и выполняет оптимизацию запросов по шаблону инструкции SQL и сохраняет результат, не выполняя его.
3. Выполнение: позже приложение привязывает значения к параметрам, и база данных выполняет оператор. Приложение может выполнить оператор столько раз, сколько захочет, с разными значениями.

По сравнению с непосредственным выполнением операторов SQL подготовленные операторы имеют три основных преимущества:

- Подготовленные операторы сокращают время анализа, поскольку подготовка запроса выполняется только один раз (хотя оператор выполняется несколько раз).
- Связанные параметры минимизируют пропускную способность сервера, поскольку вам нужно каждый раз отправлять только параметры, а не весь запрос.
- Подготовленные операторы очень полезны против SQL-инъекций, поскольку значения параметров, которые передаются позже с использованием другого протокола, не требуют правильного экранирования. Если исходный шаблон инструкции не получен из внешних входных данных, SQL-инъекция не может произойти.

В следующем примере используются подготовленные операторы и связанные параметры в MySQLi: В тело нашей страницы между тегами <body> и </body> вставим следующий php код

[index.php](#)

```
<?php
$servername = "localhost";
$username = "root";
$password = "*****";
$dbname = "my_DB";
// Создаём соединение
$conn = new mysqli($servername, $username, $password, $dbname);
// Проверим подключение
if ($conn->connect_error) {
    die("Ошибка соединения: " . $conn->connect_error);
}
// подготовить и связать
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname,
email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// задайте параметры и выполните
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
// Закроем соединение
$conn->close();
?>
```

Строки кода для пояснения из примера выше:

```
"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
```

В нашем SQL мы вставляем вопросительный знак (?) туда, где хотим заменить целое, строковое, двойное или BLOB-значение.

Затем взгляните на функцию `bind_param()`:

```
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

Эта функция привязывает параметры к SQL-запросу и сообщает базе данных, что это за параметры. Аргумент «sss» перечисляет типы данных, которыми являются параметры. Символ s сообщает MySQL, что параметр является строкой.

Аргумент может быть одного из четырех типов:

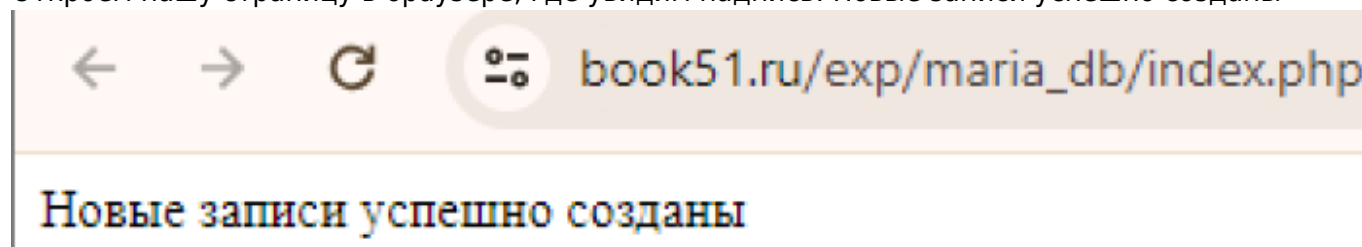
- i - целое число
- d - двойной
- s - строка
- b - БЛОБ

У нас должно быть по одному из них для каждого параметра.

Сообщая mysql, какой тип данных следует ожидать, мы минимизируем риск SQL-инъекций.

Примечание. Если мы хотим вставить какие-либо данные из внешних источников (например, пользовательский ввод), очень важно, чтобы данные были очищены и проверены.

Откроем нашу страницу в браузере, где увидим надпись: Новые записи успешно созданы



Проверим создание новой записи в таблице MyGuests базы данных my_DB в MariaDB с помощью PhpMyAdmin

	id	firstname	lastname	email	reg_date
☐ Изменить Копировать Удалить	1	John	Doe	john@example.com	2024-02-25 17:42:44
☐ Изменить Копировать Удалить	4	Vlad	Polskiy	polskiyvn@gmail.com	2024-02-25 18:04:30
☐ Изменить Копировать Удалить	5	John	Doe	john@example.com	2024-02-25 18:14:00
☐ Изменить Копировать Удалить	6	Mary	Moe	mary@example.com	2024-02-25 18:14:00
☐ Изменить Копировать Удалить	7	Julie	Dooley	julie@example.com	2024-02-25 18:14:00
☐ Изменить Копировать Удалить	8	John	Doe	john@example.com	2024-02-25 18:23:24
☐ Изменить Копировать Удалить	9	Mary	Moe	mary@example.com	2024-02-25 18:23:24
☐ Изменить Копировать Удалить	10	Julie	Dooley	julie@example.com	2024-02-25 18:23:24

From:

<https://wwoss.ru/> - worldwide open-source software

Permanent link:

https://wwoss.ru/doku.php?id=software:development:web:docs:learn:mariadb:%D0%B2atabase_creation&rev=1708874988

Last update: **2024/02/25 18:29**

