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

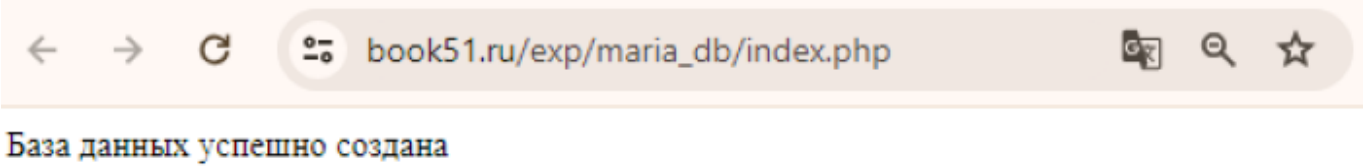
index.php

```
<?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 {
```

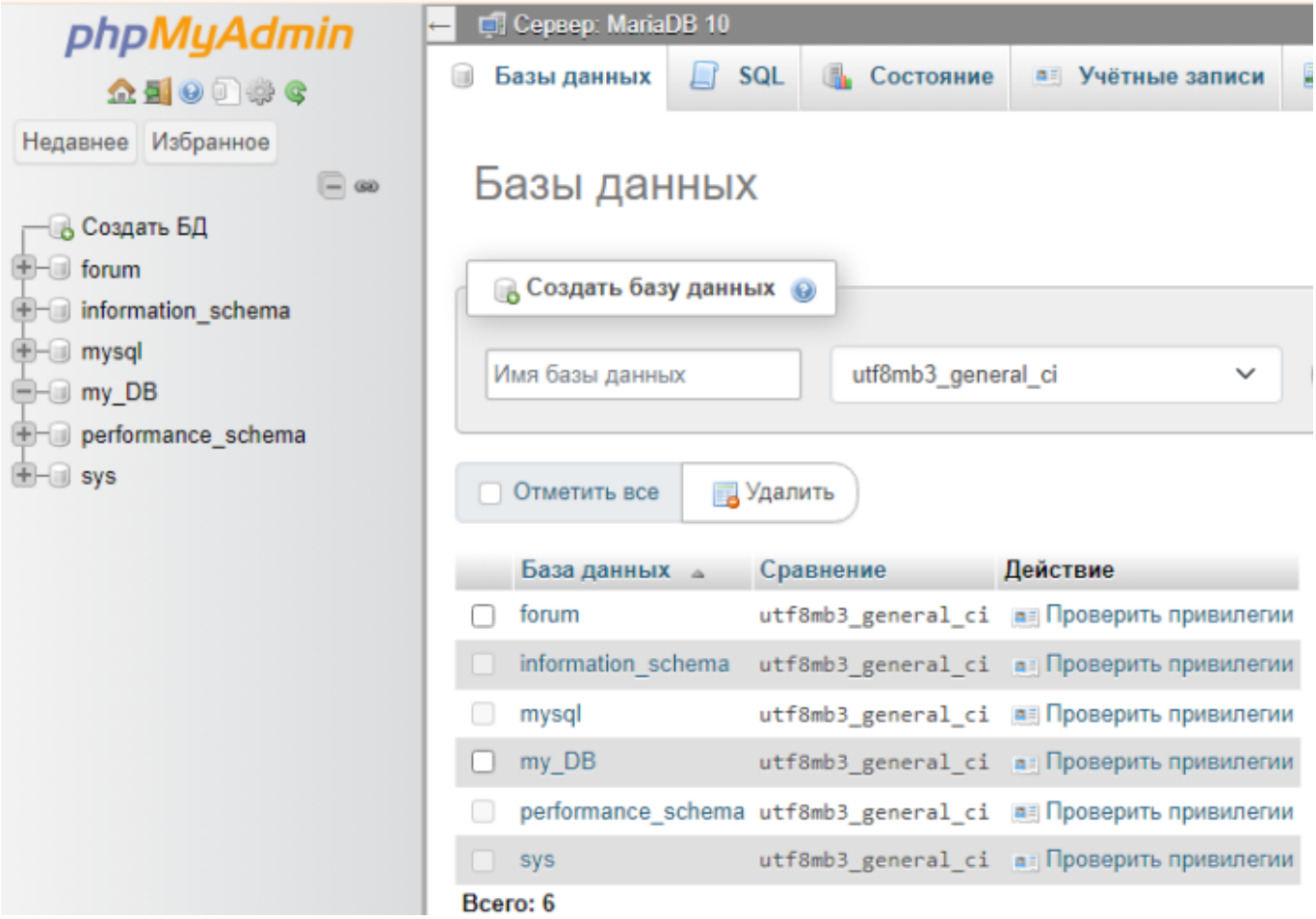
```
  echo "Ошибка создания базы данных: " . $conn->error;
}
// Закроем соединение
$conn->close();
?>
```

- **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.



Проверим создание базы данных my_DB в MariaDB с помощью PhpMyAdmin



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 .

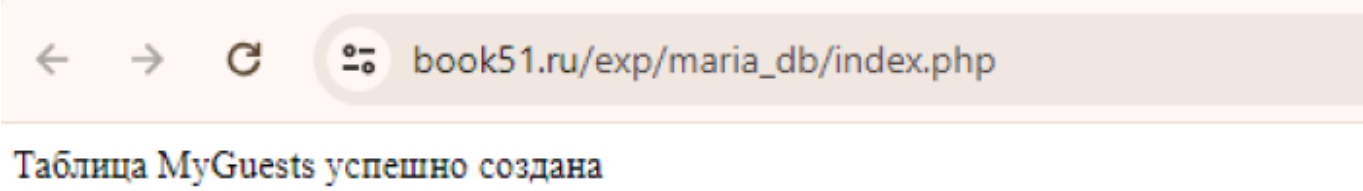
После типа данных вы можете указать другие необязательные атрибуты для каждого столбца:

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

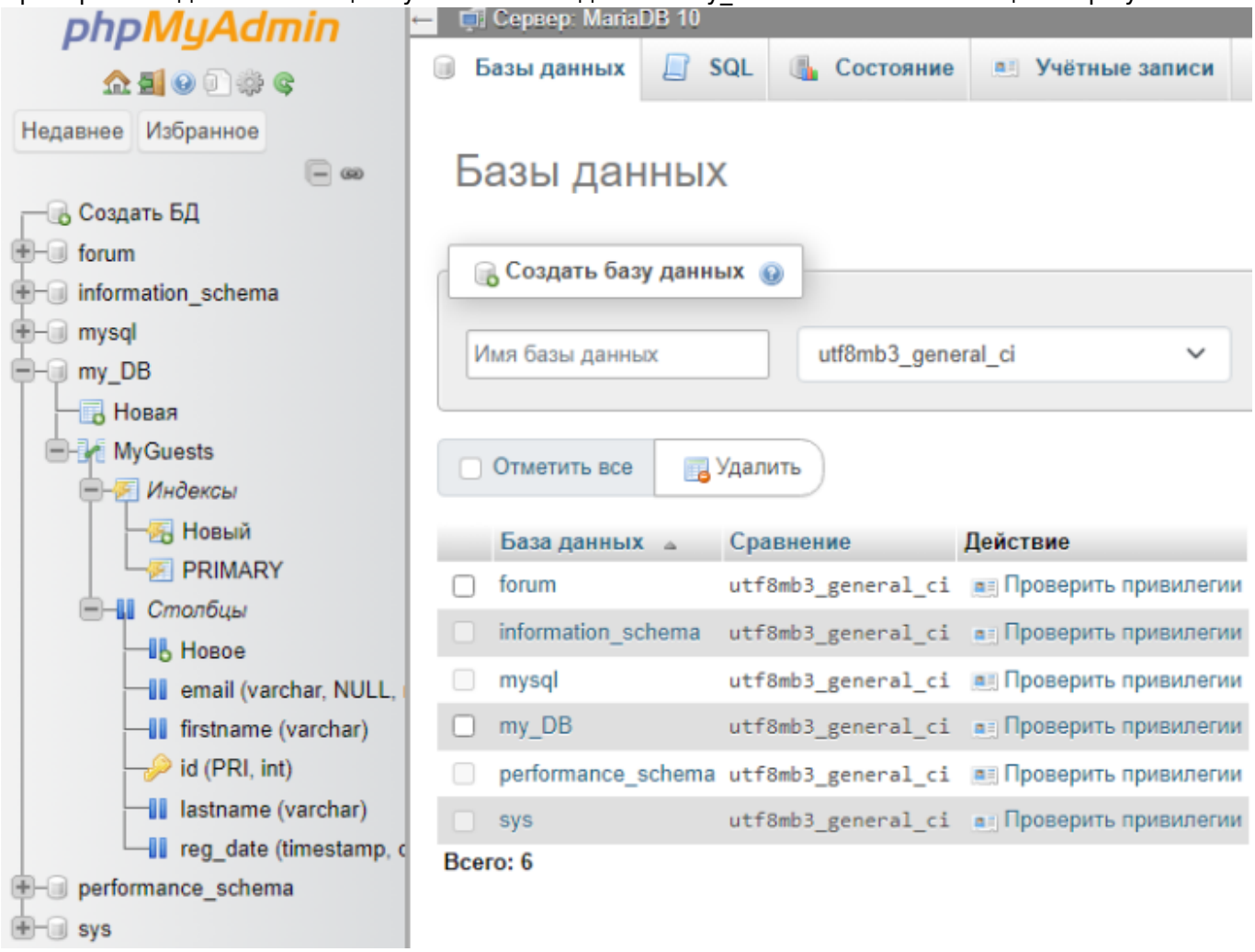
передается другое значение.
- UNSIGNED — используется для числовых типов, ограничивает сохраняемые данные положительными числами и нулем.
- АВТОУвеличение — MySQL автоматически увеличивает значение поля на 1 каждый раз, когда добавляется новая запись.
- ПЕРВИЧНЫЙ КЛЮЧ — используется для уникальной идентификации 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.



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



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_timesamp (например, столбец «reg_date»), его нет необходимости указывать в SQL-запросе; MySQL автоматически добавит это значение.

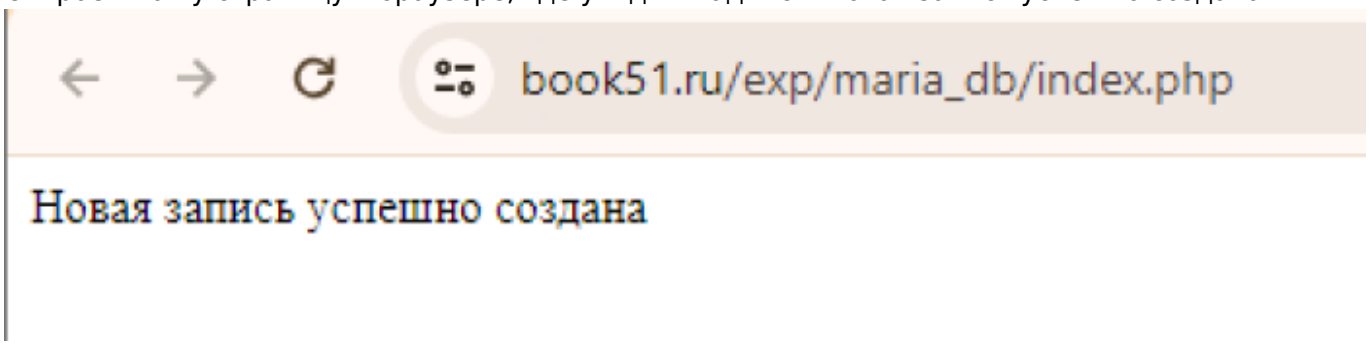
В следующих примерах в таблицу «MyGuests» добавляется новая запись. В тело нашей страницы между тегами <body> и </body> вставим следующий php код

index.php

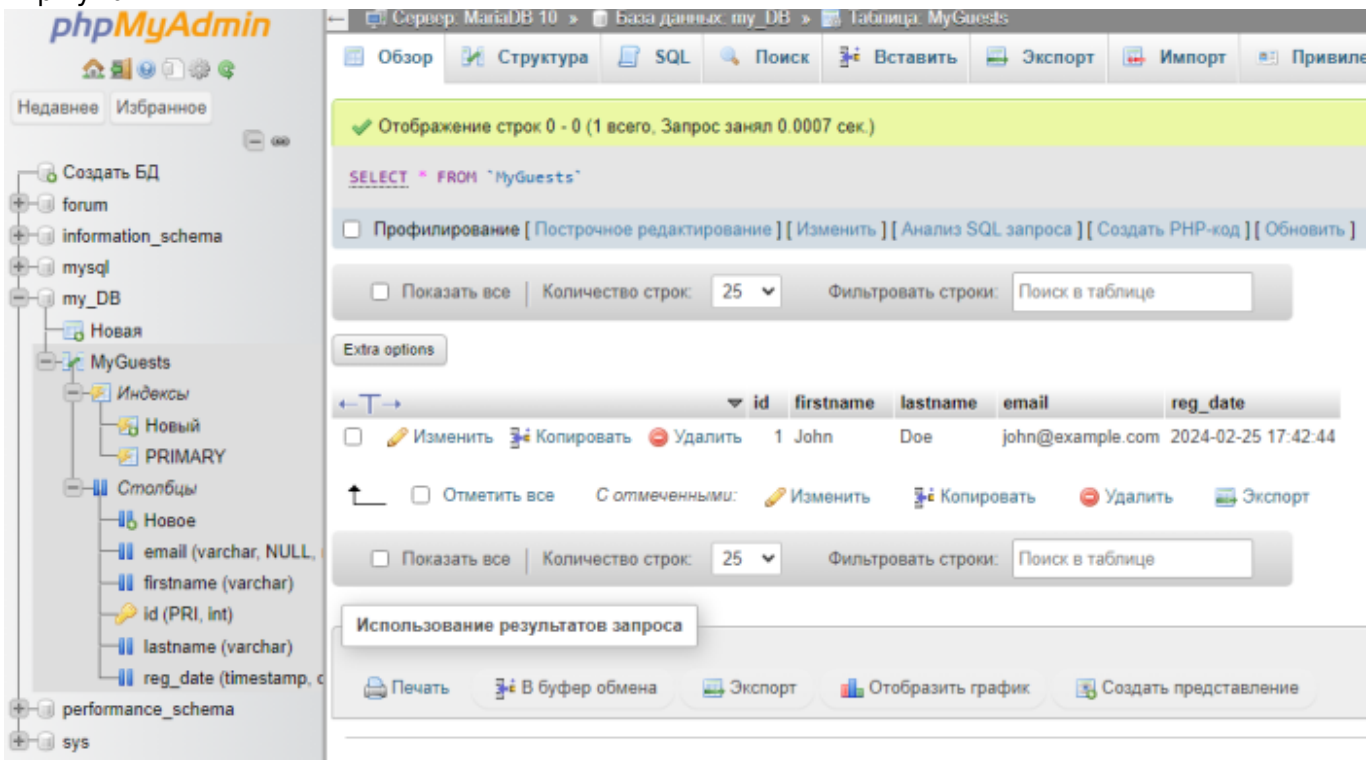
```
    <?php
    $servername = "localhost";
    $username = "root";
    $password = "******";
    $dbname = "my_DB";
    // Создаём соединение
    $conn = new mysqli($servername, $username, $password, $dbname);
    // Проверим подключение
    if ($conn->connect_error) {
      die("Ошибка соединения: " . $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,
```

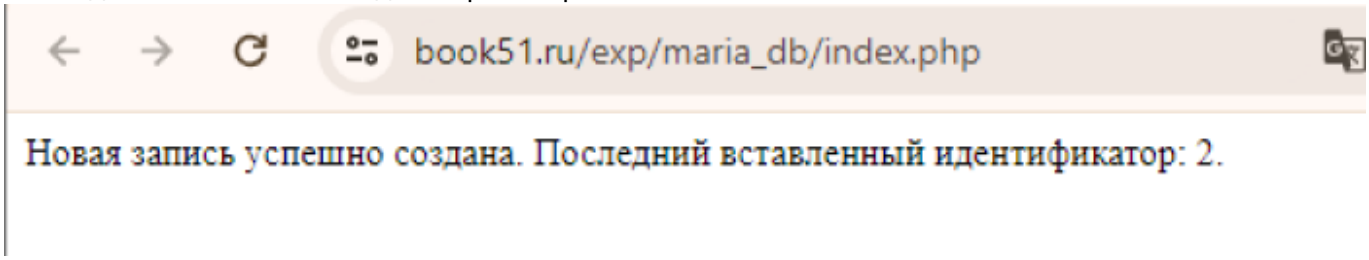
```
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
```

Следующий пример аналогичен придыдущему 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

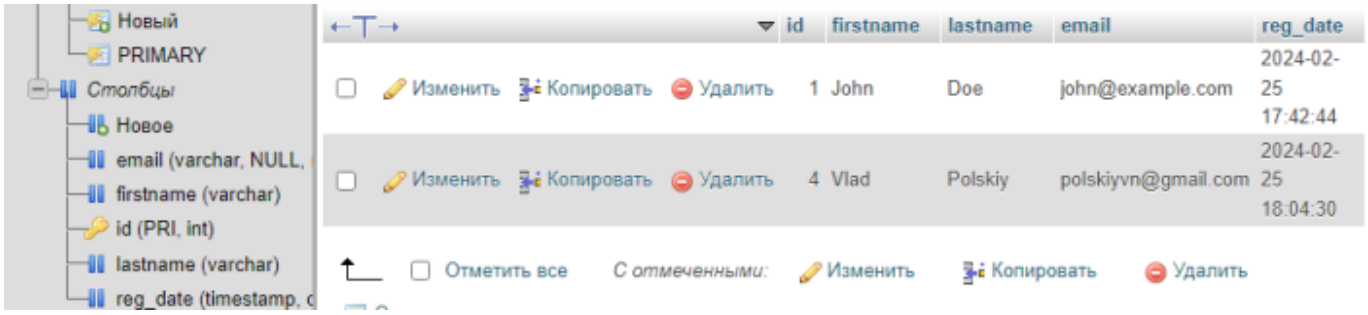
index.php

```
    <?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



Вставка нескольких записей

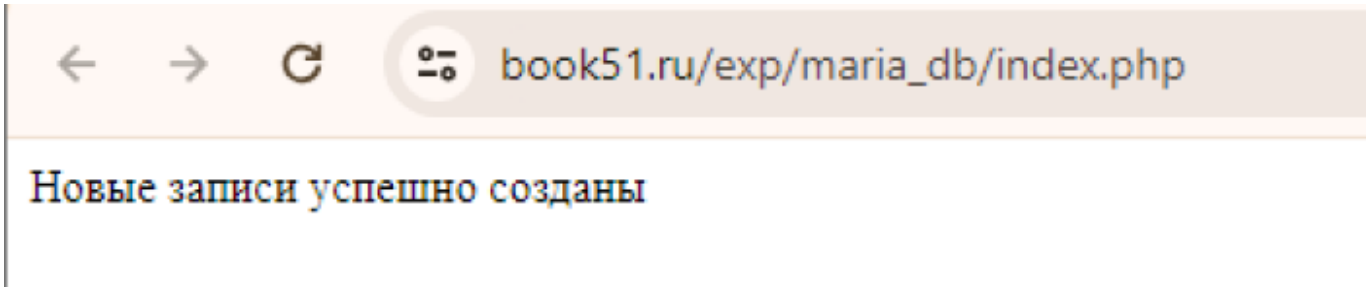
С помощью функции необходимо выполнить несколько операторов SQL mysqli_multi_query().

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

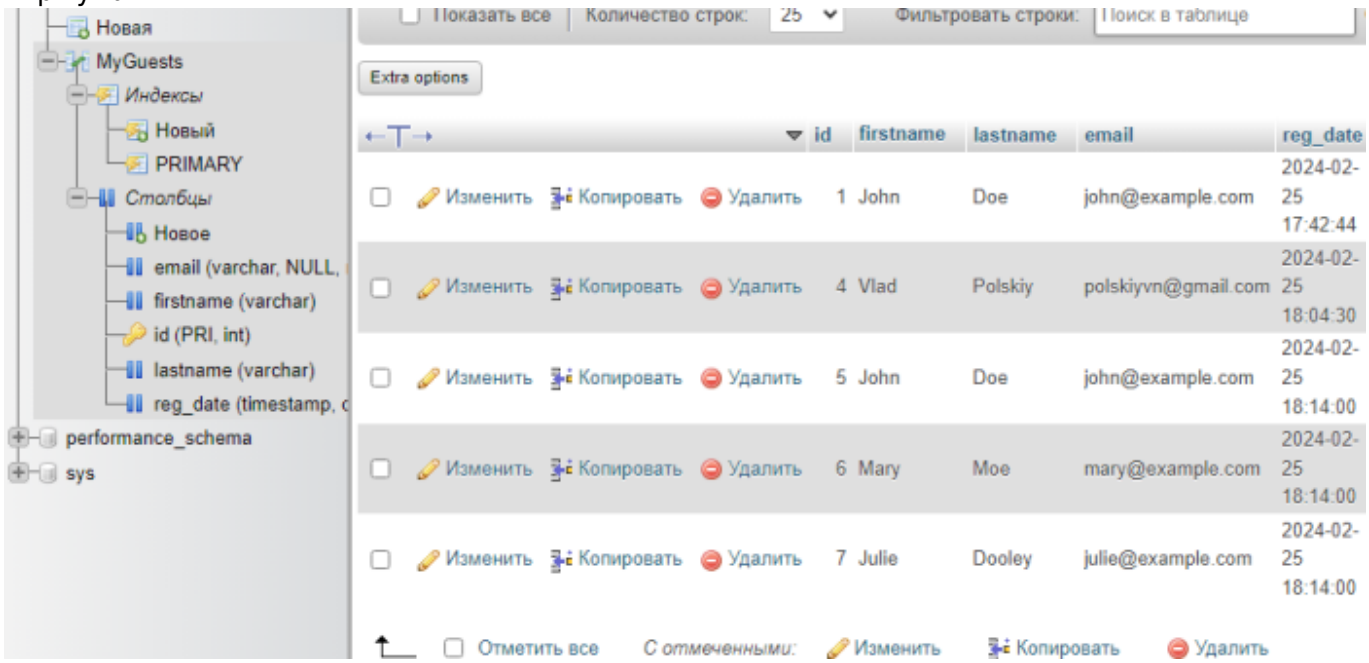
index.php

```
<?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 функциюbind_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.

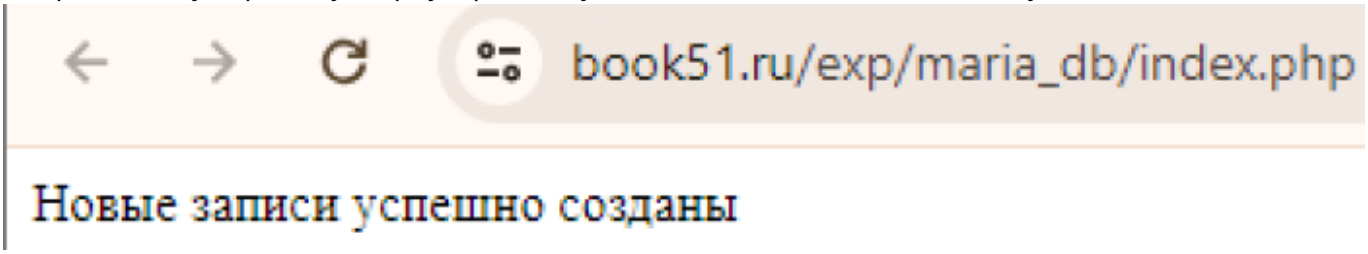
Аргумент может быть одного из четырех типов:

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

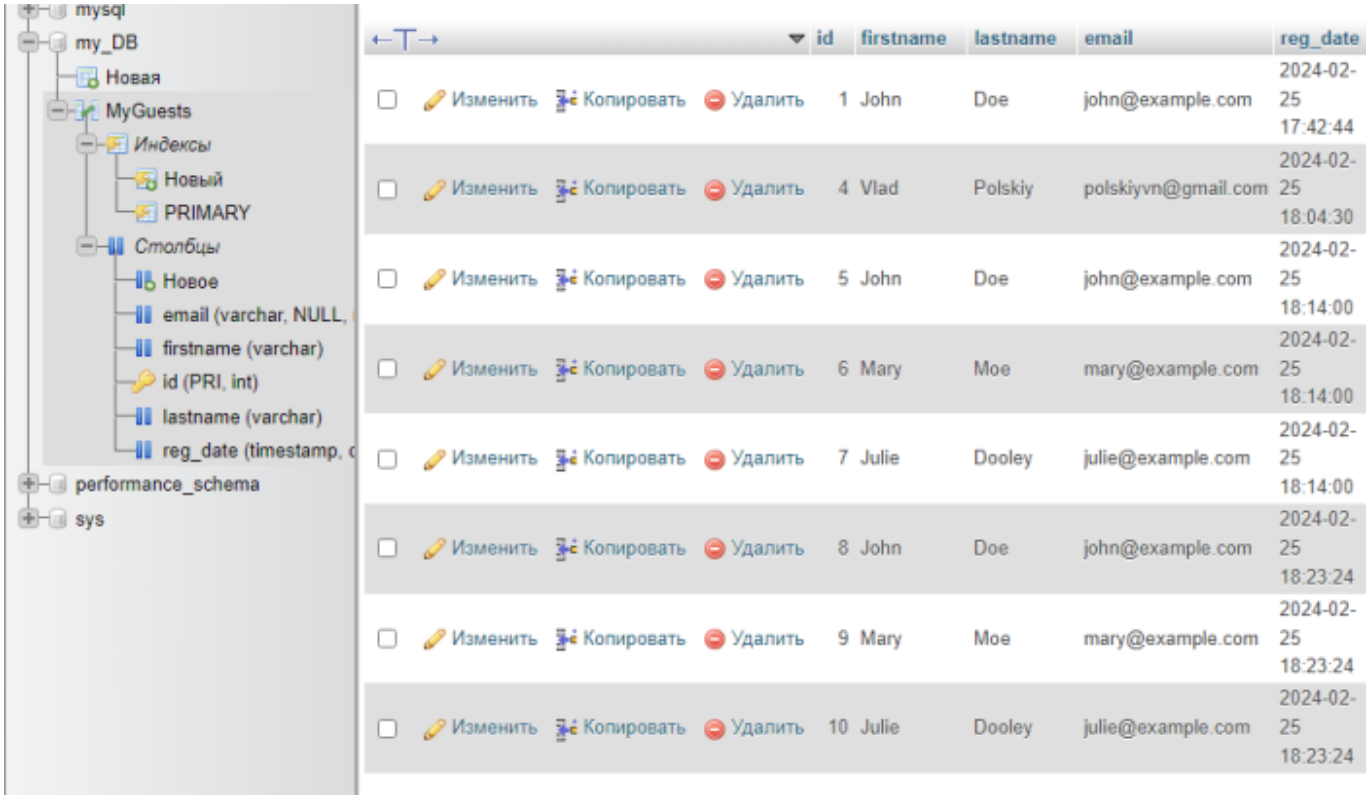
У нас должно быть по одному из них для каждого параметра.

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



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 функциюbind_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.

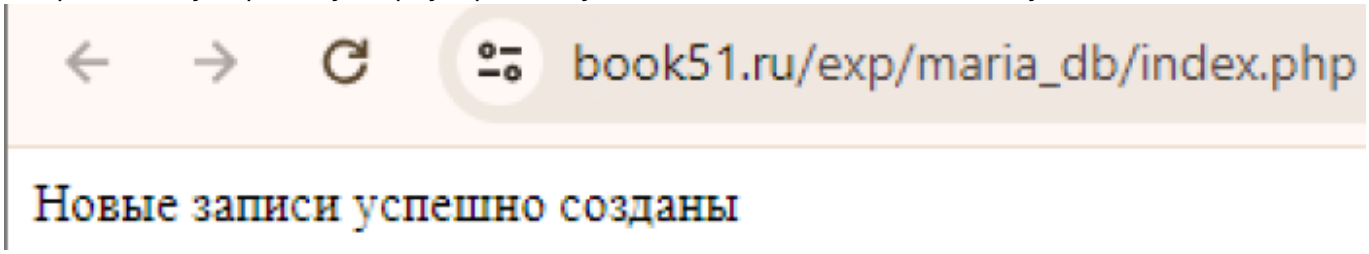
Аргумент может быть одного из четырех типов:

- 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