

3. Возможности MySQL

MySQL — это компактный многопоточковый сервер с реляционной СУБД. MySQL позволяет создавать полноценные клиент-серверные приложения и интегрированные с базами данных веб-сайты. Он характеризуется большой скоростью, устойчивостью и легкостью в использовании.

MySQL был разработан Майклом Видениусом из шведской компании ТсХ для внутренних нужд, которые заключались в быстрой обработке больших баз данных и динамической генерации веб-страниц. Что касается самого названия, то Видениус говорит об этом так: «До конца не ясно, откуда идет название MySQL. В ТсХ базовый каталог, а также значительное число библиотек и утилит в течение десятка лет имели префикс «*my*». Вместе с тем мою дочь тоже зовут Май (My). Поэтому остается тайной, какой из двух источников дал название MySQL».

Особыми целями проектирования MySQL были скорость, надежность и простота использования. Чтобы достичь такой производительности, в ТсХ приняли решение сделать многопоточным внутренний механизм MySQL. Многопоточное приложение одновременно выполняет несколько задач — так, как если бы одновременно выполнялось несколько экземпляров приложения. Каждое входящее соединение обрабатывается отдельным потоком, при этом еще один всегда выполняющийся поток управляет соединениями, поэтому клиентам не приходится ждать завершения выполнения запросов других клиентов. Пока какой-либо поток записывает данные в таблицу, все другие запросы, требующие доступа к этой таблице, просто ждут, пока она не освободится. Клиент может выполнять все допустимые операции, не обращая внимания на другие одновременные соединения. Управляющий поток предотвращает одновременную запись какими-либо двумя потоками в одну и ту же таблицу. В данном случае, выигрыш в скорости благодаря одновременному выполнению нескольких запросов значительно превосходит потери скорости, вызванные увеличением сложности.

Другое преимущество многопоточной обработки присуще всем многопоточным приложениям. Несмотря на то что потоки совместно используют память процесса, они выполняются раздельно. На рисунке 20 показана эта многопоточная природа сервера MySQL.

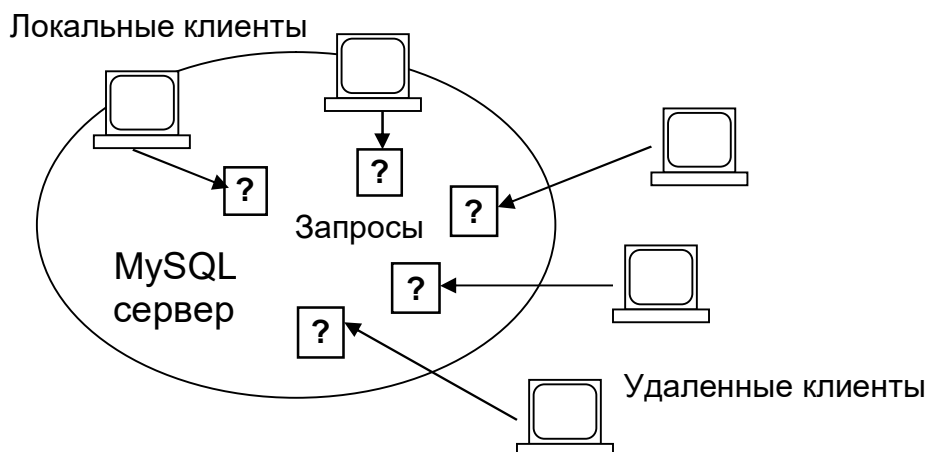


Рисунок 20 — Клиент-серверная архитектура MySQL

Компания утверждает, что использует MySQL с 1996 года в среде с более чем 40 БД, которые содержат 10 000 таблиц, из которых более чем 500 имеют, в свою очередь, более 7 миллионов записей — около 100 Гбайт данных. MySQL-сервер является бесплатным для некоммерческого использования. В ином случае необходимо приобретение лицензии.

MySQL является идеальным решением для малых и средних приложений. Наиболее полно возможности сервера проявляются на Unix-серверах, где есть поддержка многопоточности, что дает значительный прирост производительности.

MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и кроме этого имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД.

В настоящее время большую популярность приобрело сочетание MySQL с языком сценариев PHP, позволяющее обеспечить поддержку доступа к серверам баз данных с веб-страниц.

Связь с базой данных MySQL с использованием языка PHP

Одной из самых приятных особенностей языка PHP является легкость, с которой программист на нем может общаться с базами данных. В этой главе в основном будет говориться о СУБД MySQL, но и с остальными базами данных PHP работает аналогично и с такой же легкостью. Из-за чего уделяется такое внимание именно MySQL? Потому что эта система управления базами данных соответствует духу PHP, она распространяется бесплатно и обладает достаточной мощностью для того, чтобы использовать ее для решения

реальных задач. Кроме того, существуют версии MySQL для самых разных платформ.

Подключение к серверу базы данных

Перед тем как начинать работать со своей базой данных, вам нужно подключиться к серверу. Для этого в языке PHP есть функция `mysql_connect()`. Данная функция имеет три аргумента, в которых указывается имя компьютера, имя пользователя и пароль. Если вы опустите эти необязательные аргументы, то функция будет считать, что требуется компьютер `localhost`, а имя пользователя и его пароль не требуются, т.е. не установлены в таблице `mysqluser`. Конечно, такое подключение не имеет большого смысла — его можно использовать только для проверки сервера, поэтому в дальнейшем в наших примерах мы будем указывать и имя пользователя, и его пароль. Функция `mysql_connect()` возвращает идентификатор подключения, если все прошло успешно. Этот идентификатор можно сохранить в переменной и в дальнейшем пользоваться им для работы с сервером базы данных.

В следующем примере приводится фрагмент, выполняющий подключение к серверу базы данных:

```
$mysql_user = "pGG"; // здесь GG - номер группы
$mysql_password = "pGG";
$conn = mysql_connect("localhost",
$mysql_user, $mysql_password);
if (!$conn) die("Нет соединения с MySQL");
```

Если вы используете PHP в сочетании с сервером Apache, то для подключения к базе данных можете воспользоваться функцией `mysql_pconnect()`. С точки зрения программиста обе эти функции работают одинаково, однако незначительная разница все-таки есть. Если вы воспользуетесь функцией `mysql_pconnect()`, то соединение с сервером не исчезает после завершения работы программы или вызова функции `mysql_close()`.

Выбор базы данных

После того как соединение с сервером MySQL установлено, вам нужно выбрать базу данных, с которой собираетесь работать. Для этого существует функция `mysql_select_db()`. Этой функции нужно передать имя базы данных и второй необязательный аргумент — идентификатор подключения к серверу. Если этот второй аргумент опустить, то по умолчанию будет использован идентификатор последнего полученного подключения. Функция возвращает `true`, если указанная база данных существует и доступ к ней возможен. В следующем примере мы выбираем базу данных с именем `sample`.

```
$database = "sample";
```

```
mysql_select_db($database)
or die ("Нельзя открыть $database");
```

Обработка ошибок

До сих пор мы проверяли значения, возвращаемые функциями MySQL, и вызывали функцию `die()` для прекращения выполнения программы. Однако можно вывести на экран браузера более подробное сообщение об ошибке, которое может пригодиться при отладке программы. При любом неудачном завершении операции MySQL устанавливает номер ошибки и строку с ее описанием. Номер ошибки можно получить используя функцию `mysql_errno()`, а строку с описанием — с помощью функции `mysql_error()`, которая возвращает строку с описанием ошибки, если такая произойдет.

Если программа будет пытаться открыть несуществующую базу данных (`db_2`), то в таком случае функция `die()` выведет примерно такое сообщение:

```
Нельзя открыть db_2: Access denied for user:
'pGG@localhost' to database 'db_2'
```

Добавление данных в таблицу

Получив доступ к базе данных, мы можем добавлять информацию в ее таблицы. Рассмотрим это на примере. Представьте себе, что нам нужно создать узел, на котором пользователи сети могут покупать для себя доменные имена.

Мы создаем в базе данных `sample` таблицу с именем `domains_brNN` (где `NN` — номер бригады, например, 11, 12 и т.д.). Таблица содержит 4 поля: поле первичного ключа с именем `id`, значение которого будет автоматически увеличиваться при добавлении новых записей, поле `domain`, куда записана строка переменной длины типа `VARCHAR`, поле `myname`, содержащее фамилию пользователя, и поле `mail`, содержащее адрес пользователя.

Для создания этой таблицы была использована такая команда SQL:

```
create table domains_brNN
(
    id INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(id),
    domain VARCHAR(20),
    myname VARCHAR(10),
    mail VARCHAR(20)
);
```

Для добавления данных в эту таблицу нам нужно сконструировать и выполнить запрос SQL. Для этого в языке PHP есть функция `mysql_query()`. Этой

функции нужно передать строку с запросом и необязательный идентификатор подключения. Если данный идентификатор опущен, то будет использоваться последнее полученное подключение. Функция возвращает положительное число, в случае успешного выполнения запроса, и false, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса.

```
$query = "create table domains_brNN
        (id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY(id),
        domain VARCHAR(20),
        myname VARCHAR(10),
        mail VARCHAR(20)
        ) ";
// здесь NN - номер бригады
$result = mysql_query($query)
        or die ("<p>Ошибка: ".mysql_error());
```

Имейте в виду, что успешное выполнение запроса не обязательно подразумевает изменение данных в таблице. В листинге 11.1 приведена программа, которая подключается к серверу и выбирает базу данных, а функция `mysql_query()` выполняет оператор `INSERT`, вставляющий данные в таблицу `domains_brNN` базы данных `sample`.

Листинг 11.1. Подключение к базе данных и добавление записи в таблицу

```
<html> <head>
<title> Листинг 11-1. Добавление записи в таблицу
</title> </head> <body>
<?php
$user = "pGG"; // здесь GG - номер группы
$pass = "pGG";
$db = "sample"; $table = "domains_brNN";
$conn = mysql_connect("localhost", $user, $pass);
if (!$conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn )
    or die ("Нельзя открыть $db: ".mysql_error());
// вставить создание таблицы
$query = "INSERT INTO $table (domain, myname, mail)
        VALUES('123abc.com', 'abc', 'abc@mail.ru')";
mysql_query($query, $conn)
    or die ("Нельзя добавить данные в таблицу
            $table: " .mysql_error());
```

```
print "<p>Данные в таблицу $table добавлены";
mysql_close($conn);
?> </body> </html>
```

Обратите внимание на то, что мы не указываем значение поля `id`. Это поле изменяется автоматически.

Естественно, что при каждом выполнении программы из листинга 11.1 в таблицу будет добавляться новая запись, содержащая одни и те же данные. В листинге 11.2 приведена программа, которая добавляет в таблицу данные, введенные пользователем.

Листинг 11.2. Добавление в базу данных информации, введенной пользователем

```
<html> <head>
<title> Листинг 11-2. Добавление в базу данных
        информации, введенной пользователем
</title> </head> <body>
<?php
function Add_to_database($domain, $myname, $mail,
                           &$dberror)
{   #1
    $user = "pGG"; // здесь GG - номер группы
    $pass = "pGG"; $db = "sample";
    $table = "domains_brNN";
    $conn = mysql_connect("localhost", $user, $pass);
    if (!$conn )
        { $dberror = "Нет соединения с MySQL сервером";
          return false; }
    if (! mysql_select_db($db, $conn))
        { $dberror = mysql_error();
          return false;
        }
    $query = "INSERT INTO $table (domain, myname, mail)
              VALUES('$domain', '$name', '$mail')";
    if (! mysql_query($query, $conn))
        { $dberror = mysql_error();
          return false;
        }
    return true;
}   #1
```

```

function Write_form()
{
    #2
    global $PHP_SELF;
    print "<form action='$PHP_SELF'
        method='POST'>\n";

    print "<p>Введите имя домена: \n";
    print "<input type='text' name='domain'> ";
    print "<p>Введите ваш e-mail: \n";
    print "<input type='text' name='mail'> ";
    print "<p>Введите вашу фамилию: \n";
    print "<input type='text' name='myname'> ";
    print "<p><input type='submit' value='Записать! '>\n
        </form>\n";

    }
    #2
    // Ввод данных в таблицу
if (isset($domain) && isset($myname) && isset($mail) )
{
    #3
    // Обязательно проверить, что вводит пользователь!
    $dberror = "";
    $ret = Add_to_database($domain, $myname, $mail,
        $dberror);

    if (!$ret)
        {print "Ошибка: $dberror<br>";}
    else print "Спасибо";
}
#3
else Write_form();
?> </body> </html>

```

В программе из листинга 11.2 мы для простоты и краткости опустили один очень важный момент — нами не проверялись данные, введенные пользователем. А так поступать, вообще-то, не следует. Нельзя доверять пользователю в вопросах достоверности данных. Все данные, введенные им в форму, нужно тщательно проверять.

Нами проверяется существование переменных `$domain`, `$myname` и `$mail`. Если эти переменные определены, то мы можем быть уверены, что форма передана, и вызываем функцию `Add_to_database()`.

Функция `Add_to_database()` имеет 4 аргумента: переменные `$domain`, `$myname`, `$mail`, которые переданы пользователем, и строку `$dberror`. В эту строку будет записано сообщение об ошибке, если такая произойдет.

Поэтому мы передаем эту переменную по ссылке. Все изменения, произошедшие с данной переменной внутри функции, повлияют на саму переменную, а не на ее копию.

Нами создается подключение к серверу MySQL, и если оно не происходит, то прекращаем выполнение программы, вернув значение `false`. Мы выбираем базу данных, содержащую таблицу `domains` и строим SQL-запрос, добавляющий переданные пользователем данные в таблицу. Этот запрос нами передается функции `mysql_query()`. Если хотя бы в одной из функций `mysql_select_db()` или в `mysql_query()` произойдет ошибка, то строка с описанием этой ошибки будет записана в переменную `$dberror` и функция вернет значение `false`. Если все в порядке, функция возвращает `true`.

В самой программе у нас есть возможность проверить значение, возвращенное функцией `Add_to_database()`. Если функция вернула значение `true`, то мы можем быть уверены в том, что данные добавлены. Если нет, то это означает, что произошла ошибка. В таком случае выводим сообщение об ошибке на экран браузера. Мы знаем, что описание этой ошибки содержится в переменной `$dberror`, поэтому включаем ее в текст сообщения.

Если в самом первом операторе `if` мы обнаруживаем, что одна из переменных — `$domain`, `$myname` или `$mail` — не определена, то это означает, что данные не переданы пользователем. В таком случае вызываем функцию `Write_form()`, которая выводит форму на экран пользователя.

Относительно функции `mysql_query()` нужно помнить следующее: ее аргумент, т.е. строка запроса к базе данных, **не должна** содержать символа «;». Фактически это означает, что аргументом может быть только один SQL-запрос.

Следовательно, если требуется последовательно выполнить несколько SQL-запросов, необходимо столько же раз использовать `mysql_query()`, как в следующем примере:

```
$query = "INSERT INTO domains_brNN (domain, myname, mail)
        values('aaa.com', 'aaa', 'aaa@mail.ru')";
$result = mysql_query($query);
$query = "INSERT INTO domains_brNN (domain, myname, mail)
        values('bbb.com', 'bbb', 'bbb@mail.ru')";
$result = mysql_query($query);
$query = "INSERT INTO domains_brNN (domain, myname, mail)
        values('ddd.com', 'ddd', 'ddd@mail.ru')";
$result = mysql_query($query);
```

Получение значения автоматически изменяемого поля

В предыдущих примерах мы добавляли записи в таблицу, не беспокоясь о значении поля `id`, которое автоматически увеличивалось при каждом создании новой записи. Если нам понадобится значение этого поля для некоторой записи, мы всегда можем его получить с помощью SQL-запроса. Однако как поступить, если это значение нам нужно немедленно? В языке PHP есть функция `mysql_insert_id()`, возвращающая значение ключа последней добавленной записи. Этой функции нужно передать идентификатор подключения, а если его опустить, то будет использовано последнее подключение.

Таким образом, если нам необходимо сообщить пользователю номер, присвоенный его заказу, мы можем для этого воспользоваться функцией `mysql_insert_id()` непосредственно после добавления новых данных в таблицу.

```
$query = "INSERT INTO domains_brNN (domain, myname, mail)
        values('$domain', '$myname', '$mail')";
mysql_query($query, $conn);
$id = mysql_insert_id();
print "Спасибо. Ваш номер - $id.
Пожалуйста используйте его в других запросах.";
```

Доступ к информации

Мы уже умеем добавлять данные в базу данных, но необходимо еще и уметь их оттуда читать. Вы, наверное, догадываетесь, что для этого нужно сделать SQL-запрос типа `SELECT`, но как воспользоваться результатами этого запроса, т.е. возвращенными записями? При успешном выполнении запроса функция `mysql_query()` возвращает идентификатор результата запроса, и данный

идентификатор мы можем передавать другим функциям для обработки результата.

Число записей, найденных в запросе

Узнать число записей, возвращенных в результате выполнения запроса `SELECT`, можно с помощью функции `mysql_num_rows()`. Этой функции нужно передать идентификатор запроса, а возвращает она число записей, содержащихся в этом результате.

Результат запроса

После выполнения запроса `SELECT` и получения его идентификатора вы можете в цикле просмотреть все записи, найденные в результате запроса. РНР создает для вас внутренний указатель, в котором записана позиция в наборе записей результата. Этот указатель автоматически перемещается на следующую позицию после обращения к текущей записи.

С помощью функции `mysql_fetch_row()` можно для каждой записи получить массив, состоящий из ее полей. Этой функции нужно передать идентификатор запроса, а вернет она массив. По достижении конца запроса функция `mysql_fetch_row()` вернет значение `false`.

Чтение отдельных полей

Когда вы, выполнив запрос `SELECT`, получили идентификатор результата от функции `mysql_query()`, то можете определить количество возвращенных полей с помощью функции `mysql_num_fields()`. Этой функции нужно передать идентификатор результата, а возвратит она целое число, равное количеству найденных полей.

```
$result = mysql_query("SELECT * from domains_brNN");  
$num_fields = mysql_num_fields($result);
```

Каждое поле в этом наборе имеет свой номер, начиная с нуля, и вы можете, указав идентификатор результата и номер поля, определить все его свойства, включая имя, тип, максимальную длину и флаги.

Для выяснения имени поля передайте идентификатор результата и номер поля функции `mysql_field_name()`.

```
$result = mysql_query("SELECT * from domains_brNN");  
$num_fields = mysql_num_fields($result);  
for ($x=0; $x<$num_fields; $x++)  
    mysql_field_name($result, $x). "<br>\n";
```

Для того чтобы узнать максимальную длину поля, передайте идентификатор результата и номер поля функции `mysql_field_len()`.

```
$result = mysql_query("SELECT * from domains_brNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
    mysql_field_len($result, $x)."<br>\n";
```

Для того чтобы узнать флаги, связанные с этим полем, передайте идентификатор результата и номер поля функции `mysql_field_flags()`.

```
$result = mysql_query("SELECT * from domains_brNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
    mysql_field_flags($result, $x)."<br>\n";
```

Точно так же можно выяснить тип поля.

```
$result = mysql_query("SELECT * from domains_brNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
    mysql_field_type($result, $x)."<br>\n";
```

В листинге 11.3 приведен пример выполнения запроса `SELECT`, который запрашивает все строки таблицы `domains_brNN`, затем определяет размер этой таблицы с помощью функции `mysql_num_rows()` и выводит на экран всю таблицу `domains_brNN`.

Листинг 11.3. Вывод всех записей таблицы

```
<html> <head>
<title> Листинг 11-3. Вывод всех записей таблицы
</title> </head> <body>
<?php
$user = "pGG"; // здесь GG - номер группы
$pass = "pGG"; $db = "sample";
$table = "domains_brNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
    or die ("Нельзя открыть $db: ".mysql_error());
$result = mysql_query("SELECT * FROM $table");
$num_rows = mysql_num_rows($result);
// количество записей в запросе
print "<P>В таблице $table содержится $num_rows строк";
$num_fields = mysql_num_fields($result);
// количество столбцов в запросе
```

```

print "<p><table border=1>\n";
print "<tr>\n";
for ($x=0; $x < $num_fields; $x++)
{
    $name = mysql_field_name($result, $x);
    print "\t<th>$name</th>\n";
    // печатаем имя $x-того столбца
}
print "</tr>\n";
while ($a_row = mysql_fetch_row($result))
{
    // печатаем содержимое столбцов
    print "<tr>\n";
    foreach ($a_row as $field) // $a_row - массив
    print "\t<td>$field</td>\n";
    print "</tr>\n";
}
print "</table>\n";
mysql_close($conn);
?>
</body> </html>

```

Функция `mysql_fetch_row()` возвращает массив.

После подключения к серверу базы данных и выбора базы мы с помощью функции `mysql_query()` посылаем запрос на сервер базы данных. Возвращенный результат запроса сохраняем в переменной `$result`. Потом нами будет использоваться эта переменная для обращения к результату запроса: в переменной `$num_rows` мы сохраняем количество записей в запросе, а в переменной `$num_fields` — количество полей (столбцов) в запросе. Затем, в цикле, в переменную `$name` заносим название `$x`-того поля таблицы и выводим это название в виде ячеек-заголовков `<th>`.

В условном выражении цикла `while` мы присваиваем переменной `$a_row` значение, возвращенное функцией `mysql_fetch_row()`. Результатом операции присваивания является значение ее правого операнда, поэтому данное условное выражение имеет значение `true`, если функция `mysql_fetch_row()` вернет положительное число. Внутри цикла `while` мы просматриваем массив, записанный в переменной `$a_row`, и выводим каждый элемент этого массива на экран, обрамляя его тегами ячейки таблицы.

Кроме того, к полям записи можно обратиться по имени: функция `mysql_fetch_array()` возвращает ассоциативный массив, в котором в

качестве ключа используются имена полей. В следующем примере используется эта функция.

```
print "<table border=1>\n";
while ($a_row = mysql_fetch_array($result))
print "<tr>\n";
print "<td>$a_row[mail]</td>
      <td>$a_row[domain]</td>\n";
print "</tr>\n";
print "</table>\n";
```

Изменение данных

Данные в таблице можно изменять с помощью функции `mysql_query()` в сочетании с оператором `UPDATE`. Как и ранее, успешное выполнение оператора `UPDATE` не обязательно означает, что данные были фактически изменены. Для того чтобы узнать количество измененных данных в таблице, вам придется вызвать функцию `mysql_affected_rows()`. Этой функции нужно передать идентификатор подключения. Как и раньше, если данный идентификатор опустить, то будет использовано последнее подключение. Этой функцией можно пользоваться в сочетании с любым запросом, в результате которого данные могли быть изменены (`UPDATE`, `INSERT`, `REPLACE` или `DELETE`, но не `SELECT`!).

В листинге 11.4 приведен пример программы, позволяющей администратору базы данных изменять любые данные в поле `domain` таблицы `domains_brNN`.

Листинг 11.4. Использование функции `mysql_query()` для изменения данных в таблице

```
<html> <head>
<title> Листинг 11-4. Использование функции mysql_query()
      для изменения данных в таблице
</title> </head> <body>
<?php
$user = "pGG"; // здесь GG - номер группы
$pass = "pGG"; $db = "sample"; $table = "domains_brNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
or die ("Нельзя открыть $db");
```


Полученное значение (в данном примере это должна быть 1) мы выводим на экран браузера.

Нами выводится форма, с помощью которой администратор может вносить изменения. Обратите внимание на то, что мы опять используем функцию `mysql_query()` для того, чтобы получить значение полей `id` и `domain` и использовать их в HTML-тексте для формирования списка. Администратор будет выбирать в этом списке данные, подлежащие изменению. Если администратор уже передал форму и выбранное значение `id` совпадает текущим полем, мы выводим в элемент `OPTION` строку `SELECTED`. Это делается для того, чтобы новое значение, выбранное администратором, было выделено в форме.

Важное замечание. Если аргумент функции `mysql_query()` является переменной, значение которой было передано из другой программы с помощью метода `POST`, и в этой переменной содержится SQL-запрос, в котором есть апострофы, например

```
$query = "SELECT * FROM domains where id='2' ";
```

то эти апострофы будут экранированы обратной косой чертой (`\`). Для того, чтобы можно было использовать значение переменной `$query` в функции `mysql_query()`, необходимо применить функцию `stripslashes()`, которая убирает все управляющие символы (в том числе и обратную косую черту перед апострофами) из указанной строки.

В листинге 11.5 приведен пример программы, использующей функцию `stripslashes()`.

Листинг 11.5. Использование функции `stripslashes()`

```
<html> <head>
<title>Листинг 11-5. Использование функции stripslashes()
</title> </head> <body>
<?php
$user = "pGG"; $pass = "pGG"; $db = "sample";
$table = "domains_brNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
or die ("Нельзя открыть $db");

$query = "SELECT * FROM $table WHERE id='2' ";

settype($query, string);
$plain_query = stripslashes($query);
```

```
$result = mysql_query($plain_query);  
$a_row = mysql_fetch_array($result);  
  
print "<p>Ваше имя: $a_row[myname]";  
  
mysql_close($conn);  
?>  
</body> </html>
```

Перед использованием функции `stripslashes()` желательно явно задать тип «строка» для переменной, которая будет ее аргументом, что делается с помощью функции `settype()`.

Получение информации о базе данных

До сих пор мы рассматривали функции, предназначенные для сохранения и чтения данных. Однако в РНР есть несколько функций для того, чтобы вы могли получить дополнительную информацию о базах данных, доступных в данном подключении.

Список баз данных

Список всех баз данных, доступных для данного подключения, можно получить с помощью функции `mysql_list_dbs()`. Этой функции нужно передать идентификатор подключения, а вернет она другой идентификатор, с помощью которого можно получить список всех доступных баз данных. Для этого вам необходимо вызвать функцию `mysql_tablename()` для каждой найденной базы данных. Эта функция имеет два аргумента: идентификатор результата и индекс базы данных. Функция возвращает имя указанной базы данных. Базы данных индексируются, начиная с нуля. Для того чтобы узнать, сколько баз данных найдено функцией `mysql_list_dbs()`, нужно вызвать функцию `mysql_num_rows()`, передав ей идентификатор результата, возвращенный функцией `mysql_list_dbs()`.

Имейте в виду, что идентификатор результата, возвращенный функцией `mysql_list_dbs()`, можно использовать аналогично тому, как это мы делали с идентификатором результата, возвращенным функцией `mysql_query()`. В частности, его можно передавать функции `mysql_fetch_row()`, которая вернет ассоциативный массив с именами баз данных.

Список таблиц в базе данных

С помощью функции `mysql_list_tables()` есть возможность получить список всех таблиц, входящих в указанную базу данных. Этой функции нужно передать имя базы данных и необязательный аргумент — идентификатор

подключения. Если указанная база данных существует и вы имеет соответствующие права, то функция вернет вам идентификатор результата, который можно обработать функцией `mysql_fetch_row()` или аналогичной ей.

В следующем примере показано, как с помощью функции `mysql_list_tables()` можно получить список всех таблиц базы данных.

```
$result = mysql_list_tables("sample", $conn);  
while ($tab_rows = mysql_fetch_row($result))  
    print "$tab_rows[0]<br>\n";
```

Структура базы данных

В листинге 11.6 приведена программа, в которой с помощью всех рассмотренных ранее средств мы получаем информацию о базе данных, доступной нам через существующее подключение.

Листинг 11.6. Вывод структуры базы данных

```
<html> <head>  
<title> Листинг 11-6. Вывод структуры базы данных  
</title> </head> <body>  
<?php  
$user = "pGG"; $pass = "pGG"; // здесь GG - номер группы  
$db = "study";  
$conn = mysql_connect("localhost", $user, $pass);  
if (! $conn ) die("Нет соединения с MySQL");  
    $tab_res = mysql_list_tables($db, $conn);  
    print "<dl><dd>\n";  
    while ($tab_rows = mysql_fetch_row($tab_res))  
    { #1          созд-е массива имен таблиц  
        print "<b>$tab_rows[0]</b>\n";  
//$tab_rows[0] т.к. работаем только с одной БД  
        $query_res = mysql_query(  
            "SELECT * from $tab_rows[0]");  
        $num_fields = mysql_num_fields($query_res);  
        print "<dl><dd>\n";  
        for ($x=0; $x<$num_fields; $x++)  
        { #2  
            print "<i>";  
            print mysql_field_type($query_res, $x);  
            // тип поля  
            print "</i> <i>";
```

```

        print mysql_field_len($query_res, $x);
        // max-ая длина поля
        print "</i> <b>";
        print mysql_field_name($query_res, $x);
        // имя поля
        print "</b> <i>";
        print mysql_field_flags($query_res, $x);
        // флаги поля (not null и т.п.)
        print "</i><br>\n";
    } #2
    print "</dl>\n";
} #1
print "</dl>\n";
mysql_close($conn);
?>
</body> </html>

```

Мы, как обычно, подключаемся к MySQL-серверу и вызываем функцию `mysql_list_tables()`, в которой указываем имя конкретной базы данных возвращает нам идентификатор результата.

Функция `mysql_list_tables()` возвращает идентификатор результата, который мы передаем функции `mysql_fetch_row()` и начинаем цикл, перебирая имена таблиц. Если бы мы просматривали структуру нескольких баз данных, то массив таблиц `$tab_rows` содержал бы значения, соответствующие всем базам данных. В данном же случае, для одной базы массив `$tab_rows` имеет лишь одно значение: `$tab_rows[0]`.

Нами выводится имя таблицы и составляется SQL-запрос `SELECT`, в котором запрашиваются имена полей таблицы. Запрос мы передаем функции `mysql_query()` и получаем еще один идентификатор результата.

Этот идентификатор мы передаем функции `mysql_num_fields()` и получаем количество полей в найденном наборе.

Нами выполняется цикл `for`, в котором перебираются все поля. В переменной `$x` содержится номер текущего поля. Мы вызываем функции, рассмотренные ранее, которые выводят информацию о каждом поле. Эта информация выводится нами на браузер.

При выполнении программы мы должны получить полный список всех таблиц и характеристики полей, доступных в данном подключении. Ниже показан пример такого вывода.

cust

*int 4 **cnum** not_null primary_key*
*string 10 **cname** not_null*
*string 10 **city** not_null*
*int 6 **rating** not_null*
*int 5 **snum***

ord

*int 4 **onum** not_null primary_key*
*real 7 **amt** not_null*
*date 10 **odate***
*int 4 **cnum***
*int 4 **snum***

sal

*int 4 **snum** not_null primary_key*
*string 10 **sname** not_null*
*string 10 **city** not_null*
*real 7 **comm** not_null*

