Базы данных: основные понятия

**Оглавление документа**

[**1. Ключи**](http://www.webmasterwiki.ru/mysql#h177-2)

[**2. Индексирование**](http://www.webmasterwiki.ru/mysql#h177-3)

[**3. СУБД Mysql**](http://www.webmasterwiki.ru/mysql#h177-4)

[**4. Язык SQL**](http://www.webmasterwiki.ru/mysql#h177-5)

[**5. Основные операторы языка SQL**](http://www.webmasterwiki.ru/mysql#h177-6)

[**6. Описание данных MySQL**](http://www.webmasterwiki.ru/mysql#h177-7)

[**7. Построение интерфейса для добавления информации**](http://www.webmasterwiki.ru/mysql#h177-8)

[**8. Установка соединения**](http://www.webmasterwiki.ru/mysql#h177-9)

[**9. Выбор базы данных**](http://www.webmasterwiki.ru/mysql#h177-10)

[**10. Получение списка полей таблицы**](http://www.webmasterwiki.ru/mysql#h177-11)

[10.1. Синтаксис Mysql\_list\_fields](http://www.webmasterwiki.ru/mysql#h177-12)

[**11. Запись данных в базу данных**](http://www.webmasterwiki.ru/mysql#h177-13)

[11.1. Синтаксис Mysql\_query](http://www.webmasterwiki.ru/mysql#h177-14)

[**12. Отображение данных, хранящихся в Mysql**](http://www.webmasterwiki.ru/mysql#h177-15)

[**13. Доступ к MySQL**](http://www.webmasterwiki.ru/mysql#h177-16)

[**14. Безопасное взаимодействие My SQL и PHP**](http://www.webmasterwiki.ru/mysql#h177-17)

[**15. Литература**](http://www.webmasterwiki.ru/mysql#h177-18)

В жизни мы часто сталкиваемся с необходимостью хранить какую-либо информацию, а потому часто имеем дело и с базами данных.  
  
Например, мы используем записную книжку для хранения номеров телефонов своих друзей и планирования своего времени. Телефонная книга содержит информацию о людях, живущих в одном городе. Все это своего рода базы данных. Ну а раз это базы данных, то посмотрим, как в них хранятся данные. Например, телефонная книга представляет собой таблицу (табл. 10.1).  
  
В этой таблице данные – это собственно номера телефонов, адреса и ФИО., т.е. строки «Иванов Иван Иванович», «32–43–12» и т.п., а названия столбцов этой таблицы, т.е. строки «ФИО», «Номер телефона» и «Адрес» задают смысл этих данных, их семантику.  
  
Таблица 10.1. Пример базы данных: телефонная книга

|  |  |  |
| --- | --- | --- |
| **ФИО** | **Номер телефона** | **Адрес** |
| Иванов Иван Иванович | 32–43–12 | ул. Ленина, 12, 43 |
| Ильин Федор Иванович | 32–32–34 | пр. Маркса, 32, 45 |

Теперь представьте, что записей в этой таблице не две, а две тысячи, вы занимаетесь созданием этого справочника и где-то произошла ошибка (например, опечатка в адресе). Видимо, тяжеловато будет найти и исправить эту ошибку вручную. Нужно воспользоваться какими-то средствами автоматизации. Для управления большим количеством данных программисты (не без помощи математиков) придумали системы управления базами данных (СУБД). По сравнению с текстовыми базами данных электронные СУБД имеют огромное число преимуществ, от возможности быстрого поиска информации, взаимосвязи данных между собой до использования этих данных в различных прикладных программах и одновременного доступа к данным нескольких пользователей.  
  
Для точности дадим определение базы данных, предлагаемое Глоссарий.ру  
  
База данных – это совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. База данных является информационной моделью предметной области. Обращение к базам данных осуществляется с помощью системы управления базами данных (СУБД). СУБД обеспечивает поддержку создания баз данных, централизованного управления и организации доступа к ним различных пользователей.  
  
Итак, мы пришли к выводу, что хранить данные независимо от программ, так, что они связаны между собой и организованы по определенным правилам, целесообразно. Но вопрос, как хранить данные, по каким правилам они должны быть организованы, остался открытым. Способов существует множество (кстати, называются они моделями представления или хранения данных). Наиболее популярные – объектная и реляционная модели данных.  
  
Автором реляционной модели считается Э. Кодд, который первым предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение) и показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение.  
  
Таким образом, реляционная база данных представляет собой набор таблиц (точно таких же, как приведенная выше), связанных между собой. Строка в таблице соответствует сущности реального мира (в приведенном выше примере это информация о человеке).  
  
Примеры реляционных СУБД: Mysql, PostgreSql. В основу объектной модели положена концепция объектно-ориентированного программирования, в которой данные представляются в виде набора объектов и классов, связанных между собой родственными отношениями, а работа с объектами осуществляется с помощью скрытых (инкапсулированных) в них  
методов.  
  
Примеры объектных СУБД: Cache, GemStone (от Servio Corporation), ONTOS (ONTOS).  
  
В последнее время производители СУБД стремятся соединить два этих подхода и проповедуют объектно-реляционную модель представления данных. Примеры таких СУБД – IBM DB2 for Common Servers, Oracle8.  
  
Поскольку мы собираемся работать с Mysql, то будем обсуждать аспекты работы только с реляционными базами данных. Нам осталось рассмотреть еще два важных понятия из этой области: ключи и индексирование, после чего мы сможем приступить к изучению языка запросов SQL.

1. Ключи

Для начала давайте подумаем над таким вопросом: какую информацию нужно дать о человеке, чтобы собеседник точно сказал, что это именно тот человек, сомнений быть не может, второго такого нет? Сообщить фамилию, очевидно, недостаточно, поскольку существуют однофамильцы. Если собеседник человек, то мы можем приблизительно объяснить, о ком речь, например вспомнить поступок, который совершил тот человек, или еще как-то. Компьютер же такого объяснения не поймет, ему нужны четкие правила, как определить, о ком идет речь. В системах управления базами данных для решения такой задачи ввели понятие первичного ключа.  
  
Первичный ключ (primary key, PK) – минимальный набор полей, уникально идентифицирующий запись в таблице. Значит, первичный ключ – это в первую очередь набор полей таблицы, во-вторых, каждый набор значений этих полей должен определять единственную запись (строку) в таблице и, в-третьих, этот набор полей должен быть минимальным из всех обладающих таким же свойством.  
  
Поскольку первичный ключ определяет только одну уникальную запись, то никакие две записи таблицы не могут иметь одинаковых значений первичного ключа.  
  
Например, в нашей таблице (см. выше) ФИО и адрес позволяют однозначно выделить запись о человеке. Если же говорить в общем, без связи с решаемой задачей, то такие знания не позволяют точно указать на единственного человека, поскольку существуют однофамильцы, живущие в разных городах по одному адресу. Все дело в границах, которые мы сами себе задаем. Если считаем, что знания ФИО, телефона и адреса без указания города для наших целей достаточно, то все замечательно, тогда поля ФИО и адрес могут образовывать первичный ключ. В любом случае проблема создания первичного ключа ложится на плечи того, кто проектирует базу данных (разрабатывает структуру хранения данных). Решением этой проблемы может стать либо выделение характеристик, которые естественным образом определяют запись в таблице (задание так называемого логического, или естественного, PK), либо создание дополнительного поля, предназначенного именно для однозначной идентификации записей в таблице (задание так называемого суррогатного, или искусственного, PK).  
  
Примером логического первичного ключа является номер паспорта в базе данных о паспортных данных жителей или ФИО и адрес в телефонной книге (таблица выше). Для задания суррогатного первичного ключа в нашу таблицу можно добавить поле id (идентификатор), значением которого будет целое число, уникальное для каждой строки таблицы. Использование таких суррогатных ключей имеет смысл, если естественный первичный ключ представляет собой большой набор полей или его выделение нетривиально.  
  
Кроме однозначной идентификации записи, первичные ключи используются для организации связей с другими таблицами.  
  
Например, у нас есть три таблицы: содержащая информацию об исторических личностях (Persons), содержащая информацию об их изобретениях (Artifacts) и содержащая изображения как личностей, так и артефактов (Images) (рис 10.1).  
  
Первичным ключом во всех этих таблицах является поле id (идентификатор). В таблице Artifacts есть поле author, в котором записан идентификатор, присвоенный автору изобретения в таблице Persons. Каждое значение этого поля является внешним ключом для первичного ключа таблицы Persons. Кроме того, в таблицах Persons и Artifacts есть поле photo, которое ссылается на изображение в таблице Images. Эти поля также являются внешними ключами для первичного ключа таблицы Images и устанавливают однозначную логическую связь Persons-Images и Artifacts-Images. То есть если значение внешнего ключа photo в таблице личности равно 10, то это значит, что фотография этой личности имеет id=10 в таблице изображений. Таким образом, внешние ключи используются для организации связей между таблицами базы данных (родительскими и дочерними) и для поддержания ограничений ссылочной целостности данных.

2. Индексирование

Одна из основных задач, возникающих при работе с базами данных, – это задача поиска. При этом, поскольку информации в базе данных, как правило, содержится много, перед программистами встает задача не просто поиска, а эффективного поиска, т.е. поиска за сравнительно небольшое время и с достаточной точностью.   
  
Для этого (для оптимизации производительности запросов) производят индексирование некоторых полей таблицы. Использовать индексы полезно для быстрого поиска строк с указанным значением одного столбца. Без индекса чтение таблицы осуществляется по всей таблице, начиная с первой записи, пока не будут найдены соответствующие строки. Чем больше таблица, тем больше накладные расходы. Если же таблица содержит индекс по рассматриваемым столбцам, то база данных может быстро определить позицию для поиска в середине файла данных без просмотра всех данных. Это происходит потому, что база данных помещает проиндексированные поля поближе в памяти, так, чтобы можно было побыстрее найти их значения. Для таблицы, содержащей 1000 строк, это будет как минимум в 100 раз быстрее по сравнению с последовательным перебором всех записей. Однако в случае, когда необходим доступ почти ко всем 1000 строкам, быстрее будет последовательное чтение, так как при этом не требуется операций поиска по диску. Так что иногда индексы бывают только помехой. Например, если копируется большой объем данных в таблицу, то лучше не иметь никаких индексов. Однако в некоторых случаях требуется задействовать сразу несколько индексов (например, для обработки запросов к часто используемым таблицам).  
  
Если говорить о Mysql, то там существует три вида индексов: PRIMARY, UNIQUE, и INDEX, а слово ключ (KEY) используется как синоним слова индекс (INDEX). Все индексы хранятся в памяти в виде B-деревьев.  
  
PRIMARY – уникальный индекс (ключ) с ограничением, что все индексированные им поля не могут иметь пустого значения (т.е. они NOT NULL). Таблица может иметь только один первичный индекс, но он может состоять из нескольких полей.  
  
UNIQUE – ключ (индекс), задающий поля, которые могут иметь только уникальные значения.  
  
INDEX – обычный индекс (как мы описали выше). В Mysql, кроме того, можно индексировать строковые поля по заданному числу символов от начала строки.

3. СУБД Mysql

Продолжим разговор о СУБД Mysql. Mysql – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. Mysql – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов.  
  
Работать с Mysql можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс (кстати, написанный на PHP) для работы с этой СУБД. Называется он PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в Mysql.  
  
PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQLфункций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet. Но, тем не менее, стоит все же поработать и в текстовом режиме.  
  
Перед тем как переходить к детальному изучению языка SQL, несколько слов об установке Mysql и подготовке к работе. Если вы не собираетесь заниматься администрированием сервера, то информация, приведенная ниже, пригодится вам только для общего развития. Итак, устанавливается Mysql очень просто – автоматически, пару раз нажмите OK, и все. После этого вы можете зайти в директорию, где лежат файлы типа Mysql.exe, Mysqld.exe и т.п. (у нас под Windows XP это c:\Mysql\bin) Последний файл запускает Mysql-сервер. В некоторых системах сервер запускается в виде сервиса. После запуска сервера следует запустить Mysql-клиент, запустив программу Mysql.exe. Здесь даже пароля не спросят. Более того, если вы наберете shell> Mysql.exe -u root или shell>Mysql -u root Mysql то получите все права администратора Mysql сервера. Кстати, выполнять эти команды надо, находясь в той директории, где лежат файлы Mysql.exe.  
Для начала, не вдаваясь в подробности команд, исправим эти два недочета (отсутствие пароля у администратора и возможность входа анонимным пользователям):

shell> Mysql -u root Mysql  
Mysql> UPDATE user SET Password=PASSWORD('new\_password')  
WHERE user='root';  
Mysql> DELETE FROM user WHERE user='';  
Mysql> FLUSH PRIVILEGES;

Все данные о пользователях Mysql хранит в таблице user в специальной базе данных Mysql, доступ к которой имеет только администратор сервера. Поэтому, чтобы изменить какой-либо пароль, нужно изменить эту таблицу. Пароль задается с помощью функции PASSWORD, которая кодирует введенные данные. Кроме изменения пароля администратора, нужно еще удалить всех пользователей, не имеющих логина (команда DELETE). Команда Flush Privileges заставляет вступить в действие изменения, произошедшие в системной базе данных (Mysql).  
  
Теперь создадим базу данных, с которой будем работать (мы все еще работаем как администратор сервера):  
Mysql>create database book;  
  
Как можно заметить, все команды в Mysql заканчиваются точкой с запятой. Если вы забыли поставить этот знак, то выдается приглашение его поставить до тех пор, пока это не будет сделано:

Mysql> show tables  
->  
->

Теперь последнее действие – создадим простого пользователя, предоставим ему доступ к созданной базе данных, и начнем работать.

Mysql> GRANT ALL PRIVILEGES ON book.\* TO nina@localhost  
IDENTIFIED BY '123';

Команда GRANT наделяет пользователя nina, зашедшего на сервер с этой же машины (c localhost) и идентифицируемого паролем «123», определенными правами (в данном случае всеми) на все таблицы базы данных book. Теперь мы можем выйти и зайти как пользователь nina с соответствующим паролем:

shell>Mysql -u nina -p  
Enter password: \*\*\*  
Welcome to the Mysql monitor!...  
Mysql>

Если вы собираетесь пользоваться базой данных на чужом сервере, то его администратор проделает все описанные выше действия за вас, т.е. все настроит и создаст пользователя и базу данных. В следующей главе описаны команды языка SQL, которые пригодятся для работы с данными, хранящимися в СУБД Mysql.

4. [Язык SQL](http://www.webmasterwiki.ru/MySQL/JazykSQL" \o "My SQL / Язык SQL)

5. [Основные операторы языка SQL](http://www.webmasterwiki.ru/MySQL/Operatory" \o "My SQL / Операторы)

6. [Описание данных MySQL](http://www.webmasterwiki.ru/MySQL/OpisanieDannyx" \o "My SQL / Описание Данных)

7. Построение интерфейса для добавления информации

Итак, у нас есть какая-то таблица в базе данных. Чтобы построить интерфейс для добавления информации в эту таблицу, нужно ее структуру (т.е. набор ее полей) отобразить в html-форму.  
  
Разобьем эту задачу на следующие подзадачи:

* установка соединения с БД;
* выбор рабочей БД;
* получение списка полей таблицы;
* отображение полей в html-форму.

После этого данные, введенные в форму, нужно записать в базу данных.  
  
Рассмотрим все эти задачи по порядку.

8. Установка соединения

Итак, первое, что нужно сделать, – это установить соединение с базой данных.  
  
Воспользуемся функцией Mysql\_connect.  
  
Синтаксис Mysql\_connect

ресурс Mysql\_connect ( [строка server  
[, строка username [, строка password  
[, логическое new\_link  
[, целое client\_flags]]]]])

Данная функция устанавливает соединение с сервером Mysql и возвращает указатель на это соединение или FALSE в случае неудачи.   
  
Для отсутствующих параметров устанавливаются следующие значения по умолчанию:  
server = 'localhost:3306'  
username = имя пользователя владельца процесса сервера  
password = пустой пароль  
  
Если функция вызывается дважды с одними и теми же параметрами, то новое соединение не устанавливается, а возвращается ссылка на старое соединение.  
  
Чтобы этого избежать, используют параметр new\_link, который заставляет в любом случае открыть еще одно соединение.  
  
Параметр client\_flags – это комбинация следующих констант:

MYSQL\_CLIENT\_COMPRESS (использовать протокол сжатия),  
MYSQL\_CLIENT\_IGNORE\_SPACE (позволяет вставлять пробелы после имен функций), MYSQL\_CLIENT\_INTERACTIVE (ждать interactive\_timeout секунд - вместо wait\_timeout - до закрытия соединения).

Параметр new\_link появился в PHP 4.2.0, а параметр client\_flags – в PHP 4.3.0.  
  
Соединение с сервером закрывается при завершении исполнения скрипта, если оно до этого не было закрыто с помощью функции Mysql\_close().  
  
Итак, устанавливаем соединение с базой данных на локальном сервере для пользователя nina с паролем “123”:

<?  
$conn = Mysql\_connect(  
"localhost", "nina","123")  
or die("Невозможно установить  
соединение: ". Mysql\_error());  
echo "Соединение установлено";  
Mysql\_close($conn);  
?>

Действие Mysql\_connect равносильно команде

shell>Mysql -u nina -p123

9. Выбор базы данных

После установки соединения нужно выбрать базу данных, с которой будем работать. Наши данные хранятся в базе данных book. В Mysql выбор базы данных осуществляется с помощью команды use:

Mysql>use book;

В PHP для этого существует функция Mysql\_select\_db.  
  
Синтаксис Mysql\_select\_db:  
  
логическое

Mysql\_select\_db (строка database\_name [, ресурс link\_identifier])

Эта функция возвращает TRUE в случае успешного выбора базы данных и FALSE

* в противном случае.

Сделаем базу данных book рабочей:

<?  
$conn = Mysql\_connect(  
"localhost","nina","123")  
or die("Невозможно установить  
соединение: ". Mysql\_error());  
echo "Соединение установлено";  
Mysql\_select\_db("book");  
?>

10. Получение списка полей таблицы

Теперь можно заняться собственно решением задачи. Как получить список полей таблицы? Очень просто. В PHP и на этот случай есть своя команда – Mysql\_list\_fields.

10.1. Синтаксис Mysql\_list\_fields

ресурс Mysql\_list\_fields (  
строка database\_name,  
строка table\_name  
[, ресурс link\_identifier])

Эта функция возвращает список полей в таблице table\_name в базе данных database\_name. Получается, что выбирать базу данных нам было необязательно, но это пригодится позже. Как можно заметить, результат работы этой функции – переменная типа ресурс. То есть это не совсем то, что мы хотели получить. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги.  
  
Функция Mysql\_field\_name возвращает имя поля, полученного в результате выполнения запроса. Функция Mysql\_field\_len возвращает длину поля. Функция Mysql\_field\_type возвращает тип поля, а функция Mysql\_field\_flags возвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, blob и т.д. Флаги могут быть not\_null, primary\_key, unique\_key, blob,  
auto\_increment и т.д.  
  
Синтаксис у всех этих команд одинаков:

строка Mysql\_field\_name (ресурс result, целое field\_offset)  
строка Mysql\_field\_type ( ресурс result, целое field\_offset)  
строка Mysql\_field\_flags ( ресурс result, целое field\_offset)  
строка Mysql\_field\_len ( ресурс result, целое field\_offset)

Здесь result – это идентификатор результата запроса (например, запроса, отправленного функциями Mysql\_list\_fields или Mysql\_query (о ней будет рассказано позднее)), а field\_offset – порядковый номер поля в результате.  
  
Вообще говоря, то, что возвращают функции типа Mysql\_list\_fields или Mysql\_query, представляет собой таблицу, а точнее, указатель на нее. Чтобы получить из этой таблицы конкретные значения, нужно задействовать специальные функции, которые построчно читают эту таблицу. К таким функциям и относятся Mysql\_field\_name и т.п. Чтобы перебрать все строки в таблице результата выполнения запроса, нужно знать число строк в этой таблице. Команда Mysql\_num\_rows(ресурс result) возвращает число строк во множестве результатов  
result.  
  
А теперь попробуем получить список полей таблицы Artifacts (коллекция экспонатов).

<?  
$conn = Mysql\_connect(  
"localhost","nina","123")  
or die("Невозможно установить  
соединение: ". Mysql\_error());  
echo "Соединение установлено";  
Mysql\_select\_db("book");  
$list\_f = Mysql\_list\_fields (  
"book","Artifacts",$conn);  
$n = Mysql\_num\_fields($list\_f);  
for($i=0;$i<$n; $i++){  
$type = Mysql\_field\_type($list\_f, $i);  
$name\_f = Mysql\_field\_name($list\_f,$i);  
$len = Mysql\_field\_len($list\_f, $i);  
$flags\_str = Mysql\_field\_flags (  
$list\_f, $i);  
echo "<br>Имя поля: ". $name\_f;  
echo "<br>Тип поля: ". $type;  
echo "<br>Длина поля: ". $len;  
echo "<br>Строка флагов поля: ".  
$flags\_str . "<hr>";  
}  
?>

В результате должно получиться примерно вот что (если в таблице всего два поля, конечно):  
  
Имя поля: id  
Тип поля: int  
Длина поля: 11  
Строка флагов поля:  
not\_null primary\_key auto\_increment  
Имя поля: title  
Тип поля: string  
Длина поля: 255  
Строка флагов поля:  
Отображение списка полей в html-форму  
  
Теперь немножко подкорректируем предыдущий пример. Будем не просто выводить информацию о поле, а отображать его в подходящий элемент html- формы. Так, элементы типа BLOB переведем в textarea (заметим, что поле description, которое мы создавали с типом TEXT, отображается как имеющее тип BLOB), числа и строки отобразим в текстовые строки ввода <input type=text>, а элемент, имеющий метку автоинкремента, вообще не будем отображать, поскольку его значение устанавливается автоматически.  
  
Все это решается довольно просто, за исключением выделения из списка флагов флага auto\_increment. Для этого нужно воспользоваться функцией explode.  
  
Синтаксис explode:

массив explode( строка separator, строка string [, int limit])

Эта функция разбивает строку string на части с помощью разделителя separator и возвращает массив полученных строк.  
  
В нашем случае в качестве разделителя нужно взять пробел “ ", а в качестве исходной строки для разбиения – строку флагов поля.  
  
Итак, создадим форму для ввода данных в таблицу Artifacts:

<?  
$conn=Mysql\_connect("localhost","nina","123");  
// устанавливаем соединение  
$database = "book";  
$table\_name = "Artifacts";  
Mysql\_select\_db($database); // выбираем базу данных для  
// работы  
$list\_f = Mysql\_list\_fields($database,$table\_name);  
// получаем список полей в базе  
$n = Mysql\_num\_fields($list\_f); // число строк в результате  
// предыдущего запроса (т.е. сколько всего  
// полей в таблице Artifacts)  
echo "<form method=post action=insert.php>";  
// создаем форму для ввода данных  
echo "&nbsp;<TABLE BORDER=0 CELLSPACING=0 width=50% ><tr>  
<TD BGCOLOR='#005533' align=center><font color='#FFFFFF'>  
<b> Add new row in  
$table\_name</b></font></td></tr><tr><td></td></tr></TABLE>";  
echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";  
// для каждого поля получаем его имя, тип, длину и флаги  
for($i=0;$i<$n; $i++){  
$type = Mysql\_field\_type($list\_f, $i);  
$name\_f = Mysql\_field\_name ($list\_f,$i);  
$len = Mysql\_field\_len($list\_f, $i);  
$flags\_str = Mysql\_field\_flags ($list\_f, $i);  
// из строки флагов делаем массив,  
// где каждый элемент массива - флаг поля  
$flags = explode(" ", $flags\_str);  
foreach ($flags as $f){  
if ($f == 'auto\_increment') $key = $name\_f;  
// запоминаем имя автоинкремента  
}  
/\* для каждого поля, не являющегося автоинкрементом, в  
зависимости от его типа выводим подходящий элемент формы \*/  
if ($key <> $name\_f){  
echo "<tr><td align=right bgcolor='#C2E3B6'><font size=2>  
<b>&nbsp;". $name\_f ."</b></font></td>";  
switch ($type){  
case "string":  
$w = $len/5;  
echo "<td><input type=text name=\"$name\_f\"  
size = $w ></td>";  
break;  
case "int":  
$w = $len/4;  
echo "<td><input type=text name=\"$name\_f\"  
size = $w ></td>";  
break;  
case "blob":  
echo "<td><textarea rows=6 cols=60  
name=\"$name\_f\"></textarea></td>";  
break;  
}  
}  
echo "</tr>";  
}  
echo "</table>";  
echo "<input type=submit name='add' value='Add'>";  
echo "</form>";  
?>

**Листинг 11.0.1. Форма для ввода данных в таблицу Artifacts**

11. Запись данных в базу данных

Итак, форма создана. Теперь нужно сделать самое главное – отправить данные из этой формы в нашу базу данных. Как вы уже знаете, для того чтобы записать данные в таблицу, используется команда INSERT языка SQL. Например:

Mysql> INSERT INTO Artifacts  
SET title='Петров';

Возникает вопрос, как можно воспользоваться такой командой (или любой другой командой SQL) в PHP скрипте. Для этого существует функция Mysql\_query().

11.1. Синтаксис Mysql\_query

ресурс 

Mysql\_query ( строка query [, ресурс link\_identifier])  
Mysql\_query() посылает SQL-запрос активной базе данных Mysql сервера, который определяется с помощью указателя link\_identifier (это ссылка на какое-то соединение с сервером Mysql). Если параметр link\_identifier опущен, используется последнее открытое соединение.

Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции Mysql\_connect() без параметров. Результат запроса буферизируется.  
  
Замечание: строка запроса НЕ должна заканчиваться точкой с запятой.  
  
Только для запросов SELECT, SHOW, EXPLAIN, DESCRIBE, Mysql\_query() возвращает указатель на результат запроса, или FALSE, если запрос не был выполнен. В остальных случаях Mysql\_query() возвращает TRUE, если запрос выполнен успешно, и FALSE – в случае ошибки. Значение, не равное FALSE, говорит о том, что запрос был выполнен успешно. Оно не говорит о количестве затронутых или возвращенных рядов. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда. Mysql\_query() также считается ошибочным и вернет FALSE, если у пользователя недостаточно прав для работы с указанной в запросе таблицей.  
  
Итак, теперь мы знаем, как отправить запрос на вставку строк в базу данных.  
  
Заметим, что в предыдущем примере элементы формы мы назвали именами полей таблицы. Поэтому они будут доступны в скрипте insert.php, обрабатывающем данные формы, как переменные вида  
  
**$\_POST['имя\_поля'].**

<?  
$conn=Mysql\_connect("localhost","nina","123");// устанавливаем  
// соединение  
$database = "book";  
$table\_name = "Artifacts";  
Mysql\_select\_db($database); // выбираем базу данных  
$list\_f = Mysql\_list\_fields($database,$table\_name);  
// получаем список полей в базе  
$n = Mysql\_num\_fields($list\_f); // число строк в результате  
// предыдущего запроса  
// составим один запрос сразу для всех полей таблицы  
$sql = "INSERT INTO $table\_name SET "; // начинаем создавать  
// запрос, перебираем все поля таблицы  
for($i=0;$i<$n; $i++){  
$name\_f = Mysql\_field\_name ($list\_f,$i); // вычисляем имя поля  
$value = $\_POST[$name\_f]; // вычисляем значение поля  
$j = $i + 1;  
$sql = $sql . $name\_f." = '$value'"; // дописываем в  
// строку $sql пару имя=значение  
if ($j <> $n) $sql = $sql . ", "; // если поле не  
// последнее в списке, то ставим запятую  
}  
// перед тем как записывать что-то в базу,  
// можно посмотреть, какой запрос получился  
//echo $sql;  
$result = Mysql\_query($sql,$conn); // отправляем запрос  
// выводим сообщение успешно ли выполнен запрос  
if (!$result) echo " Can't add ($table\_name) ";  
else echo "Success!<br>";  
?>

**Листинг 11.0.2. insert.php**  
  
Итак, задачу добавления данных с помощью web-интерфейса мы решили. Однако тут есть одна тонкость. При решении мы не учитывали тот факт, что значения некоторых полей (author, photo) должны браться из других таблиц (Persons, Images). Поскольку Mysql с внешними ключами не работает, этот момент остается на совести разработчиков системы, т.е. на нашей совести. Нужно дописать программу таким образом, чтобы была возможность вводить в такие поля правильные значения. Но мы делать этого не будем, поскольку задача лекции состоит в том, чтобы познакомить читателя с элементами технологии, а не в том, чтобы создать работающую систему. Кроме того, имеющихся у читателя знаний вполне достаточно, чтобы решить эту проблему самостоятельно. Мы же обратимся к другой задаче – отображение данных, хранящихся в базе данных СУБД Mysql.

12. [Отображение данных, хранящихся в Mysql](http://www.webmasterwiki.ru/MySQL/OtobrazhenieDannyx" \o "My SQL / Отображение Данных)

13. [Доступ к MySQL](http://www.webmasterwiki.ru/MySQL/DostupKMySQL" \o "My SQL / Доступ К My SQL)

14. [Безопасное взаимодействие My SQL и PHP](http://www.webmasterwiki.ru/MySQL/BezopasnoeVzaimodejjstvieMySQLiPHP" \o "My SQL / Безопасное Взаимодействие My SQ Lи PHP)

15. Литература

[[http://www.webmasterwiki.ru/themes/canary/icons/web.gif](http://mysql.ru/docs/man/)http://mysql.ru/docs/man/](http://mysql.ru/docs/man/" \o "Внешняя ссылка (откроется в новом окне)" \t "_blank)  
[[http://www.webmasterwiki.ru/themes/canary/icons/web.gif](http://www.intuit.ru/department/database/mysql/)http://www.intuit.ru/department/database/mysql/](http://www.intuit.ru/department/database/mysql/)

ALTA

<https://www.site-do.ru/db/db.php>

**Основы баз данных**

* [Урок 1. Понятие базы данных](https://www.site-do.ru/db/db1.php)
* [Урок 2. Структура базы данных](https://www.site-do.ru/db/db2.php)
* [Урок 3. Реляционные базы данных](https://www.site-do.ru/db/db3.php)
* [Урок 4. Концептуальная модель базы данных](https://www.site-do.ru/db/db4.php)
* [Урок 5. Преобразование концептуальной модели в реляционную](https://www.site-do.ru/db/db5.php)

**Предварительные установки**

* [Установка сервера Apache](https://www.site-do.ru/db/installation_apache.php)
* [Установка модуля PHP](https://www.site-do.ru/db/installation_php.php)
* [Установка сервера MySQL 5](https://www.site-do.ru/db/installation_mysql.php)
* [Установка интерфейса phpMyAdmin](https://www.site-do.ru/db/installation_phpmyadmin.php)

**Уроки SQL**

* [Урок 1. Создание базы данных и таблиц](https://www.site-do.ru/db/sql1.php)
* [Урок 2. Типы данных](https://www.site-do.ru/db/sql2.php)
* [Урок 3. Создание таблиц и наполнение их информацией](https://www.site-do.ru/db/sql3.php)
* [Урок 4. Выборка данных - оператор SELECT](https://www.site-do.ru/db/sql4.php)
* [Урок 5. Вложенные запросы](https://www.site-do.ru/db/sql5.php)
* [Урок 6. Объединение таблиц (внутреннее объединение)](https://www.site-do.ru/db/sql6.php)
* [Урок 7. Объединение таблиц (внешнее объединение)](https://www.site-do.ru/db/sql7.php)
* [Урок 8. Группировка записей и функция COUNT()](https://www.site-do.ru/db/sql8.php)
* [Урок 9. Редактирование, обновление и удаление данных](https://www.site-do.ru/db/sql9.php)
* [Урок 10. Встроенные функции](https://www.site-do.ru/db/sql10.php)
* [Урок 11. Итоговые функции, вычисляемые столбцы и представления](https://www.site-do.ru/db/sql11.php)
* [Урок 12. Строковые функции](https://www.site-do.ru/db/sql12.php)
* [Урок 13. Функции даты и времени](https://www.site-do.ru/db/sql13.php)
* [Урок 14. Функции форматирования даты и времени](https://www.site-do.ru/db/sql14.php)
* [Урок 15. Хранимые процедуры. Создание, использование и удаление.](https://www.site-do.ru/db/sql15.php)
* [Урок 16. Хранимые процедуры. Просмотр имеющихся процедур и их редактирование.](https://www.site-do.ru/db/sql16.php)
* [Урок 17. Хранимые процедуры. Оператор ветвления IF и переменные.](https://www.site-do.ru/db/sql17.php)
* [Урок 18. Хранимые процедуры. Циклы.](https://www.site-do.ru/db/sql18.php)

<http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/index-1.htm>

### Структуризированный язык запросов (SQL)

Учебное пособие

*Санкт-Петербургский Государственный институт  
точной механики и оптики (технический университет)   
Кафедра вычислительной техники   
В.В.Кириллов, Г.Ю.Громов*

[Предисловие](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/intro.htm)  
Глава 1. Реляционные базы данных и язык SQL  
    1.1. [Реляционная база данных](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/1-1.htm)  
    1.2. [Почему SQL?](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/1-2.htm)  
    1.3. [Таблицы SQL](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/1-3.htm)  
Глава 2. Запросы с использованием единственной таблицы  
    2.1. [О предложении SELECT](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-1.htm)  
    2.2. [Выборка без использования фразы WHERE](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-2-1.htm)  
        2.2.1. [Простая выборка](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-2-1.htm)  
        2.2.2. [Исключение дубликатов](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-2-1.htm#2-2-2)  
        2.2.3. [Выборка вычисляемых значений](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-2-3.htm)  
    2.3. [Выборка c использованием фразы WHERE](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-1.htm)  
        2.3.1. [Использование операторов сравнения](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-1.htm)  
        2.3.2. [Использование BETWEEN](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-2.htm)  
        2.3.3. [Использование IN](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-3.htm)  
        2.3.4. [Использование LIKE](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-4.htm)  
        2.3.5. [Вовлечение неопределенного значения](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-3-5.htm)  
    2.4. [Выборка с упорядочением](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-4.htm)  
    2.5. [Агрегирование данных](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-5-1.htm)  
        2.5.1. [SQL-функции](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-5-1.htm)  
        2.5.2. [Функции без использования фразы GROUP BY](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-5-2.htm)  
        2.5.3. [Фраза GROUP BY](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-5-3.htm)  
        2.5.4. [Использование фразы HAVING](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/2-5-4.htm)  
Глава 3. Запросы с использованием нескольких таблиц  
    3.1. [О средствах одновременной работы с множеством таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-1.htm)  
    3.2. [Запросы, использующие соединения](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-1.htm)  
        3.2.1. [Декартово произведение таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-1.htm)  
        3.2.2. [Эквисоединение таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-2.htm)  
        3.2.3. [Естественное соединение таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-3.htm)  
        3.2.4. [Композиция таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-4.htm)  
        3.2.5. [Тета-соединение таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-5.htm)  
        3.2.6. [Соединение таблиц с дополнительным условием](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-6.htm)  
        3.2.7. [Соединение таблицы со своей копией](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-2-7.htm)  
    3.3. [Вложенные подзапросы](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-1.htm)  
        3.3.1. [Виды вложенных подзапросов](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-1.htm)  
        3.3.2. [Простые вложенные подзапросы](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-2.htm)  
        3.3.3. [Использование одной и той же таблицы во внешнем и вложенном подзапросе](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-3.htm)  
        3.3.4. [Вложенный подзапрос с оператором сравнения, отличным от IN](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-4.htm)  
        3.3.5. [Коррелированные вложенные подзапросы](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-5.htm)  
        3.3.6. [Запросы, использующие EXISTS](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-6.htm)  
        3.3.7. [Функции в подзапросе](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-3-7.htm)  
    3.4. [Объединение (UNION)](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-4.htm)  
    3.5. [Реализация операций реляционной алгебры предложением SELECT](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-5.htm)  
    3.6. [Резюме](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/3-6.htm)  
Глава 4. Предложения модификации данных SQL  
    4.1. [Особенности и синтаксис предложений модификации](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-1.htm)  
    4.2. [Предложение DELETE](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-2.htm)  
        4.2.1. [Удаление единственной записи](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-2.htm#1)  
        4.2.2. [Удаление множества записей](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-2.htm#2)  
        4.2.3. [Удаление с вложенным подзапросом](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-2.htm#3)  
    4.3. [Предложение INSERT](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-3-1.htm)  
        4.3.1. [Вставка единственной записи в таблицу](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-3-1.htm)  
        4.3.2. [Вставка множества записей](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-3-2.htm)  
        4.3.3. [Использование INSERT...SELECT для построения внешнего соединения](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-3-3.htm)  
    4.4. [Предложение UPDATE](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-4.htm)  
        4.4.1. [Обновление единственной записи](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-4.htm#1)  
        4.4.2. [Обновление множества записей](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-4.htm#2)  
        4.4.3. [Обновление с подзапросом](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-4.htm#3)  
        4.4.4. [Обновление нескольких таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-4.htm#4)  
    4.5. [О конструировании предложений модификации](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/4-5.htm)  
Глава 5. О предложениях определения данных и оптимизации запросов  
    5.1. [Системный каталог](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-1.htm)  
    5.2. [Создание и уничтожение базовых таблиц](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-2.htm)  
    5.3. [О индексах и производительности](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-3.htm)  
    5.4. [Представления](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-4-1.htm)  
        5.4.1. [Создание и уничтожение представлений](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-4-1.htm)  
        5.4.2. [Операции выборки из представлений](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-4-2.htm)  
        5.4.3. [Обновление представлений](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-4-3.htm)  
        5.4.4. [Для чего нужны представления](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/5-4-4.htm)  
Глава 6. О других предложениях и конструкциях SQL  
    6.1. [Безопасность и санкционирование доступа](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/6-1.htm)  
    6.2. [Обработка транзакций](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/6-2.htm)  
    6.3. [Прикладное программирование](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/6-3.htm)  
[Литература](http://orloff.am.tpu.ru/data_base/Append/SQL_Kirillov/liter.htm)

JAVA SCRIPT

<http://www.twirpx.com/file/933699/>