

ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

О СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MYSQL

2.1. Особенности реляционной базы данных MySQL

На сегодняшний день большинство сайтов, порталов, на которые вы заходите ежедневно, хранит свой контент в базах данных. Это намного более эффективно по сравнению с созданием отдельных HTML-страниц. Любой контент, который вы видите на сайтах (пункты меню, статьи, комментарии пользователей, фотографии, списки друзей в соцсетях и многое другое), хранится в удобном, структурированном виде. Базы данных состоят из множества таблиц, которые связаны друг с другом. Запросы, которые выполняются к базам данных, отличаются лаконичностью, универсальностью (для большинства баз данных используется единый язык структурированных запросов SQL). По сравнению с обработкой файлов, SQL позволяет выполнять запросы гораздо быстрее, к тому же эффективно решается проблема одновременного доступа множества запросов к одним и тем же данным.

Приступив к изучению данного курса, студенты уже должны быть знакомы с основами работы в системах управления базами данных, например Microsoft Access или Microsoft SQLServer [16]. Восполнить пробелы или освежить знания по базам данным можно с помощью данных учебных пособий [7; 17].

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. Связка «PHP и MySQL» достаточно прочно рекомендовала себя при разработке небольших проектов.

Данные в базе структурированы в виде таблиц, модель представления которых называется реляционной (*relation* – от *англ.* отношение). Автором реляционной модели является Э. Кодд, который первым предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово про-

изведение) и показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение [19, с. 163]. Все таблицы в базе данных связаны между собой. Это позволяет устранить избыточность данных, дублирование.

Каждая запись в таблице должна быть уникальна, чтобы при проведении запросов обращение к записи было однозначным. Поле или набор полей, которые однозначно характеризуют запись, называют первичным ключом [17]. Ключи также используются для связи между таблицами (например, чтобы получить связанные данные из таблицы о том, какой пользователь какую оценку получил за тест, не нужно хранить все эти данные в одной таблице, все организуется через отношения). Для удобства в качестве ключевого поля чаще всего задают не несколько полей, а отдельное поле, все значения которого генерируются по порядку. Таким образом, СУБД может организовать без участия разработчика уникальные идентификаторы записей. В MySQL мы чаще всего будем называть это поле *id*, от слова *identification* (англ.) – идентифицировать. Данное поле обозначается как первичный ключ (Primary key), и в качестве значений по умолчанию устанавливается AUTO_INCREMENT (счетчик), что означает, что каждая новая запись будет иметь последовательное значение. И даже если будут удалены какие-то записи, счетчик не сойдет, а нумерация будет происходить последовательно с увеличением на единицу.

2.2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАБОТЫ С MySQL

Для удобства работы с MySQL разработано достаточно много удобных утилит, хотя с MySQL можно работать и из интерфейса обычной командной строки.

Самым распространенным решением для работы с MySQL является утилита phpMyAdmin, которая написана на PHP и является веб-приложением. PhpMyAdmin входит в состав Denwer, и для того чтобы запустить его, достаточно стартовать веб-сервер и обратиться в браузере по адресу <http://localhost/tools/phpmyadmin>. Однако следует понимать, что веб-приложение всегда будет работать несколько медленнее десктопных приложений. Но для наших лабораторных работ данное программное обеспечение будет вполне подходящим.

Для всех утилит по работе с MySQL характерны общие возможности, такие как:

- создание, редактирование, удаление, копирование баз данных;
- создание, редактирование, удаление, копирование таблиц;

- запуск и отладка SQL-запросов;
- просмотр содержимого баз данных и таблиц;
- экспорт / импорт баз данных, таблиц и отдельных записей;
- управление пользователями и правами пользователей к доступу по управлению базами данных;
- и др. сервисные функции.

Интерфейс phpMyAdmin представлен на рисунке 19.

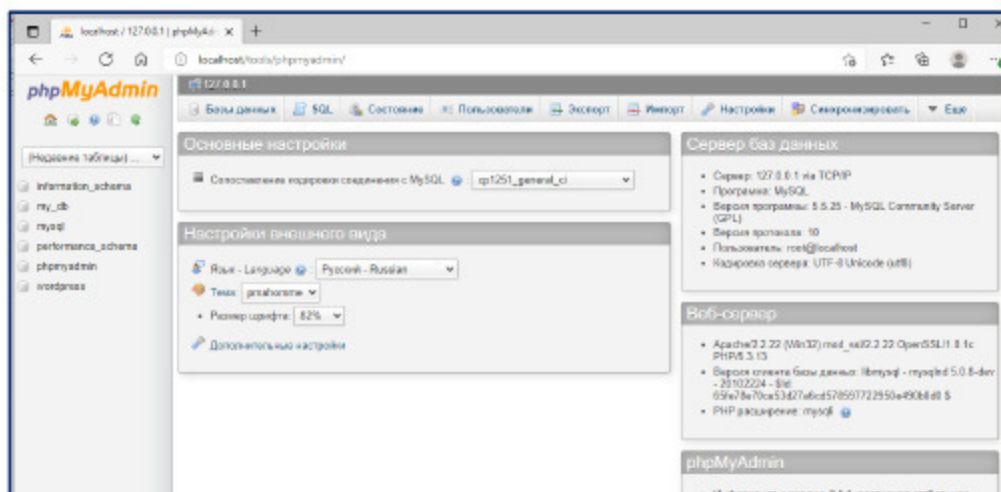


Рис. 19. Интерфейс phpMyAdmin

HeidiSQL [1] – это бесплатное, настольное (десктопное) приложение для работы с базами данных MySQL, MariaDB, Microsoft SQL, PostgreSQL и SQLite. Помимо всех базовых возможностей по работе с базами данными и таблицами, преимуществом heidiSQL является то, что ПО позволяет подключаться к различным серверам баз данных в одном окне; скорость выполнения работы значительно быстрее, чем в phpMyAdmin, работает с триггерами, процедурами SQL; есть возможность сохранения SQL-запросов; выполняет очень удобный поиск по всей таблице; позволяет легко импортировать / экспортировать данные в текстовые и табличные форматы; возможность написания запросов SQL с подсветкой синтаксиса и многое другое. Если вы серьезно заинтересовались веб-разработкой, оцените все возможности heidiSQL. Интерфейс показан на рисунке 20.

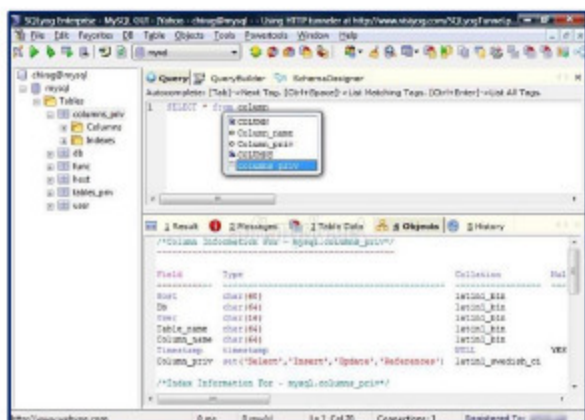


Рис. 21. Интерфейс SQLYOG

2.3. Создание базы данных, таблицы с использованием PHPMYADMIN

Рассмотрим вопросы создания базы данных, таблиц баз данных с помощью утилиты phpMyAdmin. Проектирование архитектуры базы данных приложения чаще всего предшествует написанию кода на PHP. Действительно, важно детально погрузиться в предметную область, создать группу таблиц, заполнить их предварительными сведениями и только затем переходить к коду обработки содержимого баз данных на PHP.

Для создания базы данных достаточно указать ее название в специальное поле в phpMyAdmin по адресу <http://localhost/tools/phpmyadmin/> (рисунок 22).

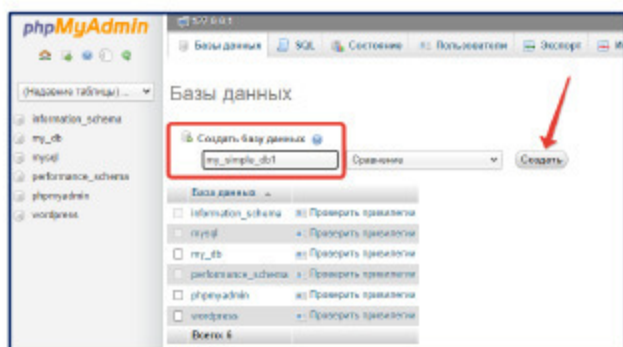


Рис. 22. Создание базы данных в phpMyAdmin

Имя базы данных может состоять из символов латинского алфавита (хотя в некоторых случаях допускается используется кириллицы, мы не рекомендуем это делать), цифр, знака доллара «\$», знака нижнего подчеркивания «_». Запрещено использовать

символ пробела « », точки «.», символы слешей «\», «/». Также ограничение накладывается на длину названия базы данных – она не должна превышать 64 символа.

Фактически база данных – это только имя, сами данные содержат таблицы баз данных. После создания базы данных в структуре каталогов вашего сервера формируется папка с названием вашей базы данных (рисунок 23).

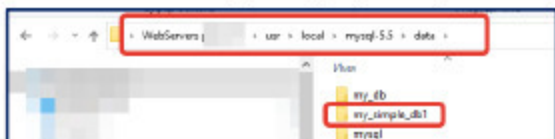


Рис. 23. Каталог созданной базы данных

Для созданной базы данных можно создать пользователя, которому будут доступны определенные привилегии по управлению базой данных. К таким привилегиям относят: управление данными в таблицах (выборка, добавление, редактирование, удаление данных); управление структурой базы данных (добавление / удаление новых таблиц, индексов, уничтожение базы данных и др.); администрирование базы данных. Создать пользователя можно на странице **Привилегии** для базы данных. Процедура не сложна: нужно придумать имя пользователя, пароль, подтвердить пароль, проверить или отметить галочкой базу данных, для которой назначаются привилегии и отметить галочкой те опции, которые пользователь сможет выполнять (рисунок 24).

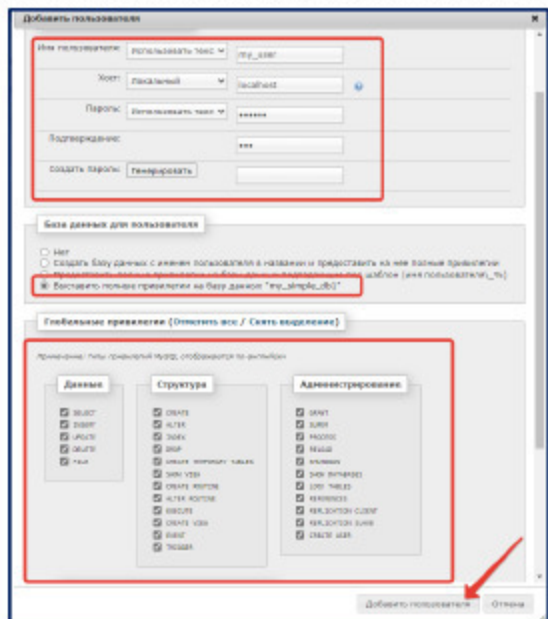


Рис. 24. Создание пользователя и определение его привилегий

После этого можно перейти к созданию таблицы в базе данных. На странице с выбранной базой данных есть поле, где можно задать имя таблицы и установить количество полей (рисунок 25). Если заранее вы не можете знать, сколько полей в таблице у вас будет, то можно поставить любое число – лишнее phpMyAdmin уберет, а если нужно будет больше полей, то их можно дополнить.

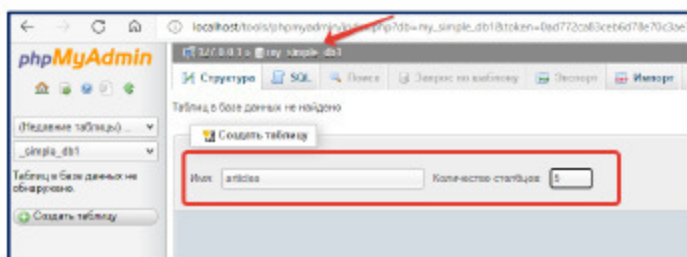


Рис. 25. Создание таблицы в базе данных

После нажатия на кнопку Ok, появится форма с созданием новых полей, указанием их типов, длины и некоторых иных параметров. Для начала рассмотрим, какие типы полей есть в MySQL. Тип поля определяет ограничения на данные и операции, которые могут быть произведены с данными этого типа. Как и в системах программирования, в MySQL тип поля определяет внутренний формат представления данных. Так, например, тип int может содержать только целочисленный набор данных, varchar – строки ограниченной длины. И очевидно, что действия, которые можно производить с целыми числами – совсем не то же самое, что действия со строками.

К целочисленным типам данных в MySQL относят:

- **TINYINT.** Диапазон значений от -127 до 128, либо 0 до 255, в зависимости от того, может ли это поле быть отрицательным.
- **SMALLINT.** Диапазон значений: -32 768 до 32 767, либо от 0 до 65 535.
- **MEDIUMINT.** Диапазон значений: от -8 388 608 до 8 388 607, либо от 0 до 16 777 215.
- **INT.** Диапазон значений: от -2 147 483 648 до 2 147 483 647, либо от 0 до 4 294 967 295.
- **BIGINT.** Диапазон значений: от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, либо от 0 до 18 446 744 073 709 551 615.

К вещественным типам данных относят:

- **FLOAT.** Точность одинарная, то есть число знаков после запятой может быть не более 24-х (для двоичного представления). Диапазон значений: от $-3,402823466E+38$ до $-1,175494351E-38, 0$, и от $1,175494351E-38$ до $3,402823466E+38$.
- **DOUBLE.** Двойная точность. Количество знаков после запятой может составлять до 53-х (для двоичного представления). Допустимые значения: от $-1,7976931348623157E+308$ до $-2,2250738585072014E-308, 0$, и от $2,2250738585072014E-308$ до $1,7976931348623157E+308$. Данный тип используется, если нужны действительно огромные числа.
- **DECIMAL.** Это число, похожее на тип DOUBLE, но хранится оно в виде строки. И, фактически, интервал допустимых значений определяется наличием знака "-" и ".". Если эти знаки отсутствуют, то допустимый интервал такой же, как и у DOUBLE.

К строковому типу данных относят:

- **TEXT.** Максимальная длина 65 535 символов. Самый используемый вариант при хранении текстовых данных.
- **TINYTEXT.** Текст с длиной от 0 до 255 символов.
- **MEDIUMTEXT.** Текст с длиной от 0 до 16 777 215 символов.
- **LONGTEXT.** Текст с длиной от 0 до 4 294 967 295 символов.
- **VARCHAR.** Текст переменной длины от 0 до 255 символов.
- **CHAR.** Длина фиксированная (независимо от количества переданных символов). Диапазон составляет от 0 до 255 символов. При передаче данных меньше 255 символов в конце к данным дописываются пробелы, чтобы длина строки достигла заданного размера.

К типу даты / времени относят:

- **DATE.** Хранит дату. Формат следующий: YYYY-MM-DD (год, месяц, день). Например, такое значение будет удовлетворять этому полю: 2021-05-02.
- **DATETIME.** Хранит дату и время. Формат следующий: YYYY-MM-DD HH:MM:SS (год-месяц-день час-минута-секунда). Например: 2021-04-21 09:41:22
- **TIMESTAMP.** Хранит дату и время. Имеет следующие форматы: YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, YYMMDD.
- **TIME.** Хранит время. Формат: HH:MM:SS. Например: 09:21:55.
- **YEAR.** Хранит дату (год). Форматы: YY, YYYY.

Самыми популярными типами данных из всех перечисленных выше являются int, float, varchar, text, date, datetime.

Некоторым полям можно задавать атрибуты, например числовое поле делать автоматическим счетчиком или определять, что значения определенного поля будут неотрицательными:

- Атрибут **AUTO_INCREMENT** – генерирует новое порядковое значение для строк;
- Атрибут **UNSIGNED** – данное числовое значение будет неотрицательным.

Подробную информацию о типах данных MySQL можно получить на ресурсе [2].

Определившись с типами данных, мы можем создать таблицу. Таблица **articles** будет содержать сведения о публикуемых статьях (подробнее, как создать функционирующий блог, мы рассмотрим в рамках лабораторного практикума).

Первое поле по традиции назовем **id** (от слова *identify* – идентифицировать). Оно предназначено для хранения уникального номера для каждой статьи. Тип этого поля **INT**, длиной 11 знаков (значит хранить можно максимально 11-значное десятичное число). Также ему добавляется атрибут **AUTO_INCREMENT**. Указывается, что поле является первичным ключом **PRIMARY**.

Второе поле предназначено для хранения названия статьи, назовем его **title**. Так как заголовок статьи вряд ли будет занимать больше двух строк текста, то его тип можно определить, как **VARCHAR**, длиной 255 символов.

Третье поле будет хранить краткий анонс статьи (описание), можно его назвать **description**. Его можно определить типа **TEXT**. Данное поле не имеет ограничений.

Четвертое поле будет хранить полный текст статьи **text**. Его тип также будет **TEXT**.

Пятое поле будет хранить ФИО автора статьи, будет называться **author**. Тип поля можно сделать **VARCHAR**, длиной до 255 символов.

Создание структуры таблицы **articles** показано на рисунке 26.

Скриншот интерфейса создания таблицы в MySQL. Вверху введено имя таблицы 'articles' и нажата кнопка 'ОК'. В центре таблица с заголовком 'Структура' и колонками: Имя, Тип, Длина/значения, По умолчанию, Сравнение, Атрибуты, Null, Индекс, A.I. Комментари. В строке для поля 'id' в колонке 'Атрибуты' выбран 'PRIMARY', а в колонке 'A.I.' – 'AUTO_INCREMENT', на что указывает красная стрелка. Другие поля: 'title' (VARCHAR, 255), 'description' (TEXT), 'text' (TEXT), 'author' (VARCHAR, 255).

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс	A.I.	Комментари
id	INT	11	Нет			<input checked="" type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
title	VARCHAR	255	Нет			<input type="checkbox"/>			
description	TEXT		Нет			<input type="checkbox"/>			
text	TEXT		Нет			<input type="checkbox"/>			
author	VARCHAR	255	Нет			<input type="checkbox"/>			

Рис. 26. Создание структуры таблицы **articles**

В любой момент можно внести изменения в структуру таблицы – добавить новые поля или удалить / отредактировать имеющиеся.

Мы можем добавить поле для хранения даты создания статьи. Назовем его `date_created`, оно будет типа `DATETIME`. Чтобы его внести, нужно перейти на вкладку **Структура** для нашей таблицы и поставить, после какого поля мы хотим добавить новое поле (рисунок 27). После этого поле заполняется аналогичным образом.

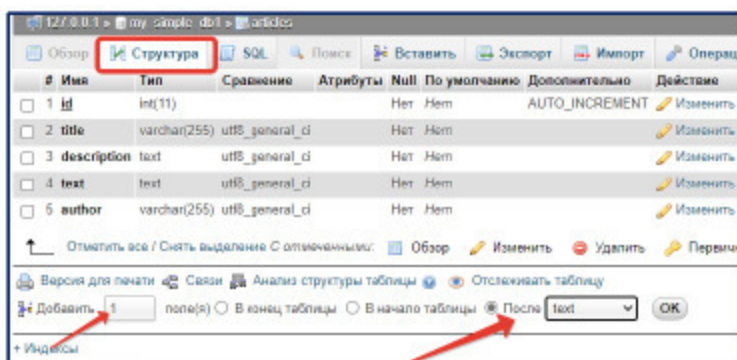




Рис. 27. Процедура добавления нового поля в таблицу

На вкладке **Структура** также можно любое поле отредактировать (нажать на карандаш ) или удалить (нажать на ).

Сейчас у нас есть таблица, но в ней нет данных. Не начинайте писать код на PHP не заполнив нужной таблицы. Как минимум две записи нужно внести в таблицу. Дело в том, что данные из таблиц будут формироваться в циклах на PHP (как это делается, мы рассмотрим позднее). Если у вас будет в таблице только одна запись, то вы не сможете протестировать работу такого цикла – одна запись может выводиться по одной выборке и без цикла. А вот если в таблице у вас две записи и при организации кода на PHP вы также получили две записи, то можно говорить, что все сделано правильно.

Чтобы вставить данные в таблицу, нужно перейти на вкладку **Вставить**. По умолчанию phpMyAdmin сразу предлагает вставить две записи. Заполняются только поля столбца **Значения**. Здесь следует отметить, что если мы установили для поля `id` атрибут `AUTO_INCREMENT`, то задавать его значения не нужно (оставляйте это поле пустым). На рисунке 28 показан процесс внесения данных одной строки таблицы `articles`.

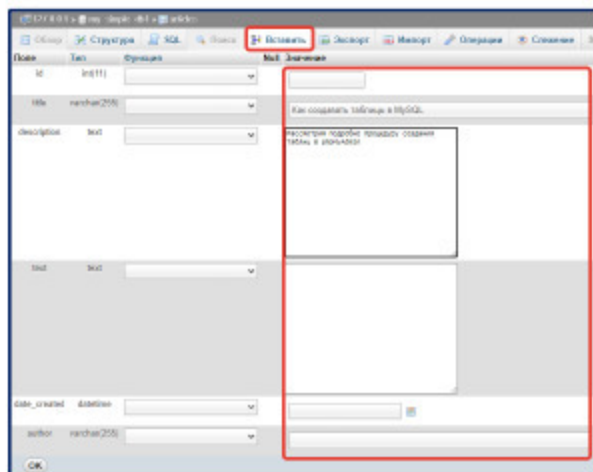


Рис. 28. Процедура внесения строки в таблицу articles

Следует отметить, что работать с phpMyAdmin просто и интуитивно понятно, если вы уже знакомы с системами управления базами данных. Многие элементы подписаны, есть подсказки, главное – быть внимательным.