

**БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»**

Кафедра информационно-вычислительных систем

ИНФОРМАЦИОННЫЕ РЕСУРСЫ

**Практикум
к лабораторным занятиям для студентов специальности
1-26 03 01 «Управление информационными ресурсами»**

В двух частях

**Часть 2
Технологии работы с клиент-серверными СУБД
(на примере СУБД MySQL)**

Гомель 2011

УДК 004
ББК 32.973
И 74

Авторы-составители: Т. А. Заяц, ассистент кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации;
В. М. Заяц, ведущий инженер-электроник информационно-вычислительного центра СОАО «Гомелькабель»

Рецензенты: О. А. Кравченко, канд. физ.-мат. наук, доцент кафедры информационных технологий Гомельского государственного технического университета им. П. О. Сухого;
Е. А. Левчук, канд. техн. наук, доцент кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации

Рекомендован к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 5 от 10 июня 2008 г.

Информационные ресурсы : практикум к лабораторным занятиям для И 74 студентов специальности 1-26 03 01 «Управление информационными ресурсами». В 2 ч. Ч. 2. Технологии работы с клиент-серверными СУБД (на примере СУБД MySQL) / авт.-сост. : Т. А. Заяц, В. М. Заяц. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2011. – 124 с.
ISBN 978-985-461-733-6

УДК 004
ББК 32.973

ISBN 978-985-461-733-6

© Учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2011

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Информационные ресурсы представляют собой документы и отдельные массивы документов в информационных системах. Знание состояния информационных ресурсов, умение работать на информационном рынке и использовать полученную информацию в своей деятельности – обязательное требование, предъявляемое к высококвалифицированному специалисту.

Специальная дисциплина «Информационные ресурсы» ставит своей целью не только ознакомить студентов с мировыми и отечественными информационными ресурсами, но и сформировать практические навыки разработки информационных ресурсов в корпоративных информационных системах и их использования на высоком профессиональном уровне.

Практикум состоит из трех разделов, в каждом из которых приводятся теоретические сведения, необходимые для выполнения лабораторных работ и индивидуальных заданий данного практикума.

В первом разделе рассматриваются возможности системы управления базами данных (СУБД) MySQL – одной из наиболее популярных СУБД клиент-серверного типа. Даются основные характеристики СУБД MySQL, указываются программные продукты, входящие в состав дистрибутива MySQL.

Второй раздел посвящен изучению языка SQL-запросов к базам данных. В разделе приводятся важнейшие конструкции языка SQL, используемые для формирования структуры базы данных и манипулирования информацией, хранящейся в ней. Завершается раздел несколькими лабораторными работами по созданию баз данных и выполнению запросов к ним.

В третьем разделе приводятся теоретические сведения по использованию средств языка веб-программирования PHP для взаимодействия с сервером MySQL. Рассматривается синтаксис основных функций языка PHP; приводятся листинги программ на PHP, содержащие примеры установки соединения с сервером баз данных, создания запросов к базе данных, обработки и отображения результатов запросов к серверу, закрытия соединения с сервером MySQL. В разделе имеются лабораторные работы, позволяющие проверить степень усвоения полученного материала.

Завершается третий раздел описанием веб-проекта «Библиотека». В указанном проекте выполняется предварительная разработка реляционной модели проектируемой учебной базы данных. Затем применяется интерфейс административной программы `mysqladmin` для подключения к серверу баз данных MySQL и создания таблиц в проектируемой базе данных. Веб-интерфейс работы с учебной базой данных реализуется в виде скриптов (программ) на языке PHP для добавления, удаления, обновления и поиска записей в таблицах реляционной базы, расположенной на сервере MySQL.

Практикум содержит теоретические сведения по курсу «Информационные ресурсы», задания лабораторных работ с примерами и задания для самостоятельной работы.

1. ОСНОВНЫЕ СВЕДЕНИЯ О СУБД MySQL

1.1. ПРЕИМУЩЕСТВА ХРАНЕНИЯ ДАННЫХ В БАЗАХ

Существуют два основных способа хранения данных: в двумерных (линейных) файлах и в базах данных. Двумерный файл может иметь множество форматов, но в общем случае под двумерным (flat) файлом понимается простой текстовый файл.

При использовании двумерных файлов могут возникать следующие проблемы:

- когда размер файлов становится большим, работа с ними существенно замедляется, а в режиме произвольного доступа сопряжена со значительной перегрузкой системы;
- поиск конкретной записи или группы записей затруднен;
- конкурирующий доступ (обращение к файлу одновременно нескольких пользователей) может породить блокировку файла;
- кроме ограничений, налагаемых правами доступа к файлам, не существует никакого способа обеспечения различных уровней доступа к данным.

Если приходится иметь дело с достаточно большим объемом информации, лучше пользоваться базами данных. Хотя двумерные файлы находят достаточно широкое применение, более рациональный способ хранения и обработки данных – база данных, созданная средствами системы управления базами данных.

Системы управления базами данных решают все проблемы, возникающие при использовании двумерных файлов:

- обеспечивают более быстрый доступ к данным, чем двумерные файлы (СУБД MySQL обладает одними из самых высоких показателей производительности среди всех СУБД);
- позволяют легко создавать запрос для извлечения наборов данных, соответствующих определенным критериям;

- обладают встроенными механизмами обработки конкурирующих обращений (сервер обеспечивает контроль параллельного доступа, что не позволяет двум пользователям одновременно модифицировать одну и ту же запись);

- обеспечивают произвольный доступ к данным;
- обладают встроенными системами определения прав доступа.

Средства эффективного хранения и выборки больших объемов информации внесли огромный вклад в успешное развитие Интернета. Работа таких известных сайтов, как *Yahoo*, *Amazon* и *Ebay*, в значительной степени зависит от надежности баз данных, хранящих громадные объемы информации.

Правильная организация базы данных обеспечивает более быстрые и гибкие возможности выборки данных. Она существенно упрощает реализацию средств поиска и сортировки, а проблемы прав доступа к информации решаются при помощи средств контроля за привилегиями, присутствующими во многих системах управления базами данных. Кроме того, упрощаются процессы репликации и архивации данных.

Одной из наиболее популярных систем управления базами данных является СУБД MySQL, реляционная СУБД типа клиент-сервер.

Создателем СУБД MySQL является Майкл Видениус из компании «ТсХ DataKonsultAB» (Стокгольм, Швеция). Начиная с 1995 г. MySQL стала одной из самых распространенных СУБД в мире, что отчасти обусловлено ее скоростью, надежностью и гибкой лицензионной политикой. Компания «ТсХ» заявляет, что СУБД MySQL используется примерно на 500 тыс. серверов (по состоянию на 2006 г.).

1.2. СОСТАВ ДИСТРИБУТИВА СУБД MySQL

Дистрибутив MySQL можно загрузить из Интернета (<http://www.mysql.ru> или <http://www.mysql.com/downloads/index.html>). В состав дистрибутива MySQL входят следующие программные продукты:

- *SQL-сервер*, обеспечивающий доступ к базам данных (принимает запросы клиентов, поступающие по сети, и осуществляет доступ к содержимому базы данных для предоставления информации, которую запрашивают клиенты);
- *клиентская программа* доступа к серверу, которая осуществляет подключение к серверу и передает запросы на сервер;
- *административные и сервисные программы*, помогающие работать с СУБД.

Примерами административных и сервисных программ являются *mysqldump*, позволяющая делать дампы (архив) базы данных в файл и восстанавливать его обратно, и *mysqladmin*, позволяющая проверять состояние сервера баз данных и выполнять административные функции.

1.3. ОСНОВНЫЕ ПРЕИМУЩЕСТВА СУБД MySQL

Среди преимуществ СУБД MySQL выделяются следующие:

- *Быстродействие*. СУБД MySQL является одной из самых быстрых баз данных из имеющихся на современном рынке.

- *Простота использования*. СУБД MySQL является высокопроизводительной и относительно простой в использовании, которую значительно проще установить и администрировать, чем многие большие системы.

- *Поддержка языка запросов*. СУБД MySQL «понимает» команды языка SQL, языка запросов, нашедшего применение во всех современных СУБД.

- *Возможности обработки*. Количество строк в таблицах может достигать 50 млн. Компания-разработчик утверждает, что использует СУБД MySQL с 1996 г. на сервере с более 40 базами данных, которые содержат 10 тыс. таблиц, из которых свыше 500 имеют более 7 млн строк.

- *Возможности доступа*. Сервер позволяет одновременно подключаться неограниченному количеству пользователей, работающих с базой данных. Доступ к серверу СУБД MySQL можно осуществить в интерактивном режиме.

- *Поддержка интерфейса ODBC*. Доступ к базам данных СУБД MySQL возможен с помощью приложений, поддерживающих обобщенный интерфейс ODBC (Open Data Bases Connectivity – открытое взаимодействие с базами данных), который может использоваться для одновременного соединения с различными СУБД.

- *Взаимодействие и безопасность*. СУБД MySQL предназначена для работы в сети и может быть доступна через Интернет. Таким образом, обращаться к базе данных и работать с данными можно из любой точки земного шара. Но при этом СУБД MySQL снабжена развитой системой защиты от несанкционированного доступа. Для обеспечения дополнительной защиты СУБД MySQL поддерживает криптируемые соединения с использованием протокола SSL.

• *Аппаратная совместимость.* СУБД MySQL работает как под управлением различных версий UNIX, так и под управлением таких систем, как Windows и OS/2. СУБД MySQL может работать на домашних ПК и на мощных серверах.

• *Малый размер.* СУБД MySQL имеет ограниченный размер по сравнению с огромным дисковым пространством, необходимым большинству коммерческих СУБД (сервер Apache – 23 Мбайта, СУБД MySQL – 120 Мбайт, PHP – 150 Мбайт).

• *Работоспособность и цена.* СУБД MySQL является проектом Open Source, легкодоступным по условиям General Public License (GPL). Это означает, что СУБД MySQL распространяется бесплатно для большинства домашних пользователей или для некоммерческого использования. В противном случае необходимо приобретение лицензии, стоимость которой составляет 190 евро. Дистрибутив СУБД MySQL легкодоступен. Для этого достаточно скачать его с сайта в Интернете, воспользовавшись интернет-браузером.

1.4. ТЕРМИНОЛОГИЯ СУБД

Система управления базами данных MySQL классифицируется как реляционная система управления базами данных или RDBMS (relational database management system). Эта аббревиатура разделяется на части следующим образом:

- слово «реляционная» или R (relational) обозначает популярную разновидность СУБД, которые поддерживают базы с реляционной моделью организации данных;
- база данных или DB (database) – это именованная совокупность информации из некоторой предметной области, структурированная определенным способом;
- система управления или MS (management system) является компьютерной программой, позволяющей пользователю создавать, поддерживать базы данных и управлять ими; вставлять, выбирать, модифицировать и удалять записи.

1.5. ПОНЯТИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

База данных представляет собой централизованное хранилище, предназначенное для накопления, доступа и обработки данных под управлением прикладной программы.

Реляционная база данных построена на основе реляционной модели, которая представляется как совокупность простейших двумерных таблиц, связанных друг с другом логическими связями. База данных реляционного типа характеризуется следующими свойствами:

- данные в базе объединены в таблицы;
- каждая таблица состоит из строк и столбцов;
- каждому столбцу таблицы присваивается имя, которое должно быть уникальным для этой таблицы;
- строка таблицы является ее записью;
- всем строкам таблицы соответствует один и тот же набор столбцов, хотя любая строка таблицы может содержать пустые значения в некоторых столбцах;
- все строки таблицы обязательно отличаются друг от друга хотя бы одним значением, что позволяет однозначно определить любую строку такой таблицы;
- в поле на пересечении строки и столбца любой таблицы всегда имеется только одно значение данных и никогда не должно быть множества значений;
- в таблице выделяется первичный ключ – столбец, значения в котором не повторяются;
- логическая связь между таблицами осуществляется по равенству значений ключевых столбцов таблиц-отношений.

Рассмотрим пример того, как реляционная база данных объединяет данные в таблицы и связывает данные из одной таблицы с данными другой таблицы. Предположим, что при сопровождении веб-узла, представляющего интернет-магазин, необходимо поддерживать контакт с компаниями, которые хотят разместить на сайте свои предложения о продаже товаров. Для этого они предоставляют информацию о своей продукции. Всякий раз, когда посетитель данного веб-узла желает приобрести какой-либо товар, оформляется заказ на приобретение товара. За помощь в реализации продукции владелец сайта получает от торговой компании небольшой гонорар.

Для хранения всей вышеуказанной информации используется реляционная база данных, информация в которой хранится в семи таблицах (рисунок 1). Рассмотрим таблицу, описывающую продавца товара (таблица 1). Таблица Торговая компания содержит столбцы Наименование торговой компании, Адрес компании, Телефон. В качестве ключевого используется столбец Номер компании, в котором каждой торговой компании присвоен уникальный номер.

Примечание – Для наглядности имя ключевого столбца в каждой таблице подчеркнуто.

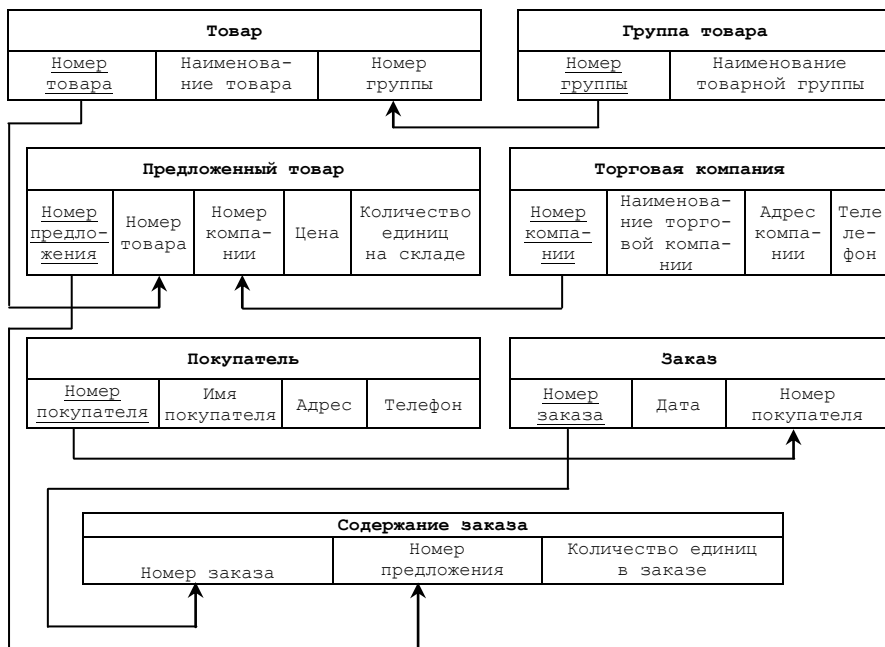


Рисунок 1 – Структура базы данных интернет-магазина

Таблица 1 – Таблица Торговая компания с ключевым полем

Торговая компания			
<u>Номер компании</u>	Наименование торговой компании	Адрес компании	Телефон

Вторая таблица Товар содержит информацию о товарах, реализуемых через интернет-магазин (таблица 2). В состав таблицы входят столбцы с именами Номер товара, Наименование товара, Номер группы. Столбец Номер группы используется для связи с таблицей Группа товара (таблица 3), в которой перечислены основные товарные группы (например, Бытовая техника, Компьютерная техника, Мебель и т. д.).

Таблица 2 – Таблица Товар

Товар		
<u>Номер товара</u>	Наименование товара	Номер группы

Таблица 3 – Таблица Группа товара

Группа товара	
<u>Номер группы</u>	Наименование товарной группы

Так как торговые компании могут предлагать к реализации одни и те же товары с одинаковыми названиями, но отличающиеся ценой, целесообразно наличие таблицы Предложенный товар (таблица 4), в которую помимо полей Номер предложения (ключевой столбец), Цена и Количество единиц на складе включены поля Номер товара и Номер компании для обеспечения связи с таблицами Товар и Торговая компания.

Таблица 4 – Таблица Предложенный товар

Предложенный товар				
<u>Номер предложения</u>	Номер товара	Номер компании	Цена	Количество единиц на складе

Посетитель сайта, изучив предложенный список товаров, может принять решение о приобретении какого-либо товара. Для этого ему необходимо оформить заказ, т. е. указать наименование покупаемого товара и его количество. Покупатель в одном заказе может оформить покупку нескольких товаров. Для хранения информации о заказах используются таблицы Заказ и Содержание заказа (таблицы 5, 6). Обратите внимание, таблица Содержание заказа не содержит столбца с именем покупателя, а содержит столбец Номер покупателя, который нужен для связи с таблицей Покупатель, хранящей информацию о всех покупателях (таблица 7).

Таблица 5 – Таблица Заказ

Заказ		
Номер заказа	Дата	Номер покупателя

Таблица 6 – Таблица Содержание заказа

Содержание заказа		
Номер заказа	Номер предложения	Количество единиц в заказе

Таблица 7 – Таблица Покупатель

Покупатель			
Номер покупателя	Имя покупателя	Адрес	Телефон

Показанная на рисунке 1 структура базы данных позволяет дать ответы на любые вопросы, касающиеся информации о работе интернет-магазина. Ответы на некоторые вопросы можно получить, пользуясь информацией из одной таблицы. Например, для определения количества компаний, с которыми вы работаете, достаточно посчитать строки в таблице Торговая компания, для подсчета количества заказов за определенный период времени – в таблице Заказ. Другие вопросы будут посложнее, потому как для получения ответов на них необходимо обратиться не к одной, а сразу к нескольким таблицам.

Например, для определения списка товаров, приобретенных покупателями за указанную дату, потребуется просмотреть пять таблиц. В таблице Заказ нужно найти все записи, в которых дата совпадает с требуемой, по номеру заказа – соответствующие ему записи в таблице Содержание заказа и получить записи с номерами предложений. Затем произвести поиск всех записей в таблице Предложенный товар, в которых номер предложения совпадает с номером предложения, найденным в таблице Содержание заказа. Полученный список будет содержать такие столбцы, как Номер товара, Номер компании и Цена. По номеру товара можно обратиться к таблице Товар и получить названия товаров, проданных за указанную дату, а по номеру компании – получить из таблицы Торговая компания название компании, реализующей указанный товар.

На первый взгляд все это непросто, но на самом деле нужно только грамотно создать реляционную модель базы данных, связав таблицы друг с другом, а выборку необходимой информации из базы поможет сделать СУБД.

1.6. ТЕРМИНОЛОГИЯ ЯЗЫКА ЗАПРОСОВ SQL

Для работы с базами данных используется структурированный язык запросов (SQL – Structured Query Language). Язык SQL «структурирован» в том отношении, что придерживается определенного набора правил. Он близок к «естественному» языку, например:

```
SELECT name FROM company WHERE number_company=2
```

Данный запрос напоминает фразу на «ломаном» английском: «Выберите название из компаний, где номер компании равен 2».

Язык SQL является основополагающим инструментом, необходимым для взаимодействия с СУБД MySQL. SQL-команды реализуют разнообразные возможности для взаимодействия с базой данных, а именно:

- определение структуры данных (определение конструкций, используемых при хранении данных);
- выборка данных (загрузка данных из базы и их представление в формате, удобном для вывода);
- обработка данных (вставка, обновление и удаление информации);
- контроль доступа (возможность разрешения (запрета) выборки, вставки, обновления и удаления данных на уровне отдельных пользователей);
- контроль целостности данных (сохранение структуры данных при возникновении таких проблем, как параллельные обновления или системные сбои).

В настоящее время SQL является стандартом работы всех основных СУБД с базами данных.

2. ОСНОВНЫЕ ПРИЕМЫ РАБОТЫ В СУБД MySQL

2.1. ИСПОЛЬЗОВАНИЕ КОМАНДНОЙ СТРОКИ ДЛЯ РАБОТЫ С БД MySQL

2.1.1. Варианты подключения к серверу MySQL

СУБД MySQL использует традиционную архитектуру клиент-сервер, поэтому, работая с СУБД MySQL, пользователь реально работает с двумя программами:

- программа сервера базы данных MySQL, расположенная на сервере ЛВС и хранящая базу данных (принимает запросы клиентов, поступающие по сети, и осуществляет доступ к содержимому базы данных для предоставления информации, которую запрашивают клиенты);
- клиентская программа mysql (осуществляет подключение к серверу, передает запросы на сервер и отображает результаты их выполнения), расположенная на рабочей станции.

Примечание – Поясним отличие обозначений MySQL и mysql. Во избежание разночтений необходимо подчеркнуть, что прописными буквами (MySQL) обозначают программу сервера СУБД MySQL, а строчными (mysql) – клиентскую программу.

Действия по подключению к серверу MySQL несколько отличаются в зависимости от того, где установлен сервер баз данных. Дистрибутив MySQL устанавливает сервер MySQL либо на компьютер, на котором работает пользователь (рисунок 2, а), либо на какой-либо другой компьютер, например, сервер ЛВС или сервер Интернета (рисунок 2, б).

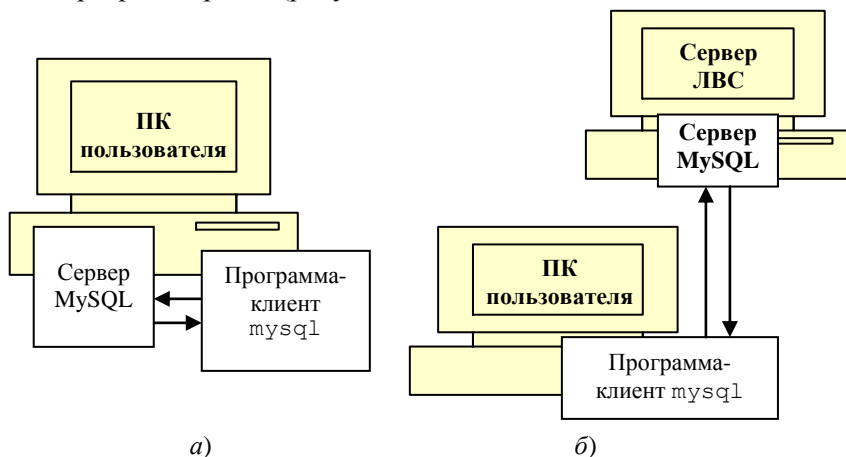


Рисунок 2 – Варианты подключения к серверу MySQL

2.1.2. Запуск программы-клиента mysql, установка соединения с сервером MySQL и завершение работы с ним

В данном пункте рассматривается технология работы с сервером MySQL. Взаимодействие с сервером осуществляется через клиентскую программу mysql, входящую в дистрибутив MySQL. С помощью этой программы можно устанавливать соединение с MySQL-сервером, выполнять SQL-запросы и просматривать результаты этих запросов. Программу – консольный клиент mysql часто называют «терминальным монитором» или просто «монитором».

Для подключения к серверу необходимо запустить программу-клиент mysql из своей оболочки. Для этого нужно выполнить следующее:

- Вызвать командную строку Пуск/Программы/Стандартные/ Командная строка.

После запуска командной строки откроется окно (рисунок 3) с приглашением-подсказкой операционной системы примерно следующего вида:

```
C:\Documents and settings\233m1>_
```

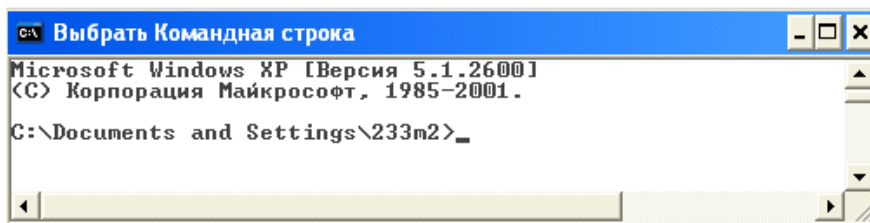


Рисунок 3 – Окно командной строки

Примечание – Для задания правильного отображения текста на русском языке в окне командной строки нужно установить шрифт `lucida console`, воспользовавшись контекстно-зависимым меню строки заголовка окна.

- В окне командной строки вводимые символы отображаются в кодировке 866 (по умолчанию). Чтобы иметь возможность в дальнейшем правильно отображать вводимую информацию, следует задать кодировку 1251 следующей командой:

chcp 1251

- Выполнить переход в корневой каталог диска C с помощью команды `cd \` (рисунок 4).

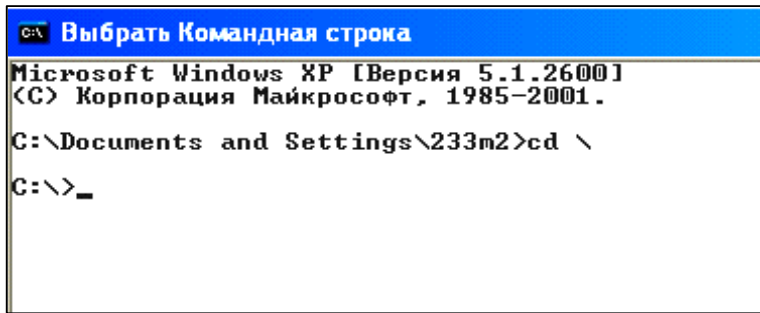


Рисунок 4 – Результат выполнения команды перехода в корневой каталог

- Перейти в каталог, в котором находится файл программы-клиента `mysql.exe`. Для этого в окне командной строки нужно набрать

```
cd C:\Program Files\MySQL\MySQL Server 5.0\bin
```

и нажать клавишу *Enter*. В ответ будет выведено приглашение командной строки (рисунок 5).

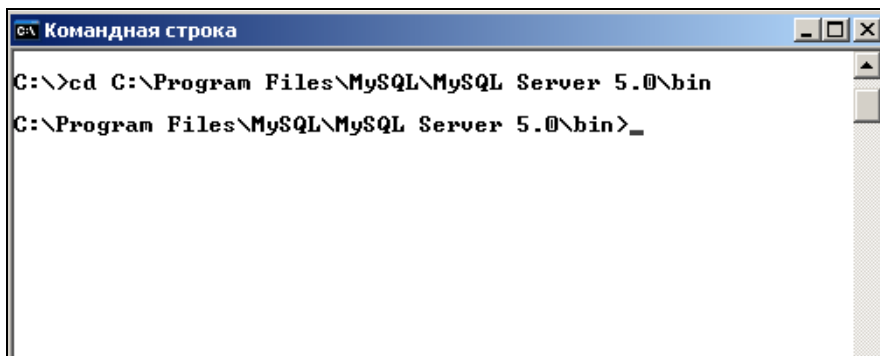


Рисунок 5 – Окно с приглашением-подсказкой в каталоге с программой `mysql`

- Теперь можно приступить к подключению к серверу баз данных MySQL. Вызов подключения выполняется с помощью команды

```
mysql -hhost_name -user_name -ppassword
```

Параметры команды обозначают следующее:

`-hhost_name` указывает имя серверного узла (хоста), к которому вы хотите подключиться и на котором находится сервер MySQL (если сервер баз данных работает на том же компьютере, что и программа-клиент, этот параметр всегда имеет имя `localhost` и его можно не писать вообще);

`-user_name` указывает имя пользователя, зарегистрированное в СУБД MySQL (по умолчанию при первоначальной установке сервера MySQL, в базе данных имеется единственный пользователь с именем `root`);

`-ppassword` указывает пароль доступа к базе данных.

Примечание – Между ключами и именами в команде пробела нет.

Возможны два способа подключения к серверу MySQL:

1. Для подключения к серверу баз данных, установленному на том же компьютере, на котором работает пользователь (см. рисунок 2, а), достаточно задать команду

```
mysql -hlocalhost -uroot -p
```

и нажать клавишу *Enter* (рисунок 6).

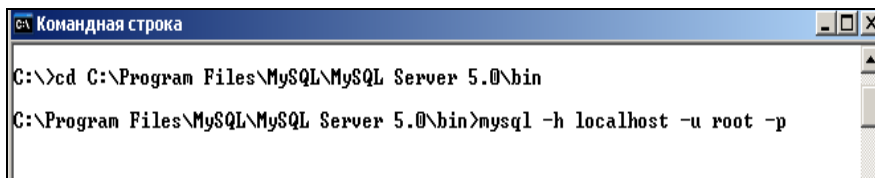


Рисунок 6 – Окно с командой подключения к серверу MySQL

В ответ на введенную команду пользователь получит команду с просьбой ввода пароля (рисунок 7).

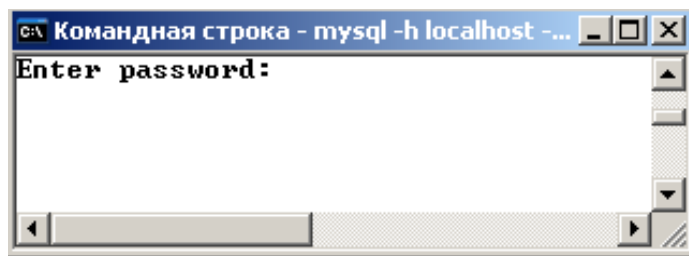


Рисунок 7 – Окно с командой ввода пароля

При получении такой подсказки нужно ввести свой пароль и нажать клавишу *Enter*. Если пароль правильный, `mysql` ответит выводом информации о справочных ключах и подсказкой `mysql>` (рисунок 8). Это свидетельствует о том, что СУБД MySQL готова и ожидает ввода новых запросов.

При отсутствии пароля у данной учетной записи `mysql` не выводит подсказки `Enter password:` (рисунок 7). Пользователь сразу видит окно, изображенное на рисунке 8.

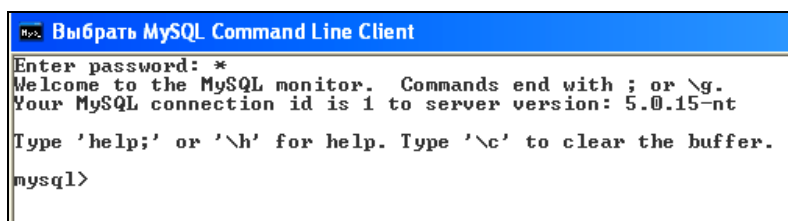


Рисунок 8 – Окно готовности `mysql`

2. Для подключения к серверу, работающему на каком-либо другом компьютере (см. рисунок 2, б), необходимо вначале с помощью администратора баз данных зарегистрироваться на сервере MySQL и получить учетную запись с паролем доступа к серверу.

Например, если пользователь получил от администратора учетную запись с именем `student` без пароля на учебном сервере университета, имеющем имя `uchserv`, то команда подключения к серверу будет иметь вид

```
mysql -huchserv -ustudent -p
```

После выполнения подключения необходимо выполнить следующее:

- Убедиться, что СУБД MySQL готова к работе (наличие приглашения `mysql>` (рисунок 8)) и ожидает ввода новых запросов.

- Ввести команду для задания кодировки хранения символов в базе данных:

```
set names cp1251;
```

Теперь СУБД готова к обработке новых запросов пользователя.

- После выполнения запросов к базе данных пользователь должен прервать установленное соединение с сервером командой `quit` или `exit`.

2.1.3. Ввод запросов (SQL-команд)

Запросы к базе данных можно вводить только *после подключения к серверу* (подраздел 2.1.2). Для ввода запроса необходимо выполнить следующее: напечатать команду запроса, добавить в конце точку с запятой и нажать клавишу *Enter*. Запрос передается серверу на выполнение, после обработки которого сервер передает результат программе-клиенту `mysql`. Клиент отображает полученный результат в окне. Рассмотрим пример простейшего запроса.

Пример 1. Запрос на получение текущей даты и времени. После приглашения `mysql>` напечатайте команду

```
SELECT NOW();
```

Результат ее выполнения представлен на рисунке 9.

```
mysql> SELECT NOW();
+-----+
| NOW() |
+-----+
| 2008-01-11 19:01:41 |
+-----+
1 row in set (0.13 sec)
mysql> _
```

Рисунок 9 – Результат выполнения команды `SELECT NOW()`

Примечание – При вызове функции в запросе пробел между именем функции и последующими скобками не допускается. Если поставить пробел перед скобками (например, `SELECT NOW ()`), то программа-клиент `mysql` выдаст ошибку `ERROR 1064: You have an error in your SQL syntax near '()' at.`

На рисунке 9 клиент `mysql` отображает результат запроса и строку, которая показывает, из скольких записей состоит результат и сколько времени затрачено на выполнение запроса.

Клиентская программа `mysql` воспринимает точку с запятой как окончание запроса. Если запрос занимает несколько строк, нужно нажать клавишу `Enter` для перехода на следующую строку.

Примечание – После ввода первой строки запроса подсказка меняется с `>` на `->`. В случае, если пользователь забыл напечатать точку с запятой, клиент `mysql` ожидает ее ввода и показывает, что ввод запроса еще не завершен.

Утилита `mysql` ведет себя точно также, когда нет завершения строки, заключенной в двойные (") или одинарные (') кавычки. При переходе на следующую строку при незакрытой двойной кавычке приглашение командной строки меняется с `mysql>` на `">`. При незакрытой одинарной кавычке приглашение меняется на `'>`.

Пример 2. Многострочный запрос на вывод даты, имени пользователя и версии программы в три строки:

```
SELECT NOW(),<Enter>
USER(),<Enter>
VERSION()<Enter>
;
```

Запрос и результат его выполнения представлены на рисунке 10.

```
mysql> SELECT NOW(),
-> USER(),
-> VERSION()
-> ;
+-----+-----+-----+
| NOW() | USER() | VERSION() |
+-----+-----+-----+
| 2008-01-11 19:14:04 | root@localhost | 5.0.18-nt-max |
+-----+-----+-----+
1 row in set (0.05 sec)
mysql>
```

Рисунок 10 – Результат выполнения многострочного запроса

Пример 3. Запросы на вывод фразы, заключенной в кавычки, без перехода на следующую строку, а также вывод фразы с использованием перехода на следующую строку при незакрытой двойной кавычке и незакрытой одинарной кавычке:

```
SELECT "Hello, World!";<Enter>
SELECT "Hello,<Enter>
"> World!";<Enter>
SELECT 'Hello,<Enter>
'> World!';<Enter>
```

Запросы и результаты их выполнения представлены на рисунке 11.

```
mysql> SELECT "Hello, world!";
+-----+
| Hello, world! |
+-----+
| Hello, world! |
+-----+
1 row in set (0.00 sec)

mysql> SELECT "Hello,
> world!";
+-----+
| Hello,
world! |
+-----+
| Hello,
world! |
+-----+
1 row in set (0.00 sec)

mysql> SELECT 'Hello,
> world!';
+-----+
| Hello,
world! |
+-----+
| Hello,
world! |
+-----+
1 row in set (0.00 sec)

mysql>
```

Рисунок 11 – Результаты выполнения ввода текста, заключенного в двойные и одинарные кавычки

В большинстве случаев регистр, в котором печатается команда, не имеет особого значения. Запросы, приведенные ниже, эквивалентны:

```
SELECT USER();
select user();
```

Однако, старайтесь избегать записи команды, используя различные регистры, например:

```
SeLeCt UsEr();
```

Можно использовать параметр \G, который отображает результат запроса в вертикальном формате. Рассмотрим пример.

Пример 4. Запрос на вывод текущей даты, имени пользователя и версии СУБД в вертикальном формате:

```
SELECT NOW(), USER(), VERSION() \G
```

Запрос и результат его выполнения представлены на рисунке 12.

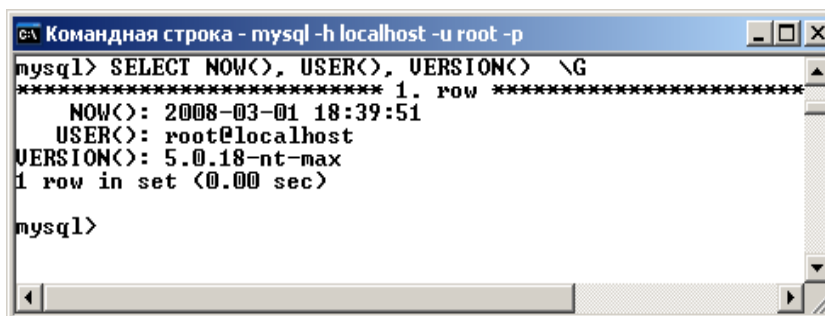


Рисунок 12 – Отображение результата запроса в вертикальном формате

Пример 5. Отмена запроса. Для отмены ввода запроса напечатайте \c:

```
SELECT NOW(), <Enter>
VERSION() \c
```

После ввода \c подсказка опять изменилась на mysql>. Это свидетельствует о том, что программа mysql ожидает ввода нового запроса.

Примечание – Примеры, приведенные в данном практикуме, выполнены в соответствии со следующим правилом: для написания всех ключевых слов и функций языка SQL используются прописные буквы, а для имен баз данных, таблиц и столбцов – строчные.

2.1.4. Правила редактирования команд в окне командной строки

В процессе ввода командных строк можно вызывать уже введенные строки, повторять их и модифицировать. Если при вводе команды была допущена ошибка, программа позволяет вернуться назад или переместиться вперед:

- на несколько символов в пределах строки с помощью клавиш управления курсором (← или →);
- на одно слово в строке с помощью комбинаций *Ctrl* ← или *Ctrl* →.

Для быстрого перехода в начало или конец командной строки используйте клавиши *Home* и *End* соответственно.

Для отображения предыдущей (предыдущих) строки используется клавиша ↑, а для отображения следующей (следующих) строки – клавиша ↓.

Очистить строку запроса можно с помощью клавиши *Esc*.

Для выделения фрагмента или целой строки и последующего копирования в другое место используется команда контекстного меню *Пометить*. Технология копирования следующая:

- вызвать команду *Пометить*;
- выделить с помощью мыши нужный фрагмент строки;
- нажать клавишу *Enter*;
- перейти в место вставки фрагмента;
- выполнить из контекстного меню команду *Вставить*.

2.1.5. Сохранение запросов и результатов их выполнения в файл текстового редактора

Работая в оконной среде, можно сохранять и воспроизводить нужные запросы или их отдельные части с помощью обычных операций копирования и вставки. Вот примерная последовательность действий:

- В окне терминала или в консоли DOS вызывается клиентская программа *mysql*.
- В отдельном окне открывается простейший текстовый редактор (например, блокнот), в котором следует набрать нужную команду SQL-запроса.
- Для выполнения в клиентской программе *mysql* запроса, набранного в текстовом редакторе, его нужно выделить и скопировать в буфер обмена.
- Затем следует перейти в окно командной строки *mysql* и вставить запрос.
- Процедура достаточно громоздкая в печатном виде, но это самый легкий путь быстрого ввода запросов. Этот метод позволяет не только редактировать запросы в окне текстового редактора, но там же создавать новые запросы путем копирования и вставки фрагментов старых запросов.

Операциями копирования и вставки можно пользоваться и *в обратном направлении*: из окна терминала – в текстовый редактор, а затем сохранить в текстовый файл.

2.1.6. Синтаксис комментариев

Сервер MySQL различает три вида комментариев.

Строка, начинающаяся с символа «#» или «*/*», рассматривается как однострочный комментарий.

Выражение, заключенное между символами «*/**» и «**/*», рассматривается как многострочный комментарий.

В MySQL 3.2 и более поздних версиях комментарии можно начинать с символов «*--* » (два тире и пробел). При этом весь текст, расположенный между этими символами и концом строки, рассматривается как комментарий.

2.1.7. Правила присвоения имен в СУБД MySQL

При присвоении имен элементам базы данных пользователь ограничен набором допустимых символов и длиной имени.

Допустимые символы. Имя может состоять из набора любых алфавитно-цифровых символов латинского языка, а также символов «*_*» (нижнее подчеркивание) и «*\$*» (знак доллара). Имена могут начинаться с любого допустимого символа, включая цифры. Однако имя столбца не может состоять из одних цифр, так как это сделает его неотличимым от чисел.

Начиная с версии 3.2, имена можно заключать в обратные одинарные кавычки, что позволяет использовать в именах любые символы, за исключением обратной одинарной кавычки.

Для имен баз данных и имен таблиц существует два дополнительных ограничения, даже при условии, что имя взято в кавычки: во-первых, не использовать точку; во-вторых, не использовать разделители, используемые при написании пути в операционной системе Windows (*/* или **).

Длина имени. Имена баз данных, таблиц, столбцов и индексов имеют длину до 64 символов. Имена псевдонимов могут иметь до 256 символов.

Чувствительность к регистру. Ключевые слова, имена функций, имена баз данных и таблиц, имена столбцов и индексов не чувствительны к регистру. Чувствительны к регистру только имена псевдонимов.

Лабораторная работа 1 РАБОТА С БД MySQL ЧЕРЕЗ КОМАНДНУЮ СТРОКУ

Порядок выполнения работы

1. Изучите последовательность команд по подключению к серверу MySQL (2 способа) и запишите их в конспект или текстовый файл.

2. Запустите программу-клиент `mysql` и установите соединение с сервером MySQL, работающим на компьютере пользователя (см. рисунок 2, а). Информацию об имени пользователя и пароле уточните у преподавателя.

3. Введите команду `mysql` для вывода текущей даты (см. пример 1, рисунок 9).

4. Задайте команды для вывода имени пользователя и версии MySQL. Запись команд выполните как многострочный запрос (см. пример 2, рисунок 10).

5. Введите команду для вывода сообщения «Привет от MySQL!». Проведите эксперименты с кавычками (см. пример 3, рисунок 11).

6. Используя информацию из подраздела 2.1.4, осуществите возврат к ранее введенной команде вывода имени пользователя и версии MySQL.

7. Откорректируйте команду так, чтобы результат отображался в вертикальном формате (см. пример 4, рисунок 12).

8. Команду и результат ее выполнения сохраните в текстовый файл блокнота под именем *первые команды mysql* в папке MySQL, созданной в вашей папке (для выполнения задания изучите подраздел 2.1.5).

9. Вернитесь к команде вывода текущей даты. Добавьте в нее комментарий *Вывод текущей даты*.

10. Команду и результат ее выполнения сохраните в уже созданный текстовый файл под именем *первые команды mysql*.

11. Завершите связь с сервером MySQL, работающим на компьютере пользователя.

12. Запустите программу-клиент `mysql` и установите соединение с сервером MySQL, работающим на учебном сервере ЛВС университета (см. рисунок 2, б). Информацию об имени пользователя и пароле уточните у преподавателя.

13. Завершите связь с сервером MySQL, работающим на учебном сервере.

В отчет по лабораторной работе необходимо представить файл *первые команды mysql* и устные ответы по материалу, изложенному в подразделах 2.1.1– 2.1.7 данного практикума.

2.2. СОЗДАНИЕ БАЗЫ ДАННЫХ

Работа с базой данных предполагает несколько этапов:

- создание (или инициализация) базы данных;
- создание таблиц в базе данных;
- взаимодействие с таблицами посредством запросов (выполнение операций вставки, выборки, модификации или удаления данных).

2.2.1. Команда создания базы данных

Создание базы данных осуществляется при помощи следующей команды:

```
CREATE DATABASE [IF NOT EXISTS] имя_базы_данных;
```

Примечание – Необязательные параметры заключаются в квадратные скобки.

Здесь `имя_базы_данных` является именем создаваемой базы данных. Необязательная ключевая фраза `IF NOT EXISTS` сообщает, что базу данных следует создавать, только если база данных с таким именем отсутствует, что позволяет предотвратить завершение запроса ошибкой.

Например, для того чтобы создать новую базу данных `forum`, нужно набрать в строке-приглашении клиента `mysql` команду, приведенную в листинге 1 и указать название базы данных.

Листинг 1. Создание базы данных

```
mysql> CREATE DATABASE forum /*создание базы данных forum*/;
```

2.2.2. Просмотр списка баз данных

Для того чтобы убедиться, что база данных `forum` успешно создана, можно выполнить команду `SHOW DATABASES` (листинг 2), которая покажет, какие базы данных существуют в вашей системе.

Листинг 2. Просмотр списка созданных баз данных

```
mysql> SHOW DATABASES;
```

Среди различных баз данных в списке можно увидеть и только что созданную базу данных `forum` (рисунок 13).

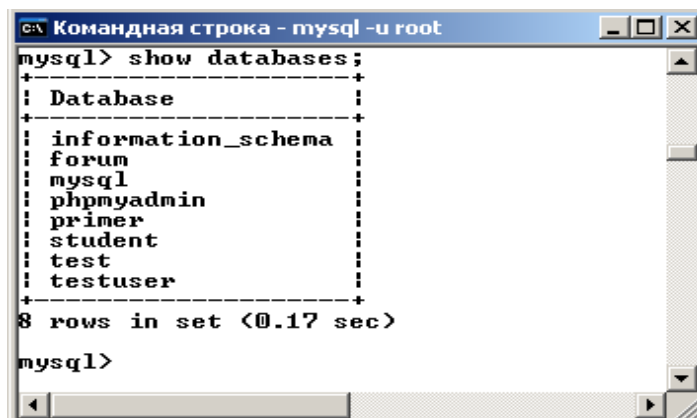


Рисунок 13 – Базы данных, созданные в MySQL

Примечание – Изначально, после инсталляции сервера MySQL, на нем хранятся только две базы данных – `test` и `mysql`. Первая используется пользователями для создания своих таблиц, она доступна всем пользователям. Во второй хранится системный каталог привилегий пользователей СУБД MySQL.

2.2.3. Команда активизации (выбора) базы данных

Для того чтобы начать работу с таблицами, необходимо сообщить серверу MySQL, в какой базе данных пользователь намерен создавать свои таблицы. Это осуществляется при помощи команды `USE`:

```
USE имя_базы_данных;
```

Здесь `имя_базы_данных` – название выбираемой базы данных.

Активируем созданную базу `forum` (листинг 3).

Листинг 3. Выбор базы данных

```
mysql> USE forum;
```

2.3. СОЗДАНИЕ ТАБЛИЦ В БАЗЕ ДАННЫХ

2.3.1. Команда создания таблиц в базе данных

Команда `CREATE TABLE` создает новую таблицу в выбранной базе данных. В общем случае команда имеет следующий синтаксис:

```
CREATE TABLE [IF NOT EXISTS] имя_таблицы (описание_столбца, ... , описание_столбца) [дополнительные_опции];
```

где `имя_таблицы` – имя создаваемой таблицы;

`описание_столбца` – объявление столбцов и индексов в таблице, если они имеются (индексы нужны для ускорения просмотра таблицы, о них мы поговорим позже);

`дополнительные_опции` – указание дополнительных свойств таблицы.

Как правило, попытка создать таблицу с уже используемым именем приводит к появлению ошибки. При определении предложения `IF NOT EXISTS` ошибка не выводится, таблица при этом не создается.

Синтаксис параметра `описание_столбца` в команде `CREATE TABLE` имеет вид

```
Имя_столбца тип_данных_столбца [параметры]
```

`Имя_столбца` всегда располагается в начале. Имя может содержать до 64 символов. Оно может со-

стоять из совокупности алфавитно-цифровых символов в кодировке, принятой по умолчанию на сервере, символа подчеркивания и символа доллара (_ и \$). Имя может начинаться с любого допустимого символа, включая цифру, но не может включать символы прямого слэша (/), обратного слэша (\) и точки (.

Тип `данных_столбца` определяет тип значений, которые может принимать столбец. Спецификация типа также может определять максимальную длину значений, хранимых в столбце. Для некоторых типов длина определяется явным образом с указанием числа. В других случаях длина определяется именем типа. В качестве типов данных столбца можно выбрать любой тип из описанных в таблицах подраздела 2.3.2.

После типа данных столбца определяются параметры: как необязательные, зависящие от типа столбца, так и более общие. Параметры действуют как модификаторы типов и вызывают изменение работы со столбцом.

Параметры столбцов

`UNSIGNED` (беззнаковый). Запрещает ввод отрицательных значений.

`SIGNED` (знаковый). Значения столбца могут принимать отрицательные значения для целого типа данных. По умолчанию установлено значение `SIGNED`.

`BINARY`. Поле, чувствительное к регистру букв (для типов `CHAR` и `VARCHAR`).

`NOT NULL`. Определяет, что поле обязательно должно иметь значение при вставке новой записи в таблицу (если не задано значение по умолчанию). Если параметр `NOT NULL` не задан, то это означает, что поле может содержать пустые значения. По умолчанию установлено значение `NULL`.

`DEFAULT значение_по_умолчанию`. Задаёт для поля значение по умолчанию, которое будет использовано, если при вставке записи для этого поля не было явно указано значение. `DEFAULT` нельзя использовать для типа `BLOB` или `TEXT`.

Примечание – Если значение по умолчанию для столбца не задано (параметр `DEFAULT` отсутствует), но столбец объявлен как `NOT NULL`, то значение по умолчанию присваивается автоматически следующим образом:

- для числовых столбцов, кроме столбцов `AUTO_INCREMENT`, значение по умолчанию устанавливается равным 0; в столбце `AUTO_INCREMENT` значением по умолчанию является следующее значение в последовательности столбца;
- для типов даты и времени, кроме `TIMESTAMP`, по умолчанию устанавливается значение «0000-00-00»;
- для строковых типов, кроме типа `ENUM`, значением по умолчанию является пустая строка; для типа `ENUM` таким значением является первый элемент перечня.

`ZEROFILL` (только для цифровых типов данных). Он указывает на то, что значения столбца при их отображении дополняются ведущими нулями до максимальной длины поля. Значения, длина которых превышает длину отображения, будут отображаться полностью без усечения.

Пример 6. Создадим столбец в таблице `mytbl` с именем `nomer`:

```
CREATE TABLE mytbl (nomer INT(5) ZEROFILL);
```

Затем введем в него значения 1, 100, 1000, 1 000 000 следующей командой:

```
INSERT INTO mytbl VALUES (1), (100), (1000), (1000000);
```

Тогда получим следующее содержимое столбца `nomer`: 00001, 00100, 01000, 1000000.

`AUTO_INCREMENT`. Этот параметр указывает, что поле является счетчиком. Обычно начальное значение равно 1 с автоматическим приращением 1.

Если при вставке новой записи указать `NULL`, то автоматически будет сгенерировано значение, на единицу большее максимального значения, уже существующего в поле.

Параметр `AUTO_INCREMENT` применяется только к столбцам, являющимся первичными ключами таблиц. Число столбцов `AUTO_INCREMENT` в таблице не может превышать одного. Кроме того, так как порядковые номера всегда целые (`INT`) и положительные (`UNSIGNED`), их можно объявлять одновременно в описании столбца.

Пример 7. Пример создания таблицы с именем `primer`. В состав таблицы входят столбец `kod`, значения которого формируются автоматически с шагом 1, начиная с начального значения 1; а также два столбца с именами `i1`, `i2` целого типа (типа `INT`) со значениями по умолчанию –1, 1 соответственно:

```
CREATE TABLE primer
(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
i1 INT DEFAULT -1,
i2 INT DEFAULT 1);
```

`PRIMARY KEY`. Указывает, что данный столбец является первичным ключом, содержащим только уни-

кальные значения. PRIMARY KEY должен быть NOT NULL, т. е. не содержать пустых значений. Каждая таблица может иметь только один PRIMARY KEY.

UNIQUE. Определяет столбец как индекс UNIQUE, содержащий только уникальные значения. В отличие от PRIMARY KEY индекс UNIQUE может содержать несколько пустых значений. Индексов UNIQUE может быть несколько в одной таблице.

INDEX и KEY являются синонимами и определяют индексы, которые могут содержать повторяющиеся значения (используются для задания внешних ключей).

FULLTEXT. Параметр, который используется при осуществлении полнотекстового поиска или так называемого поиска по контексту. Поддерживается только для таблиц типа MyISAM.

COMMENT 'string'. Определяет описательный комментарий, связанный со столбцом.

Дополнительные опции

AUTO_INCREMENT = число. Задаёт начальное значение для автоматической генерации значений столбца, заданного с типом AUTO_INCREMENT.

TYPE=тип_таблицы. Определяет структурный тип создаваемой таблицы. На практике обычно используются два типа таблиц – MyISAM и InnoDB.

DEFAULT CHARSET = кодировка. Определяет тип кодировки для создаваемой таблицы.

Рассмотрим пример команды создания таблицы.

Пример 8. Команда создания таблицы mytbl, имеющей в своем составе столбцы kod, one, two и three. Для создания таблицы можно воспользоваться командой CREATE TABLE следующего вида:

```
mysql>CREATE TABLE mytbl
->(kod INT UNSIGNED NOT NULL AUTO_INCREMENT,
->one FLOAT(5,2),
->two CHAR(15) NOT NULL DEFAULT "none",
->three TINYINT UNSIGNED
->PRIMARY KEY (kod))
->AUTO_INCREMENT=10
->TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Примечание – Столбец kod является первичным ключом, значения которого формируются автоматически с шагом 1, начиная с начального значения, равного 10. Столбец one содержит значения вещественного типа FLOAT(5,2), цифра 5 указывает общее количество цифр в числе, а цифра 2 показывает количество цифр в дробной части. Это позволяет задавать значения столбца с точностью до сотых.

Значения столбца two объявлены как строка символов длиной 15 (CHAR(15)), указана недопустимость пустых значений в поле (NOT NULL); по умолчанию в столбец записываются символы «none» (DEFAULT "none").

Тип TINYINT столбца three указывает, что столбец может принимать целые значения от 0 до 255, причем только положительные (UNSIGNED), а также может содержать пустые значения (NULL).

2.3.2. Типы данных в столбцах таблицы

Типы данных, поддерживаемые СУБД MySQL, можно разделить на четыре группы:

- числа (Numbers);
- текст (Text);
- дата и время (Date and Time);
- списки (Defined group).

Целые и вещественные типы данных

Система управления базами данных различает целые числа (например, 5 и 345) и вещественные (с дробной частью, например, 123,5). Целые числа можно представить в двоичном или шестнадцатеричном формате. Также СУБД может работать с числами в экспоненциальной форме (например, 1,34E+14 или 2,53e-3). Целые и вещественные типы перечислены в таблице 8. Целые значения могут содержать положительные или отрицательные числа, а также 0. Таким образом, любой из этих типов может быть знаковым (SIGNED) или беззнаковым (UNSIGNED).

Таблица 8 – Числовые типы столбцов

Тип	Значение	Требуемая память для хранения значения
TINYINT [(m)]	Целое число от 0 до 255 для беззнаковых и от -128 до 127 для знаковых <i>Пример.</i> TINYINT (2) – отображает значения от -9 до 99 для беззнаковых и от 0 до 99 для знаковых	1 байт

Тип	Значение	Требуемая память для хранения значения
SMALLINT [(m)]	Целое число от 0 до 65 535 для беззнаковых и от -32 768 до 32 767 для знаковых	2 байта
MEDIUMINT [(m)]	Целое число от 0 до 16 777 215 для беззнаковых и от -8 388 608 до 8 388 607 для знаковых	3 байта
INT [(m)]	Целое число от 0 до 4 294 967 295 для беззнаковых и от -2 147 683 648 до 2 147 483 647 для знаковых	4 байта
BIGINT [(m)]	Целое число от 0 до 18 446 744 073 709 551 615 для беззнаковых и от 9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 для знаковых	8 байт
FLOAT [(m, d)]	Число с плавающей точкой обычной (одинарной) точности. Значения допустимы в пределах от -3,402 823 466 E+38 до -1,175 494 351 E-38 и от 1,175 494 351 E-38 до 3,402 823 466 E+38 <i>Пример.</i> Поле типа FLOAT (7, 2) будет отображать значения как 9 999,99	4 байта
DOUBLE [(m, d)]	Число с плавающей точкой двойной точности. Значения допустимы в пределах от -1,797 693 134 862 315 E+308 до -2,225 073 858 507 201 4 E-308 и от 1,797 693 134 862 315 E+308 до 3,402 823 466 E+38	8 байт
DECIMAL [(m, d)]	Число с плавающей точкой, хранящееся в виде строки. Интервал значений тот же, что и для типа DOUBLE	m+2 при DECIMAL>0 m+1 при DECIMAL=0
Примечание – Параметр m обозначает общее количество знаков в числе, a d – количество знаков после запятой.		

Строковые типы данных

Строковые типы СУБД MySQL приведены в таблице 9. Текстовое поле может хранить любые символы, а также произвольные двоичные данные, такие как изображения и звуки.

Таблица 9 – Строковые типы столбцов

Тип	Значение	Требуемая память для хранения значения
CHAR (m)	Строка фиксированной длины (m принимает значения от 1 до 255 символов). Любой введенный текст меньшей длины будет дополнен пробелами в конце строки. Любой текст большей длины будет обрезан до заданной	m байт
VARCHAR (m)	Строка переменной длины (m принимает значения от 1 до 65 535 символов)	m+1 байт для записи длины строки
BLOB	Двоичная строка переменной длины до 65 535 символов, чувствительная к регистру	До 64 кбайт
MEDIUMBLOB		До 16 Мбайт
LONGBLOB		До 4 Гбайт
TINYTEXT	Текстовая строка сверхмалого размера до 255 символов, нечувствительная к регистру	m+1 байт (m<256)
TEXT	Строка с текстом малого размера до 65 535 сим-волов	До 64 кбайт
MEDIUMTEXT		До 16 Мбайт
LONGTEXT		До 4 Гбайт

Типы BLOB и TEXT используются для хранения изображений.

Типы CHAR и VARCHAR используются чаще всего. Единственное различие между ними заключается в том, что первый является типом с фиксированной длиной, а второй – с переменной. Все значения типа CHAR (m) занимают по m байт каждое, более короткие значения дополняются пробелами справа. Значения типа VARCHAR (m) занимают столько байт, сколько им необходимо, плюс 1 байт для хранения длины строки. При сохранении значений типа VARCHAR хвостовые пробелы отсекаются.

При выборе типов столбцов необходимо руководствоваться следующими принципами:

- при равных длинах значений типов CHAR и VARCHAR последний потребует больше памяти, так как для него требуется дополнительный байт для хранения длины значения;
- тип CHAR будет предпочтительней типа VARCHAR в том случае, если длина значений столбца изменяется незначительно (причина этого заключается в том, что строки фиксированной длины обрабатываются эффективней строк переменной длины);

- в одной таблице нельзя смешивать столбцы типа CHAR и VARCHAR;
- типы CHAR, BLOB и TEXT несовместимы и не могут присутствовать в одной таблице;
- для столбцов типа BLOB и TEXT не может быть задан атрибут DEFAULT.

Списки

В MySQL имеются два типа списков: ENUM (перечисление), SET (множество). Описание указанных типов приведено в таблице 10.

Таблица 10 – Типы списков

Тип	Значение	Требуемая память для хранения значения
Перечисление ENUM ('значение1', 'значение2', ...)	Значения в столбце могут принимать только одно из текстовых значений набора. Набор может содержать до 65 535 различных элементов. Например, столбец, определенный как ENUM ('один', 'два'), может принимать значения 'один' или 'два'	1 байт, если в наборе меньше 256 элементов, иначе – 2 байта
Множество SET ('значение1', 'значение2', ...)	Значения в столбце могут принимать несколько значений из набора. Набор может содержать до 64 различных элементов. Например, столбец, определенный как SET ('один', 'два'), может принимать значения 'один', 'два', 'один два'	От 1 до 8 байт в зависимости от количества перечисленных элементов множества
Примечание – В поле типа ENUM не рекомендуется сохранять числа, так как это может привести к излишней путанице и задавать тип ENUM желательно вместе с параметром NOT NULL. Например, объявить столбец пол типом «перечень», принимающим значения «Муж» и «Жен», можно следующим образом: пол ENUM ('МУЖ', 'ЖЕН') NOT NULL.		

Типы данных для хранения даты и времени

Типы календарных данных, имеющиеся в СУБД MySQL, показаны в таблице 11. Здесь YY, MM, DD, hh, mm и ss соответствуют годам, месяцам, дням, часам, минутам и секундам.

Таблица 11 – Столбцы календарного типа

Тип	Значение
DATE	Дата в формате YYYY-MM-DD
TIME	Время в формате hh:mm:ss
DATETIME	Дата и время в формате YYYY-MM-DD hh:mm:ss
TIMESTAMP	Значение временной отметки в формате YYYYMMDDhhmmss начиная с 19700101000000 и заканчивая неопределенной датой в 2037 г. Этот диапазон привязан к таймеру ОС UNIX, в котором отсчет начинается с первого дня 1970 г., который является «нулевой датой», или «началом эпохи» ОС UNIX. Столбец типа TIMESTAMP (временная отметка) имеет такое название, потому что записывается в момент создания или модификации записи
YEAR [(2 4)]	Значение года в формате YY или YYYY
Примечание – Даты должны задаваться в порядке год-месяц-день.	

Полный формат TIMESTAMP составляет 14 десятичных разрядов, но можно создавать поля типа TIMESTAMP и меньшей длины (таблица 12).

Таблица 12 – Зависимость формата вывода от длины поля типа TIMESTAMP

Тип поля	Формат
TIMESTAMP (14)	YYYYMMDDhhmmss
TIMESTAMP (12)	YYMMDDhhmmss
TIMESTAMP (10)	YYMMDDhhmm
TIMESTAMP (8)	YYYYMMDD
TIMESTAMP (6)	YYMMDD
TIMESTAMP (4)	YYMM
TIMESTAMP (2)	YY

Преобразовать отображение даты в более удобный для восприятия вид можно при помощи внутренней функции MySQL:

`DATE_FORMAT (имя_столбца, формат)`

Указанная функция форматирует данные столбца календарного типа, указанного в параметре `имя_столбца`, в соответствии со строкой `формат`. В строке `формат` могут использоваться следующие определители формата:

- `%Y` – год, 4 цифры;
- `%y` – год, 2 цифры;
- `%M` – название месяца (January, Desember и т. д.);
- `%m` – номер месяца, число (01–12);
- `%b` – сокращенное наименование месяца (Jan, Des и т. д.);
- `%c` – номер месяца, число (1–12);
- `%d` – день месяца, число (01–31);
- `%e` – день месяца, число (1–31).

Пример 9. Пример форматирования столбца `day`, объявленного в таблице `table` как `day DATETIME`:

```
mysql> SELECT DATE_FORMAT(day, '%d.%m.%Y %k:%i') FROM table
```

До применения команды форматирования содержимое столбца `day` отображалось в следующем виде:

```
2004-10-07 13:26:12
```

После форматирования будет получен следующий результат:

```
07.10.2004 13:26
```

Для обозначения специальных символов применяются управляющие последовательности (таблица 13). Последовательность начинается с символа обратной косой черты «\» для временного отключения от обычных правил интерпретации символов. Обратите внимание, что байт `NUL` – это не то же самое, что значение `NULL`. `NUL` – это байт с нулевым значением, а `NULL` символизирует отсутствие какого-либо значения вообще.

Таблица 13 – Управляющие последовательности

Последовательность	Значение
<code>\0</code>	NUL (ASCII 0)
<code>\'</code>	Одинарная кавычка
<code>\"</code>	Двойная кавычка
<code>\b</code>	<i>Backspace</i>
<code>\n</code>	Новая строка
<code>\r</code>	Возврат каретки
<code>\t</code>	Табуляция
<code>\\</code>	Обратная косая черта
<code>\z</code>	<i>Ctrl+Z</i> (символ EOF в ОС Windows)

Итак, достаточно традиционная проблема включения кавычек в строку может быть решена следующим образом:

- повторение символа кавычки, если это одинаковые кавычки;
- текст заключается в кавычки, отличные от кавычек, используемых в самом тексте (повторять кавычки не нужно);
- перед символом кавычки ставится обратная косая черта (тип кавычек, заключающих строку, роли не играет).

2.3.3. Выбор типа данных для поля

При создании баз данных наиболее часто используются строковые поля, которые хранят любые значения. Но использование только строковых полей создает довольно много проблем. Во-первых, при этом неэффективно используется память, во-вторых, числа и строки обрабатываются по-разному. Например, при сортировке или операциях сравнения потребуется каждый раз преобразовывать строковое поле в числовой формат. Все это окажет влияние на общую производительность системы.

На первый взгляд, все очевидно: числа хранятся в числовых полях, строки – в строковых, дата и время – в полях календарного типа. Но на самом деле не все так просто, есть множество нюансов, о которых следует помнить.

Например, если требуется хранить параметры каких-то товаров (ширину, высоту и т. д.), то можно это сделать различными способами. Допустим, можно хранить высоту, как строку «2/20», т. е. 2 м 20 см. Это значение легко для понимания, но его нельзя использовать для математических операций. Можно создать

два поля: одно будет хранить значение в метрах, другое в сантиметрах, но все же это не настолько удобно, как если бы параметр хранился в одном поле. Значит, надо создать одно поле и хранить высоту товара в миллиметрах. Возможно, это не очень понятно для пользователя, но зато удобно для хранения в БД.

Что же касается восприятия пользователем, то любые данные при выводе можно отформатировать и представить в более понятной форме. Форматированием должно заниматься приложение, с которым работает пользователь (например, веб-интерфейс, написанный на языке PHP).

Другой пример, если нужно хранить цену товара, то здесь тоже есть несколько вариантов. Типы `FLOAT` и `DOUBLE` не очень подходят, так как они округляются, а все, что касается денег, требует особой точности. Тогда можно хранить величины в полях типа `DECIMAL (length, 2)`, выбрав подходящую длину (значение параметра `length`). В этом случае значения не округляются, но операции со строками, как правило, менее эффективны, чем с числами. Есть еще один вариант: можно хранить цену как одно число – значение в копейках, но в этом случае при вводе (выводе) потребуется привести его к необходимому формату вывода.

После выбора типа данных надо решить, какой диапазон значений будет храниться, т. е. будут ли значения находиться в пределах от 0 до 100 или достигать миллионов. Можно, конечно, использовать самый большой тип (например, `BIGINT`, для числовых значений) и ни о чем не беспокоиться. Однако следует использовать минимальный из возможных типов, для того чтобы сократить объем дискового пространства, занимаемого таблицей. Это позволит повысить производительность (помните, что на жестком диске сервера хранится не только ваша база данных). Если диапазон не известен совсем, то выбирайте тип `BIGINT`. Если выбранный тип оказался слишком маленьким, и его не хватает для хранения величин, то это не так страшно – впоследствии все можно поправить с помощью оператора `ALTER TABLE`. Необходимо также помнить о том, что числовые поля по умолчанию допускают ввод отрицательных чисел и вмещают в положительном диапазоне только половину возможных значений. Чтобы решить эту проблему, используйте атрибут `UNSIGNED`.

При создании строковых полей нужно также определиться с длиной хранимого значения. Если строки не будут превышать 256 символов, то подойдет тип `CHAR`, `VARCHAR`, `TINYTEXT` или `TINYBLOB`. Для строк большей длины используйте типы `TEXT` или `BLOB`. Если строка – это набор фиксированных величин, то используйте типы `SET` или `ENUM`, которые позволяют выполнять цифровые операции и экономят память.

2.3.4. Ключи и индексы

MySQL-таблица может иметь до 32 ключей и индексов, каждый из которых может иметь до 15 полей. Максимальная поддерживаемая длина ключа – 120 байт. Обратите внимание, что длинные ключи могут привести к низкой эффективности.

Ключи могут иметь имена. В случае первичного ключа имя будет всегда `PRIMARY KEY`. Если имя ключа не задано в процессе создания таблицы, то по умолчанию им станет первое имя столбца с факультативным суффиксом (`_2`, `_3` и т. д.), чтобы сделать это имя уникальным.

Пример 10. Создание в базе данных `postavka` таблицы `product` с четырьмя столбцами: `kod` (код продукта), `name` (наименование продукта), `cena` (цена продукта), `post` (наименование поставщика). Столбец `kod` объявлен как первичный ключ (`PRIMARY KEY`), а столбцы `name` и `post` имеют индексы.

Для создания таблицы можно воспользоваться командой `CREATE TABLE`, представленной в листинге 4.

Листинг 4. Команда создания таблицы `product`

```
mysql> CREATE DATABASE postavka;
mysql> USE postavka;
mysql> CREATE TABLE product
-> (kod INT NOT NULL AUTO_INCREMENT,
-> name CHAR(20),
-> cena FLOAT(5,1),
-> post CHAR(25),
-> PRIMARY KEY (kod),
-> INDEX (name),
-> INDEX (post))
-> TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Примечание – Таблица `product` создается в базе данных `postavka`. Если база с таким именем была создана ранее, то команду создания базы данных `CREATE DATABASE postavka` писать не нужно.

Выполнив SQL-команду `SHOW TABLES`, можно убедиться, что таблица `product` успешно создана в базе данных `postavka` (рисунок 14).

```
mysql> show tables;
+-----+
| Tables_in_postavka |
+-----+
| product              |
+-----+
1 row in set (0.00 sec)
```

Рисунок 14 – Таблица **product** успешно создана

При создании ключа или индекса можно факультативно определить, что только первые N символов поля будут использоваться под ключ или для создания индекса.

Пример 11. Создание индекса для поля name из таблицы предыдущего примера, в котором только первые пять символов из двадцати являются уникальными.

Напишем команду CREATE TABLE:

```
CREATE TABLE product
(kod INT NOT NULL AUTO_INCREMENT,
 name CHAR(20),
 cena FLOAT(5,1),
 post CHAR(25),
 PRIMARY KEY (kod),
 INDEX name_idx(name(5)))
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Если при создании таблицы (команда CREATE TABLE) ключ или индекс не были заданы, то это можно сделать, воспользовавшись командой ALTER TABLE. Эта же команда используется, чтобы удалить созданный индекс или ключ. Синтаксис команды рассматривается ниже, в подразделе 2.4.1.

2.3.5. Пример создания таблицы в базе данных

Создадим таблицу authors в базе данных forum. Она содержит различные данные о зарегистрированных посетителях форума: код посетителя (id_author); имя посетителя (name); пароль (passw); email (email); веб-адрес сайта посетителя (url); номер ICQ (icq); сведения о посетителе (about); дату добавления запроса (time); последнее время посещения форума (last_time); счетчик сообщений, оставленных посетителем на форуме (themes); статус посетителя – модератор, администратор или обычный посетитель (statususer).

Сведения о типах столбцов таблицы: поле id_author принимает целые значения, задаваемые автоматически (не может быть пустым); поля name, passw, email, url, icq, about имеют тип строки переменной длины; поле pol выбирается из набора (М – мужской, Ж – женский); поля themes, statususer имеют целый числовой тип; поля time, last_time имеют тип дата и дата/время (не могут быть пустыми); поле id_author является первичным ключом таблицы; поле name индексируется по первым двум буквам имени, индексный столбец получает имя name_1.

Команда создания таблицы authors базы данных forum имеет следующий вид:

```
CREATE TABLE authors
(id_author INT NOT NULL AUTO_INCREMENT,
 name VARCHAR(15),
 pol ENUM('М', 'Ж') NOT NULL,
 passw VARCHAR(10),
 email VARCHAR(20),
 url VARCHAR(15),
 icq VARCHAR(15),
 about VARCHAR(15),
 time DATE NOT NULL,
 last_time DATETIME NOT NULL,
 themes INT DEFAULT NULL,
 statususer INT DEFAULT NULL,
 PRIMARY KEY (id_author),
 INDEX name_1(name(2)))
TYPE=MyISAM DEFAULT CHARSET=cp1251;
```

Примечание – Таблица `authors` создается в базе данных `forum`. Если базы с таким именем еще нет, то необходимо записать команды создания и активизации базы данных (`CREATE DATABASE forum` и `USE forum`), а затем команду создания таблицы `CREATE TABLE`.

Выполнив SQL-команду `SHOW TABLES`, можно убедиться, что таблица `authors` успешно создана в базе данных `forum`.

2.3.6. Просмотр структуры созданных таблиц

С помощью команды `DESCRIBE` можно показать структуру созданной таблицы. Синтаксис команды следующий:

```
DESCRIBE имя_таблицы;
```

Здесь `имя_таблицы` – имя таблицы, структура которой запрашивается.

Просмотреть структуру таблицы `product`, описанной в листинге 4, можно, выполнив SQL-запрос из листинга 5.

Листинг 5. Команда `DESCRIBE`

```
mysql> DESCRIBE product;
```

После выполнения этой команды интерпретатор MySQL выведет таблицу в соответствии с рисунком 15.

```
mysql> DESCRIBE product;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| kod   | int(11) | NO   | PRI | NULL    | auto_increment |
| name  | char(80) | YES  | MUL | NULL    |                |
| post  | char(100) | YES  | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)
```

Рисунок 15 – Результат выполнения команды `DESCRIBE`

Примечание – Более полное описание структуры таблицы `product`, включающее права доступа и комментарии, можно получить, воспользовавшись следующим оператором:

```
SHOW FULL COLUMNS FROM имя_таблицы;
```

Команда `SHOW` может принимать множество форм и предназначена для мониторинга таблиц, баз данных и сервера MySQL. При помощи команды `SHOW CREATE TABLE имя_таблицы` можно просмотреть синтаксис команды создания таблицы. При помощи команды `SHOW CHARACTER SET` можно выяснить кодировки, поддерживаемые сервером MySQL. Команда `SHOW COLUMN TYPES FROM имя_таблицы` позволяет вывести перечень информации о типах столбцов, которые использовались при создании таблиц MySQL.

При помощи команд `SHOW DATABASES` и `SHOW TABLES` можно просматривать список баз данных, размещенных на сервере, и список таблиц текущей базы данных,

Вывести список всех столбцов выбранной таблицы можно при помощи запроса `SHOW FIELDS FROM имя_таблицы` (листинг 6).

Листинг 6. Вывод структуры таблицы командой `SHOW`

```
mysql> SHOW FIELDS FROM product;
```

Результат приведен в таблице на рисунке 15.

Отображение информации обо всех индексах конкретной таблицы легко получить с помощью листинга 7.

Листинг 7. Отображение информации об индексах таблицы

```
mysql> SHOW INDEX FROM product;
```

Примечание – Получить результат, представленный в таблице на рисунке 15, можно при помощи альтернативных операторов `DESCRIBE product` и `SHOW COLUMNS FROM product`.

Информацию о таблицах текущей базы данных можно получить при помощи команды `SHOW TABLE STATUS \G`, выводящей многочисленные сведения: имя и тип таблицы; формат хранения строк; среднее число байтов, занимаемых таблицами; реальный размер файла данных таблицы, файла индекса; следующее значение для столбца с атрибутом `AUTO_INCREMENT` и т. п.

2.3.7. Команды удаления таблиц и баз данных

Команда `DROP TABLE` предназначена для удаления одной или нескольких таблиц:

```
DROP TABLE имя_таблицы [, имя_таблицы, ...];
```

К примеру, для удаления таблицы `product` нужно выполнить SQL-запрос из листинга 8.

Листинг 8. Удаление таблицы

```
mysql> DROP TABLE product;
```

Команда `DROP DATABASE` удаляет базу данных со всеми таблицами, входящими в ее состав:

```
DROP DATABASE имя_базы_данных;
```

Так, удалить базу данных `postavka` можно следующим SQL-запросом из листинга 9.

Листинг 9. Удаление базы данных

```
mysql> DROP DATABASE postavka;
```

Примечание – Прежде чем выполнять команды по удалению таблиц и баз данных, следует убедиться, что это действительно нужно сделать.

Лабораторная работа 2 СОЗДАНИЕ ТАБЛИЦ В БАЗЕ ДАННЫХ

Порядок выполнения работы

1. Запустите программу-клиент `mysql` и установите соединение с сервером MySQL, работающим на компьютере пользователя (параметры соединения с сервером уточните у преподавателя).

2. Выполните команду просмотра списка баз данных.

3. Создайте базу данных `postavka`. В ней создайте таблицу `product` (см. листинг 4).

4. Выполните команды просмотра списка баз данных, списка таблиц активизированной базы. Отобразите структуру созданной вами таблицы. Удалите созданную таблицу `product`, а затем созданную базу данных `postavka` (см. листинги 5–9).

5. Самостоятельно создайте базу данных `studentFAM` (FAM – ваша фамилия, набранная латинскими буквами).

6. Снова выполните команду просмотра списка баз данных. Убедитесь, что база данных создана.

7. Самостоятельно задайте SQL-команду создания таблицы `klient` в базе данных `studentFAM`. Для написания команды воспользуйтесь примером команды, описанной в подразделе 2.3.5. Таблица `klient` должна иметь следующие поля:

- код клиента (целое число, беззнаковое, пустые значения недопустимы, значения вводятся автоматически);

- клиент (символьное, постоянной длины в 20 символов, пустые значения недопустимы);

- статус (принимает одно из двух значений списка: «Г» – госу-дарственная организация, «К» – коммерческая структура);

- email (символьное, постоянной длины в 10 символов);

- телефон (символьное, постоянной длины в 10 символов, пустые значения недопустимы);

- город (символьное, постоянной длины в 10 символов, пустые значения недопустимы);

- код страны (целое, 5 знаков);

- страна (символьное, постоянной длины в 20 символов, пустые значения недопустимы);

Поле код клиента является первичным ключом. По полю клиент создается индекс.

Примечание – При присвоении имен столбцам таблицы желательно задавать имя столбца и через знак «нижнее подчеркивание» указывать имя таблицы, в которой создается данный столбец. Например, столбец телефон в таблице `klient` можно назвать `tel_klient`.

8. Скопируйте и сохраните в текстовый файл с именем *создание таблиц в mysql* (папка MySQL) синтаксис введенной вами команды.

9. Выполните SQL-команду просмотра структуры таблицы `klient` (синтаксис команды и результат ее выполнения скопируйте в тот же текстовый файл).

В отчет по лабораторной работе необходимо представить текстовый файл с именем *создание таблиц в mysql* в папке MySQL и результаты работы на сервере MySQL (база данных `studentFAM` и таблица `klient`).

2.4. ТЕХНОЛОГИИ РАБОТЫ С ТАБЛИЦАМИ

2.4.1. Команда изменения структуры таблиц

Команда ALTER TABLE предназначена для изменения структуры таблицы. Эта команда позволяет добавлять и удалять столбцы, создавать и уничтожать индексы, переименовывать столбцы и саму таблицу. Команда имеет следующий синтаксис:

```
ALTER TABLE имя_таблицы операция_преобразования;
```

Основные значения параметра операция_преобразования приведены в таблице 14.

Таблица 14 – Основные преобразования, выполняемые оператором ALTER TABLE

Синтаксис операции преобразования	Описание команды
ADD имя_нового_столбца тип [FIRST AFTER имя_столбца]	Добавление нового столбца с именем имя_нового_столбца и типом тип. Конструкция FIRST добавляет новый столбец перед столбцом с именем имя_столбца. Конструкция AFTER добавляет новый столбец после столбца имя_столбца. Если место добавления не указано, по умолчанию столбец добавляется в конец таблицы
ADD INDEX [имя_индекса] (имя_индексируемого_столбца, ...)	Добавление индекса с именем имя_индекса для столбца имя_индексируемого_столбца. Если имя индекса не указывается, ему присваивается имя, совпадающее с именем индексируемого столбца
ADD PRIMARY KEY (имя_столбца, ...)	Делает столбец имя_столбца или группу столбцов первичным ключом таблицы
CHANGE имя_столбца новое_имя_столбца тип	Изменение столбца с именем имя_столбца на столбец с именем новое_имя_столбца и типом тип
DROP имя_столбца	Удаление столбца с именем имя_столбца
DROP PRIMARY KEY	Удаление первичного ключа таблицы
DROP INDEX имя_индекса	Удаление индекса с именем имя_индекса или именем индексируемого столбца таблицы
старое_имя_таблицы RENAME новое_имя_таблицы	Переименование таблицы

Прежде чем рассмотреть примеры изменения структуры таблицы, создадим таблицу, с которой будем работать.

Пример 12. Создание таблицы study в базе данных studentFAM (FAM – ваша фамилия, набранная латинскими буквами), в которой даны следующие сведения о студентах:

- код студента (kod) – целое число, не может быть отрицательным, не может быть пустым, задается автоматически с шагом 1, является первичным ключом;
- фамилия студента (fam) – строка символов постоянной длины, равной 15 символам (поле индексируется по первым трем буквам фамилии);
- имя студента (im) – строка символов постоянной длины, равной 15 символам;
- пол (pol) (М – мужской, Ж – женский) – выбирается из набора, не может быть пустым;
- дата рождения (data) – тип «дата», не может быть пустой;
- средний балл аттестата (srbal) – вещественное, с одной цифрой до запятой и одной цифрой после запятой.

SQL-запрос для создания таблицы study приведен в листинге 10.

Листинг 10. Создание таблицы study в базе данных

```
mysql> CREATE DATABASE studentFAM;  
mysql> USE studentFAM;  
mysql> CREATE TABLE study  
-> (kod INT UNSIGNED NOT NULL AUTO_INCREMENT,  
-> fam CHAR(15),  
-> im CHAR(15),  
-> pol ENUM('M','Ж') NOT NULL,  
-> data DATE NOT NULL,  
-> srbal FLOAT(3,1),  
-> PRIMARY KEY (kod),  
-> INDEX fam_1(fam(3))  
-> TYPE=MyISAM;
```

Примечание – Таблица study создается в базе данных studentFAM. Если база с таким именем уже создана, то команду CREATE DATABASE писать не нужно.

Для добавления в таблицу `study` нового столбца `test` целого типа с размещением его после столбца `data` нужно выполнить SQL-запрос из листинга 11.

Листинг 11. Добавление столбца в таблицу

```
mysql> ALTER TABLE study ADD test INT(10) AFTER data;
```

Выполнив команду `DESCRIBE study`, можно увидеть, что столбец `test` успешно добавлен после столбца `data`.

Переименование столбца `test` в текстовый столбец `new_test` можно осуществить с помощью листинга 12.

Листинг 12. Переименование столбца

```
mysql> ALTER TABLE study CHANGE test new_test TEXT;
```

Удаление столбца `new_test` можно осуществить запросом из листинга 13.

Листинг 13. Удаление столбца из таблицы

```
mysql> ALTER TABLE study DROP new_test;
```

2.4.2. Команда добавления записей в таблицу

Существует несколько методов добавления информации в таблицу базы данных. Можно добавить записи вручную с помощью оператора `INSERT`. Можно добавлять записи прямо из текстового файла путем копирования в буфер. А можно готовый файл с исходными данными загрузить в таблицу с помощью оператора `LOAD DATA`. В этом разделе демонстрируется метод с использованием команды `INSERT`. Указанная команда может записываться в нескольких вариантах.

Вариант 1. Команда `INSERT` вводит значения одной записи таблицы:

```
INSERT INTO имя_таблицы VALUES (значение1, значение2, ...);
```

Примечание – Список `VALUES` должен содержать значения всех полей одной записи таблицы (обычно это порядок следования названий столбцов, заданный в операторе `CREATE TABLE`). Если вы не знаете порядка следования столбцов, воспользуйтесь оператором `DESCRIBE имя_таблицы`.

Вводить строковые значения и значения типа «дата» можно как в одинарных, так и двойных кавычках. Числовые поля задаются без использования кавычек.

Для того чтобы вставить в таблицу `study` из листинга 10 несколько записей с информацией о студентах, можно воспользоваться несколькими операторами `INSERT` (листинг 14).

Листинг 14. Оператор INSERT

```
mysql> INSERT INTO study VALUES
-> (0, 'Иванов', 'Михаил', 'М', '1990-01-05', 7.8);
mysql> INSERT INTO study VALUES
-> (0, 'Горовой', 'Максим', 'М', '1991-01-03', 5.2);
mysql> INSERT INTO study VALUES
-> (0, 'Максимова', 'Екатерина', 'Ж', '1991-01-03', 9.0);
```

Команда `INSERT INTO...VALUES` вставляет новые записи в существующую таблицу. После ключевого слова `VALUES` в скобках через запятую перечисляются все значения полей таблицы в соответствии с их типами.

Примечание – Несмотря на то, что первичный ключ (`код`) в таблице `study` объявлен как `AUTO_INCREMENT` (автоматически нумеруемый системой), при вводе записей командой `INSERT` значение первичного ключа следует задавать равным нулю.

Вариант 2. Начиная с версии MySQL 3.22.5, можно добавить сразу несколько записей с помощью многострочного оператора `INSERT`:

```
INSERT INTO имя_таблицы VALUES (значение1_столбца1, значение1_столбца2, ...),
(значение2_столбца1, значение2_столбца2, ...), (...);
```

При использовании такого оператора записи таблицы приводятся в круглых скобках через запятую после ключевого слова `VALUES`. Пример многострочного оператора `INSERT` приведен в листинге 15.

Листинг 15. Многострочный оператор INSERT

```
mysql> INSERT INTO study VALUES
-> (0, 'Иванов', 'Михаил', 'М', '1990-01-05', 7.8),
-> (0, 'Горовой', 'Максим', 'М', '1991-01-03', 5.2),
-> (0, 'Максимова', 'Екатерина', 'Ж', '1990-01-03', 9.0);
```

Вариант 3. Порядок добавления столбцов можно задавать самостоятельно, воспользовавшись следующей формой оператора INSERT:

```
INSERT INTO имя_таблицы (имя_столбцаi, имя_столбцаj) VALUES (значение_столбцаi,
значение_столбцаj);
```

Например, запись в таблицу study можно осуществить при помощи SQL-запроса, представленного в листинге 16. Столбцы, не перечисленные в списке столбцов, получают значение по умолчанию. Так, первичный ключ получает значение NULL, которое интерпретируется для полей с атрибутом AUTO_INCREMENT генерацией уникального числа.

Листинг 16. Определение порядка добавления столбцов

```
mysql> INSERT INTO study (fam, pol, srbal) VALUES ('Смирнова', 'Ж', 8.5);
```

Вариант 4. MySQL позволяет задавать значения полей в операторе INSERT в форме имя_столбца=значение:

```
INSERT INTO имя_таблицы SET имя_столбцаi=значение, имя_столбцаj=значение,
... ;
```

Пример такого оператора приведен в листинге 17.

Листинг 17. Альтернативная форма оператора INSERT

```
mysql> INSERT INTO study SET fam='Смирнова', pol='Ж';
```

Для полей, не получивших значения, будут выставлены значения по умолчанию.

2.4.3. Команда удаления записей из таблицы

Команда DELETE удаляет из таблицы записи, удовлетворяющие заданным условиям, и возвращает число удаленных записей:

```
DELETE FROM имя_таблицы [WHERE условие];
```

Важной частью запросов DELETE, UPDATE и SELECT является оператор WHERE, который позволяет задавать условия для выбора записей, на которые будут действовать эти команды. Запрос из листинга 18 удаляет из таблицы запись, значение первичного ключа в которой равно 2.

Листинг 18. Удаление записей по указанному критерию

```
mysql> DELETE FROM study WHERE kod=2;
```

Условия отбора могут быть значительно сложнее. Так, в листинге 19 удаляются все записи, в которых значение поля «пол» равно «мужской» и значение первичного ключа в которых превышает 3.

Листинг 19. Удаление записей со сложным критерием

```
mysql> DELETE FROM study WHERE pol='М' AND kod>3;
```

Оператор AND реализует логическое И. В запросах можно также применять логическое ИЛИ через оператор OR.

Примечание – СУБД MySQL в качестве альтернативы операторам AND и OR поддерживает синтаксис, принятый в Си-подобных языках программирования, т. е. вместо AND можно применять &&, а вместо OR – | (вертикальную черту).

Удаление всех записей из таблицы study представлено в листинге 20.

Листинг 20. Удаление всех записей

```
mysql> DELETE FROM study;
```

Лабораторная работа 3 ТЕХНОЛОГИИ РАБОТЫ С ТАБЛИЦАМИ В БАЗЕ ДАННЫХ

Порядок выполнения работы

1. Запустите программу-клиент `mysql` и установите соединение с сервером MySQL, работающим на компьютере пользователя.
2. Укажите имя используемой базы данных (команда `USE имя_базы_данных`).
3. Изучите команду изменения структуры таблиц. Выполните листинги 10–13.
4. Изучите команду добавления записей в таблицу. Выполните листинги 15–17.
5. Изучите команду удаления записей из таблицы. Выполните листинги 18–20.
6. Активизируйте базу данных `studentFAM`.
7. Выполните команду просмотра списка таблиц в базе данных `studentFAM`.
8. Задайте SQL-команды добавления, изменения, удаления записей из таблицы `client` в базе данных `studentFAM` в соответствии с заданиями 8.1–8.7.

Примечание – Скопируйте и сохраните в текстовый файл с именем *команды работы с таблицами в mysql* (папка MySQL) синтаксис всех введенных вами команд.

- 8.1. В описание таблицы `client` добавьте индекс по полю «город».
- 8.2. В таблицу `client` добавьте еще один столбец с именем `adres` (символьный, длиной 30) после столбца «Город».
- 8.3. Измените имя столбца `adres` на `adres1`.
- 8.4. Удалите столбец `adres1` из таблицы `client`.
- 8.5. С помощью многострочного оператора `INSERT` добавьте в таблицу `client` записи (таблица 15).

Таблица 15 – Записи таблицы `client`

Клиент	Статус	Email	Телефон	Город	Код страны	Страна
Промстрой	Г	prom@mail.ru	2463581	Москва	4567	Россия
БелАВИА	К	ba@tut.by	1475217	Минск	1258	Беларусь
Candi	К	candi@st.com	5287892	Каир	8547	Египет
SunSea	К	sun@st.com	8587892	Стамбул	1546	Турция
Вернисаж	Г	v@mn.com	2463581	Москва	4567	Россия
Спутник	Г	sp@one.by	2463581	Гомель	1258	Беларусь
Альфа	К	alfa@rt.ru	4578718	Гетеборг	2467	Швеция

- 8.6. Измените статус с «К» на «Г» для клиента «Альфа».
- 8.7. Удалите из таблицы `client` записи, в которых поле `email` пустое (используя условие `IS NULL`) или страна «Швеция».

В отчет по лабораторной работе необходимо представить текстовый файл с именем *команды работы с таблицами в mysql* в папке MySQL и результаты работы на сервере MySQL (база данных `studentFAM` и таблица `client`).

2.4.4. Команда выборки записей из таблицы по заданным критериям

Команда `SELECT` предназначена для извлечения строк данных из одной или нескольких таблиц и имеет в общем случае следующий синтаксис:

```
SELECT имя_столбца, ...  
FROM имя_таблицы WHERE условие  
[ORDER BY имя_столбца [ASC|DESC], ...]  
[LIMIT [№записи, ] количество_записей]  
[GROUP BY имя_столбца, ...];
```

Здесь `имя_столбца` – имя выбираемого столбца. Можно указать несколько столбцов через запятую, а если необходимо выбрать все столбцы, то можно просто ввести символ «звездочка» (*). Ключевое слово `FROM` указывает имя таблицы, из которой извлекаются записи. Ключевое слово `WHERE` определяет, как и в операторе `DELETE`, условия отбора строк. Ключевое слово `ORDER BY` сортирует строки результата по столбцу `имя_столбца` в прямом (`ASC`) или обратном порядке (`DESC`). Ключевое слово `LIMIT` сооб-

щает MySQL о выводе такого количества записей, которое задано в параметре количество_записей после записи, номер которой задан в параметре №записи.

Нумерация записей начинается с нуля. Ключевое слово GROUP BY задает условие группировки записей.

Для применения команды выборки записей из базы данных при помощи оператора SELECT воспользуемся ранее созданной нами таблицей study в базе данных studentFAM (см. листинг 10) и добавим в нее несколько записей (листинг 21).

Листинг 21. Вставка пяти записей в таблицу study

```
mysql> INSERT INTO study VALUES
-> (0, 'Петров', 'Михаил', 'М', '1990-15-05', 6.8),
-> (0, 'Горовая', 'Марина', 'Ж', '1991-04-02', 8.2),
-> (0, 'Семенова', 'Екатерина', 'Ж', '1991-01-03', 9.0),
-> (0, 'Карась', 'Николай', 'М', '1990-01-05', 7.8),
-> (0, 'Горкин', 'Максим', 'М', '1991-01-03', 5.2);
```

Для того чтобы посмотреть содержание всей таблицы study, выполняется запрос из листинга 22. Значения всех столбцов возвращаются в том же порядке, в котором они хранятся в таблице. Этот порядок совпадает с порядком, в котором столбцы перечислены оператором DESCRIBE.

Листинг 22. Команда SELECT

```
mysql> SELECT * FROM study;
```

В данном запросе происходит выборка всех столбцов из таблицы study без ограничений. Результат запроса показан на рисунке 16.

```
mysql> select * from study;
+----+-----+-----+----+-----+-----+
| kod | fam      | im      | pol | data      | srbal |
+----+-----+-----+----+-----+-----+
| 1   | Петров   | Михаил  | М   | 1990-05-05 | 6.8   |
| 2   | Горовая  | Марина  | Ж   | 1991-04-02 | 8.2   |
| 3   | Семенова | Екатерина | Ж   | 1991-01-03 | 9.0   |
| 4   | Карась   | Николай | М   | 1990-01-05 | 7.8   |
| 5   | Горкин   | Максим  | М   | 1991-01-03 | 5.2   |
+----+-----+-----+----+-----+-----+
5 rows in set (0.00 sec)
```

Рисунок 16 – Результат выполнения запроса из листинга 22

Выборка из определенных полей

Можно выбрать не все столбцы таблицы, а лишь часть, для этого необходимо перечислить имена выбираемых столбцов (листинг 23).

Листинг 23. Частичная выборка

```
mysql> SELECT kod, fam FROM study;
```

В этом случае MySQL выведет лишь два столбца с первичным ключом kod и фамилией студента fam (рисунок 17).

```
mysql> SELECT kod, fam FROM study;
+----+-----+
| kod | fam      |
+----+-----+
| 1   | Петров   |
| 2   | Горовая  |
| 3   | Семенова |
| 4   | Карась   |
| 5   | Горкин   |
+----+-----+
5 rows in set (0.00 sec)
```

Рисунок 17 – Результат выполнения запроса из листинга 23

Ограничение количества строк результата запроса

Ключевое слово LIMIT используется для ограничения количества строк, возвращаемых командой SELECT (листинг 24).

Листинг 24. Использование ключевого слова LIMIT в операторе SELECT

```
mysql> SELECT * FROM study LIMIT 3;
```

В результате этого запроса будут выведены только первые 3 записи из 5.

LIMIT может также принимать два целочисленных аргумента. В этом случае первый аргумент сообщает MySQL, *после какой строки* производить отсчет, а второй аргумент задает максимальное количество возвращаемых строк (листинг 25).

Листинг 25. Альтернативная форма задания предложения LIMIT

```
mysql> SELECT * FROM study LIMIT 1,3;
```

В этом случае будут возвращены строки 2, 3 и 4 (рисунок 18).

```
mysql> SELECT * FROM study LIMIT 1,3;
```

kod	fam	im	pol	data	srbal
2	Горова	Марина	Ж	1991-04-02	8.2
3	Семенова	Екатерина	Ж	1991-01-03	9.0
4	Карась	Николай	М	1990-01-05	7.8

3 rows in set (0.00 sec)

Рисунок 18 – Результат выполнения запроса из листинга 25

Определение критериев выбора записей

Ограничение набора выбираемых командой SELECT записей производится с помощью предложения WHERE, которым определяется набор выбираемых строк.

В качестве критериев можно задавать цифровые значения, значения типа DATA, а также комбинации значений и даже арифметические операторы. Оператор WHERE применяется в команде SELECT точно так же, как и в команде DELETE. Выберем из таблицы study только те записи, у которых значение kod больше 2 и меньше 5 (листинг 26).

Листинг 26. Задание условий в команде SELECT

```
mysql> SELECT * FROM study WHERE kod>2 and kod<5;
```

Результат листинга показан на рисунке 19.

```
mysql> SELECT * FROM study WHERE kod>2 and kod<5;
```

kod	fam	im	pol	data	srbal
3	Семенова	Екатерина	Ж	1991-01-03	9.0
4	Карась	Николай	М	1990-01-05	7.8

2 rows in set (0.03 sec)

Рисунок 19 – Результат выполнения запроса из листинга 26

Сортировка результатов запроса

Порядок сортировки выводимых записей можно задавать при помощи оператора ORDER BY (листинг 27).

Листинг 27. Сортировка результатов запроса

```
mysql> SELECT * FROM study WHERE kod>2 ORDER BY data;
```

В этом запросе выводятся все записи со значением поля kod больше 2, которые при этом сортируются по значению поля data (рисунок 20).

```
mysql> SELECT * FROM study WHERE kod>2 ORDER BY data;
```

kod	fam	im	pol	data	srbal
4	Карась	Николай	М	1990-01-05	7.8
3	Семенова	Екатерина	Ж	1991-01-03	9.0
5	Горкин	Максим	М	1991-01-03	5.2

3 rows in set (0.00 sec)

Рисунок 20 – Результат выполнения запроса из листинга 27

Ключевыми словами ASC и DESC можно задать порядок сортировки столбца по возрастанию и убыванию соответственно. Например, для отображения списка студентов в порядке убывания их порядковых

номеров, можно сделать запрос в соответствии с листингом 28, результат выполнения которого можно посмотреть на рисунке 21.

Листинг 28. Сортировка результатов запроса по убыванию

```
mysql> SELECT * FROM study ORDER BY kod DESC;
mysql> SELECT * FROM study ORDER BY kod DESC;
+----+-----+-----+-----+-----+-----+
| kod | fam      | im      | pol | data      | srbal |
+----+-----+-----+-----+-----+-----+
| 5   | Горкин   | Максим  | М   | 1991-01-03 | 5.2   |
| 4   | Карась   | Николай | М   | 1990-01-05 | 7.8   |
| 3   | Семенова | Екатерина | Ж   | 1991-01-03 | 9.0   |
| 2   | Горвая   | Марина  | Ж   | 1991-04-02 | 8.2   |
| 1   | Петров   | Михаил  | М   | 1990-05-05 | 6.8   |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Рисунок 21 – Результат выполнения запроса из листинга 28

Сравнение по шаблону

В СУБД MySQL реализовано сравнение по следующим шаблонам:

LIKE 'шаблон' – значение поля соответствует шаблону;

NOT LIKE 'шаблон' – значение поля не соответствует шаблону.

При описании можно использовать символы шаблона:

'_' (нижнее подчеркивание) – соответствует любому одиночному символу;

'%' (процент) – соответствует любому количеству символов.

В MySQL шаблоны по умолчанию не чувствительны к регистру символов.

Так SQL-запрос к таблице *study*, сформированный в листинге 29, приводит к выводу списка фамилий студентов мужского пола, имя которых начинается на букву «М». Отображаемый список содержит только три столбца: фамилия, имя, пол (рисунок 22).

Листинг 29. Применение оператора SELECT с шаблоном

```
SELECT fam, im, pol FROM study WHERE im LIKE 'M%' and pol='M';
```

```
mysql> SELECT fam, im, pol FROM study WHERE im LIKE 'M%' and pol='M';
+----+-----+-----+
| fam | im      | pol |
+----+-----+-----+
| Петров | Михаил | М   |
| Горкин | Марк   | М   |
+----+-----+-----+
2 rows in set (0.00 sec)
```

Рисунок 22 – Результат выполнения SQL-запроса из листинга 29

Получение итоговых результатов

Функция COUNT позволяет посчитать количество строк, выбранных запросом. Например, SQL-запрос в листинге 30 позволяет выяснить общее количество студентов.

Листинг 30. Общее количество строк в таблице study

```
mysql> SELECT COUNT(*) FROM study;
```

Результат работы листинга представлен на рисунке 23.

```
mysql> SELECT COUNT(*) FROM study;
+-----+
| COUNT(*) |
+-----+
|          5 |
+-----+
1 row in set (0.02 sec)
```

Рисунок 23 – Результат выполнения SQL-запроса из листинга 30

В качестве аргумента функции может выступать имя столбца таблицы. Запрос, приведенный ниже, аналогичен по результату запросу из листинга 30.

Листинг 31. Альтернативный оператор *SELECT*

```
mysql> SELECT COUNT(fam) FROM study;
```

Функция `COUNT (*)` подсчитывает каждую выбранную строку. Функция `COUNT (fam)` подсчитывает строки, содержащие ненулевые значения только в столбце `fam`.

Исключение дубликатов

Часто данные в таблицах дублируются. Получить выборку уникальных значений позволяет ключевое слово `DISTINCT`, которое помещается перед именем столбца. В листинге 32 приводится выборка неповторяющихся значений по столбцу `pol`.

Листинг 32. Использование ключевого слова *DISTINCT*

```
mysql> SELECT DISTINCT pol FROM study;
```

Результат листинга представлен на рисунке 24.

```
mysql> SELECT DISTINCT pol FROM study;
+-----+
| pol   |
+-----+
| М     |
| Ж     |
+-----+
2 rows in set (0.00 sec)
```

Рисунок 24 – Результат выполнения SQL-запроса из листинга 32

Ключевое слово `DISTINCT` может использоваться совместно с агрегирующей функцией `COUNT`. Все студенты, занесенные в таблицу `study` должны иметь уникальные порядковые номера (неповторяющиеся коды студента). Проверить выполнение этого требования можно при помощи SQL-запроса, приведенного в листинге 33.

Листинг 33. Проверка уникальности номеров студентов

```
mysql> SELECT COUNT(DISTINCT kod) FROM study;
```

Если полученный результат не совпадает с результатом работы листинга 30, то это значит, что предъявленное выше требование не соблюдается, т. е. в таблице имеются студенты с одинаковым кодом.

Можно подводить итоги по отдельным категориям. Например, общее число студентов мужского пола можно определить с помощью запроса из листинга 34.

Листинг 34. Общее количество студентов мужского пола

```
mysql> SELECT COUNT(*) FROM study WHERE pol='M';
```

Кроме `COUNT ()`, существуют и другие агрегирующие функции. Функции `MIN ()`, `MAX ()`, `SUM ()` и `AVG ()` предназначены для вычисления минимума, максимума, суммы и среднего значения столбца соответственно. Их можно использовать одновременно. Запрос, определяющий минимальное и максимальное значения среднего балла аттестата, представлен в листинге 35 (рисунок 25).

Листинг 35. Использование функций *MIN ()*, *MAX ()*

```
mysql> SELECT MIN(srbal) AS minimum,
-> MAX(srbal) AS maximum FROM study;
```

```
mysql> SELECT MIN(srbal) AS minimum, MAX(srbal) AS maximum FROM study;
+-----+-----+
| minimum | maximum |
+-----+-----+
|      5.2 |      9.0 |
+-----+-----+
1 row in set (0.00 sec)
```

Рисунок 25 – Результат выполнения SQL-запроса из листинга 35

Группировка записей таблицы

Наиболее продуктивно функция COUNT () используется совместно с оператором группировки GROUP BY. Так, подсчет количества студентов отдельно мужского и отдельно женского пола можно осуществить одним SQL-запросом, приведенным в листинге 36.

Листинг 36. Совместное использование COUNT () и GROUP BY

```
mysql> SELECT pol, COUNT(*) AS vsego FROM study GROUP BY pol;
```

Результат работы запроса представлен на рисунке 26, из которого видно, что в таблицу внесено 3 студента мужского пола и 2 – женского.

```
mysql> SELECT pol, COUNT(*) AS vsego FROM study GROUP BY pol;
+-----+-----+
| pol | vsego |
+-----+-----+
| М   |      3 |
| Ж   |      2 |
+-----+-----+
2 rows in set (0.00 sec)
```

Рисунок 26 – Результат выполнения SQL-запроса из листинга 36

Выборка данных из нескольких таблиц (создание многотабличных запросов)

Пусть имеются две таблицы: студенты (stud) и сессия (ses):
stud (kod_stud, fam, gruppa, kurs)
ses (data1, data2, sr_bal, kod_stud)

Пример 13. Получение информации об успеваемости студентов третьего курса с помощью многотабличного запроса:

```
SELECT stud.kod_stud, stud.fam, ses.sr_bal FROM stud, ses WHERE stud.kurs=3 AND stud.kod_stud=ses.kod_stud;
```

Настройка и использование переменных в SQL-запросах. Вложенные запросы

Пример 14. Вывод кодов студентов, у которых значение среднего балла равно минимальному значению.

Получить ответ на поставленную задачу можно двумя способами:

- использовать переменную:

```
SELECT @min_ball:=MIN(sr_ball) FROM ses;
SELECT kod_stud,sr_bal FROM ses WHERE sr_ball= @min_ball;
```

- использовать вложенный запрос:

```
SELECT kod_stud,sr_bal FROM ses WHERE sr_ball= (SELECT MIN(sr_ball) FROM ses);
```

2.4.5. Команда обновления значений столбцов

Команда UPDATE обновляет содержимое столбцов таблицы в соответствии с их новыми значениями:

```
UPDATE имя_таблицы
SET имя_столбца=новое_значение
[, имя_столбца=новое_значение ...]
[WHERE условие]
[LIMIT количество_записей];
```

В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутствует, задается условие отбора строк для обновления. В остальных случаях обновляются все строки. Ключевое слово LIMIT позволяет ограничить число обновляемых строк.

В листинге 37 для студента с фамилией Горкин меняется имя с Максима на Марк. Результат выполнения запроса представлен на рисунке 27.

Листинг 37. Применение оператора **UPDATE**

```
mysql UPDATE student SET im='Марк' WHERE fam='Горкин';
```

kod	fam	im	pol	data	srba1
1	Петров	Михаил	М	1990-05-05	6.8
2	Горова	Марина	Ж	1991-04-02	8.2
3	Семенова	Екатерина	Ж	1991-01-03	9.0
4	Карась	Николай	М	1990-01-05	7.8
5	Горкин	Марк	М	1991-01-03	5.2

5 rows in set (0.00 sec)

Рисунок 27 – Результат выполнения SQL-запроса из листинга 37

Лабораторная работа 4 ВЫБОРКА ДАННЫХ ИЗ ТАБЛИЦ

Порядок выполнения работы

1. Запустите программу-клиент `mysql` и установите соединение с сервером MySQL, работающим на компьютере пользователя.
2. Укажите имя используемой базы данных (команда `USE имя_базы_данных`).
3. Изучите команду выборки записей из таблицы по заданным критериям. Выполните листинги 21–36.
4. Изучите команду обновления значений столбцов. Выполните листинг 37.
5. Активизируйте базу данных `studentFAM`.
6. Выполните команду просмотра списка таблиц в базе данных `studentFAM`.
7. Задайте SQL-команды добавления, изменения, удаления записей из таблицы `client` в базе данных `studentFAM` в соответствии с заданиями 7.1–7.10.

Примечание – Скопируйте и сохраните в текстовый файл с именем *команды выборки записей из таблиц* (папка MySQL) синтаксис всех введенных вами команд.

- 7.1. Выберите все записи, которые имеются в таблице.
- 7.2. Извлеките и отобразите содержимое столбцов «Код клиента», «Клиент», «Email».
- 7.3. Отобразите 3 записи по столбцам «Клиент», «Город», «Страна» таблицы `client`, начиная со второй.
- 7.4. Выберите все записи по клиентам из страны, наименование которой начинается на букву «Е» или «Ш».
- 7.5. Выберите все записи по странам, наименование которых состоит из двух символов. Отобразите результат только по столбцам «Клиент», «Email» и «Страна».
- 7.6. Выберите все записи, в которых «Код клиента» больше 3 и меньше 6, отсортировав их по убыванию кода клиента. Отобразите результат только по столбцам «Код клиента» и «Клиент».
- 7.7. Рассчитайте общее количество клиентов из России.
- 7.8. Рассчитайте общее количество клиентов, являющихся государственными организациями, и количество клиентов из коммерческих структур, назвав столбцы «Статус» и «Итого».
- 7.9. Какая из стран является лидером по числу клиентов? Создайте запрос, отвечающий на поставленный вопрос.
- 7.10. Создайте запрос по расчету общего количества клиентов в каждом домене (*ru, com, by*). Результат представьте в виде двух столбцов: «Домен» и «Количество».

В отчет по лабораторной работе должны быть представлены: текстовый файл с именем *команды выборки записей из таблиц* в папке MySQL и результаты работы на сервере MySQL (база данных `studentFAM` и таблица `client`).

2.5. СОЗДАНИЕ СТРУКТУРЫ ТАБЛИЦЫ С ПОМОЩЬЮ ПРОГРАММЫ `mysqladmin`

Программа `mysqladmin` позволяет проверять состояние сервера баз данных и выполнять административные функции. Кроме того, данная программа значительно упрощает задачи по подключению к серверу баз данных MySQL и созданию таблиц базы данных.

Рассмотрим более подробно использование программы `mysqladmin` для установления соединения с сервером и создания таблиц.

Порядок действий по подключению к серверу баз данных MySQL и созданию таблицы

Подключение к серверу баз данных и создание таблицы выполняется в следующем порядке:

1. Запустите программу `mysqladmin`, выполнив команду:

Пуск/Программы/MySQL/MySQL Administrator

В результате откроется окно задания параметров подключения к серверу MySQL (рисунок 28).



Рисунок 28 – Стартовое окно программы `mysqladmin`

Как видно из рисунка, пользователю необходимо указать:

- *имя серверного узла* (Server Host), к которому нужно подключиться, и на котором установлен сервер MySQL (по умолчанию в данном поле записано имя `localhost`);
- *имя пользователя* (User_name), зарегистрированное в СУБД MySQL (по умолчанию указывается пользователь с именем `root`);
- *пароль* (Password) доступа к базе данных.

2. Указав требуемые параметры (всегда уточнять у преподавателя) и нажав кнопку *OK*, переходим к основному окну программы с перечнем режимов работы (рисунок 29).

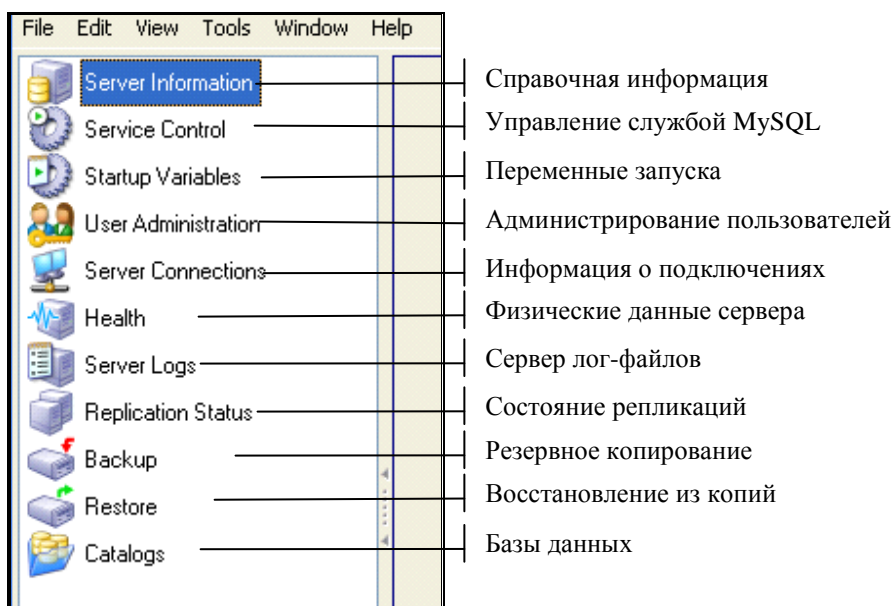


Рисунок 29 – Режимы работы программы `mysqladmin`

Так как основное назначение изучаемой программы заключается в выполнении функций администрирования сервера, следовательно, и большинство режимов предназначено для этих же целей (и доступно только пользователю, обладающему правами администратора сервера баз данных). Для реализации своей задачи (создания таблицы в базе данных) воспользуемся режимом Catalogs.

3. Щелкнув мышью по режиму Catalogs, пользователь в открывшемся окне сможет увидеть *список имен баз данных*, с которыми ему разрешено работать (в левом нижнем углу), и *4 вкладки*, определяющие направления работы с выбранной базой данных: Schema Tables(), Schema Indices(), Views(), Stored procedures().

Создайте таблицу с именем events в базе данных student, имеющую следующую структуру:

```
id_event INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(30) NOT NULL,
data DATE NOT NULL
```

4. Выделите базу данных с именем student, откройте вкладку Schema Tables и нажмите кнопку Create Table. Откроется окно, представленное на рисунке 30.

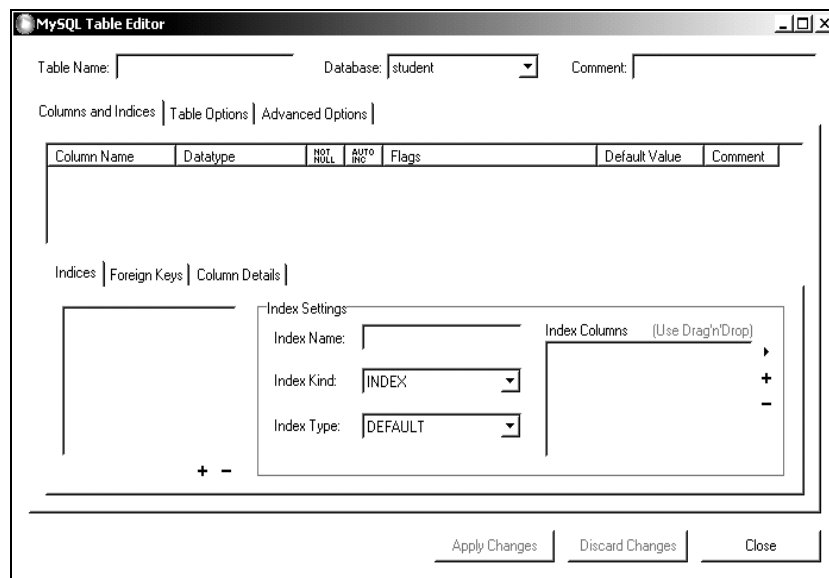


Рисунок 30 – Стартовое окно создания таблицы в программе mysqladmin

5. В поле Table Name введите имя создаваемой таблицы, например, events.

6. Далее щелкните на вкладке Column and Indices и в поле Column Name после двойного щелчка мышью (или нажатия клавиши Enter) введите имя первого столбца таблицы – id_event.

После ввода имени столбца все его свойства заполняются программой автоматически. Пользователю необходимо только откорректировать выведенные значения свойств столбца. Так свойства столбца id_event должны быть заполнены в соответствии с первой строкой рисунка 31.

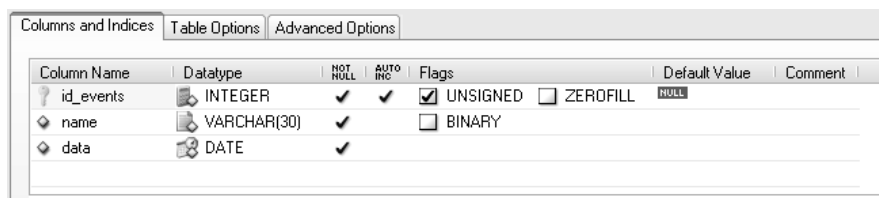



Рисунок 31 – Свойства полей таблицы events

Примечание – Для поля, являющегося первичным ключом таблицы (id_event в нашей таблице), нужно обязательно заполнить свойство поля Default Value значением NULL. Для этого щелкните кнопкой мыши на вкладке Column Details, а затем – по кнопке , расположенной левее поля Default Value (рисунок 32).

7. Создайте еще два столбца таблицы events – name и data. Окончательный вариант перечня столбцов таблицы и описания их свойств должен иметь вид, соответствующий рисунку 31.

8. Просмотрите полную информацию о каждом созданном столбце, используя вкладку Column Details (рисунок 32).

Рисунок 32 – Полная информация по столбцу `id_events`

9. Затем перейдите на вкладку `Table Options` и укажите следующее:

- тип создаваемой таблицы `MyISAM`, поставив переключатель в соответствующее значение;
 - кодировку `cp1251` в поле `Charset`.
10. Сохраните структуру созданной таблицы щелчком по кнопке `Apply Changes`.
11. В открывшемся окне подтвердите сохранение таблицы, нажав кнопку `Execute`.
12. Закройте окно создания таблицы кнопкой `Close`.

Лабораторная работа 5 СОЗДАНИЕ ТАБЛИЦ С ПОМОЩЬЮ ПРОГРАММЫ `mysqladmin`

Порядок выполнения работы

1. Создайте таблицу с именем `eventsFAM` (`FAM` – ваша фамилия, набранная латинскими буквами) в базе данных `student` (полное описание порядка ее создания содержится в подразделе 2.5.1). Параметры подключения к серверу баз данных уточните у преподавателя.

2. С помощью программы `mysqladmin` в базе данных (имя базы данных уточните у преподавателя) создайте таблицу `authorsFAM` (`FAM` – ваша фамилия, набранная по-английски).

Таблица должна содержать данные о зарегистрированных посетителях форума: код посетителя (`id_author`), имя посетителя (`name`), пол (`pol`), пароль (`passw`), `email` (`email`), веб-адрес сайта посетителя (`url`), сведения о посетителе (`about`), дату посещения форума (`time`), статус посетителя (`statususer`) – модератор, администратор или обычный посетитель.

Сведения о типах столбцов таблицы: поле `id_author` принимает целые значения, задаваемые автоматически, не может быть пустым; поля `name`, `passw`, `email`, `url`, `about` имеют тип строки переменной длины; поле `pol` выбирается из набора (М – мужской, Ж – женский); поле `time` имеет тип `date`; поле `statususer` имеет тип целого числа, содержащего одну цифру; поле `id_author` является первичным ключом таблицы.

Примечание – Если в таблице имеется столбец, являющийся первичным ключом, то при описании такого столбца обязательно выберите параметры `PRIMARY KEY`, `AUTO_INCREMENT` и заполните поле `Default Value` значением `NULL`.

3. ПРИЕМЫ РАБОТЫ С SQL-СЕРВЕРОМ СРЕДСТВАМИ PHP

3.1. ОСНОВНЫЕ PHP-ФУНКЦИИ ДЛЯ РАБОТЫ С СЕРВЕРОМ БАЗ ДАННЫХ

3.1.1. Принципы работы с базой данных через веб-интерфейс

Порядок работы с базой данных на MySQL-сервере через веб-интерфейс, организованный средствами PHP, выглядит приблизительно следующим образом (рисунок 33):

- пользователь в адресной строке веб-браузера задает имя открываемой HTML-страницы;
- веб-сервер выдает пользователю HTML-страницу с формой, в которой можно сформулировать запрос к базе данных;
- веб-сервер принимает запрос, открывает соответствующий PHP-сценарий и передает его на обработку механизму PHP;
- модуль PHP приступает к разбору сценария, сценарий содержит команду открытия соединения с базой данных и текст запроса;
- механизм PHP открывает соединение с MySQL-сервером и перенаправляет соответствующий запрос серверу MySQL;
- сервер MySQL принимает запрос к базе данных, обрабатывает его и отправляет результат (выборку из базы данных) обратно механизму PHP;
- механизм PHP завершает выполнение сценария, что обычно включает в себя форматирование результатов запроса в HTML, после этого результат в формате HTML возвращается веб-серверу;

- веб-сервер пересылает HTML-документ в браузер, и пользователь имеет возможность просмотреть результаты своего запроса к базе данных.

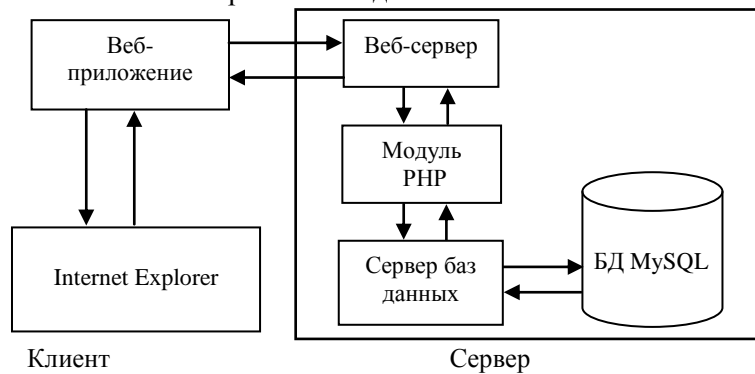


Рисунок 33 – Реализация работы с базами данных через веб-интерфейс

3.1.2. Функция установки соединения с сервером баз данных

Первым шагом при работе с SQL-сервером средствами PHP является установка соединения с ним. Это осуществляется при помощи функции `mysql_connect()`, которая имеет следующий синтаксис:

```
mysql_connect("server", "username", "password") or die ("Ошибка соединения с сервером");
```

Эта функция устанавливает соединение с сервером MySQL, сетевой адрес которого задается параметром `server`. Вторым и третьим аргументами этой функции являются имя пользователя базы данных `username` и его пароль `password` соответственно.

По умолчанию повторный вызов функции `mysql_connect()` с теми же аргументами не приводит к установлению нового соединения, вместо этого функция возвращает дескриптор уже существующего соединения.

Примечание – Все аргументы функции являются необязательными. В случае их отсутствия по умолчанию для этой функции устанавливаются следующие параметры: `server='localhost:3306'`, `username` принимает значение владельца сервера, `password` – пустая строка.

В случае успеха функция возвращает дескриптор соединения* с сервером, при неудаче – значение `false`.

Рассмотрим пример соединения с сервером MySQL, установленным на учебном сервере университета. Допустим, вы получили учетную запись с именем `student` без пароля. Тогда придется написать PHP-сценарий, представленный в листинге 38.

Листинг 38. Параметры подключения к серверу (файл `config.php`)

```
<?php
// в переменную name_server заносится имя сервера
$name_server="uchserv";
// в переменную user заносится имя пользователя
$user="student";
// в переменную pass заносится пароль
$pass="";
/* в переменную connect заносится дескриптор соединения с сервером */
$connect=mysql_connect($name_server, $user, $pass);
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

Примечание – Для подавления вывода сообщений об ошибках, генерируемых PHP в окно браузера, в листинге 39 перед функцией `mysql_connect` нужно поместить символ `@`:

```
$connect=@mysql_connect($name_server, $user, $pass);
```

* Дескриптор соединения представляет собой указатель на открытое соединение клиента с сервером, который используется для работы с несколькими подключениями.

Переменные `$name_server`, `$user`, `$pass` хранят имя сервера, имя пользователя и пароль соответственно. Их можно сохранить в отдельный файл. Например, сценарий из листинга 38 сохраним в файл с именем `config.php`. В дальнейшем при написании новых скриптов на PHP для подключения к серверу баз данных можно будет воспользоваться уже сохраненным файлом `config.php`. Включение существующего файла в новый скрипт выполняется с помощью команды `include` (листинг 39).

Листинг 39. Включение одного сценария в другой

```
<?php
include "config.php";
?>
```

3.1.3. Функция закрытия соединения с SQL-сервером

После завершения работы с базой данных необходимо разорвать соединение с SQL-сервером с помощью функции `mysql_close()`, имеющей следующий синтаксис:

```
mysql_close([дескриптор соединения]);
```

В качестве необязательного параметра функция принимает дескриптор открытого соединения. Если этот параметр не указан, закрывается последнее открытое соединение. Функция возвращает `true` в случае успеха и `false` при возникновении ошибки.

В листинге 40 приведен пример использования функции.

Листинг 40. Функция `mysql_close()`

```
<?php
// устанавливаем соединение с сервером
$name_server="uchserv";
$user="student";
$pass="";
$connect=mysql_connect($name_server, $user, $pass);
// выбираем базу данных
$dbname="student";
mysql_select_db($dbname, $connect);
// закрываем соединение с сервером
mysql_close($connect);
?>
```

3.1.4. Функция указания имени открываемой базы данных

После того как соединение установлено, необходимо выбрать базу данных, с которой пользователь желает работать. Это осуществляется при помощи функции `mysql_select_db()`, которая имеет следующий синтаксис:

```
mysql_select_db(имя_базы_данных[, дескриптор соединения]);
```

Использование этой функции эквивалентно вызову команды `USE` в SQL-запросе, т. е. функция `mysql_select_db()` выбирает базу данных для дальнейшей работы, и все последующие SQL-запросы применяются к выбранной базе данных. Функция возвращает `true` при успешном выполнении операции и `false` в противном случае.

Имеет смысл помещать функции соединения с сервером и выбора базы данных в один и тот же файл (`config1.php`), где объявлены переменные с именами сервера, пользователя и паролем (листинг 41) или включать в начало нового скрипта файл `config.php` для открытия соединения с сервером (листинг 42).

Листинг 41. Файл `config1.php`

```
<?php
// в переменную name_server заносится имя сервера
$name_server="uchserv";
// в переменную user заносится имя пользователя
$user="student";
```

```
// в переменную pass заносится пароль
$pass="";
/* в переменную connect заносится дескриптор открытого соединения с сервером
*/
$connect=mysql_connect($name_server, $user, $pass);
/* в переменную $dbname заносится имя открываемой базы */
$dbname="student";
// запрос на открытие базы student
mysql_select_db($dbname, $connect);
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

Листинг 42. Файл config2.php

```
<?php
// подключение к серверу
include "config.php";
// в переменную $dbname заносится имя открываемой базы
$dbname="student";
// запрос на открытие базы student
mysql_select_db($dbname, $connect);
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

3.1.5. Создание запросов к базе данных

После установки соединения с сервером баз данных и указания имени используемой базы выполняются SQL-запросы к базе данных из PHP-сценариев при помощи функции `mysql_query()`. Она имеет следующий синтаксис:

```
mysql_query(SQL-запрос[, дескриптор соединения])
or die("Ошибка: ".mysql_error());
```

Первый аргумент функции (SQL-запрос) представляет собой строку с SQL-запросом, второй (дескриптор соединения) представляет собой указатель соединения, возвращаемый функцией `mysql_connect()`.

Примечание – При передаче запроса функции `mysql_query()` точку с запятой в конце запроса, обязательную при работе с клиентом `mysql`, можно не ставить.

Функция возвращает дескриптор запроса в случае успеха и `false` в случае неудачного выполнения запроса.

Пример 15. Запрос на создание новой базы данных с именем `newbase`. Для создания базы данных требуется установить соединение с SQL-сервером (эта операция уже была рассмотрена выше). Затем необходимо сформировать SQL-запрос на создание базы данных `newbase` (листинг 43).

Листинг 43. Создание базы данных (файл create_base.php)

```
<?php
$connect=mysql_connect("uchserv", "student", "")
or die("Ошибка соединения с сервером");
$query="CREATE DATABASE IF NOT EXISTS newbase";
or die("Ошибка при выполнении запроса: ".mysql_error());
mysql_close($connect);
?>
```

Примечание – Пользователь, работающий с сервером баз данных, расположенном на учебном сервере университета, не обладает правами на создание баз данных. Поэтому текст листинга 44 можно рассматривать только как теоретические сведения.

Пример 16. Создание таблицы `productFAM` в базе данных `student`. Таблица `productFAM` должна содержать пять столбцов:

- kod (идентификатор продукта) – целое, задается автоматически, является первичным ключом;
- name (наименование продукта) – тип строковый, постоянной длины 20 символов;
- post (наименование поставщика) – тип строковый, постоянной длины 25 символов;
- kol (количество продукта) – целое, длина семь знаков;
- cena (цена продукта) – целое, длина десять знаков.

Для создания таблицы требуется установить соединение с SQL-сервером и выбрать базу данных, в которой она будет размещена. Эти операции уже были рассмотрены выше. Затем необходимо сформировать SQL-запрос, который описывает структуру создаваемой таблицы, и исполнить его при помощи функции `mysql_query()`.

Скрипт на создание таблицы `productFAM` в базе данных `student` приведен в листинге 44.

Листинг 44. Создание таблицы (файл `create_table.php`)

```
<HTML>
<HEAD>
<TITLE>
Создание таблицы
</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>Создание таблицы</H1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
// создание запроса на создание новой таблицы
$query="CREATE TABLE productFAM (kod INT AUTO_INCREMENT PRIMARY KEY, name CHAR(20),
post CHAR(25), kol INT(7), cena INT(10))";
$result=mysql_query($query) or die("Ошибка при выполнении запроса: ".mysql_error
());
echo "Таблица 'productFAM' успешно создана";
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>
```

Итак, таблица успешно создана, и можно переходить к наполнению ее данными.

Примечание – Создание таблиц в базе данных удобнее выполнять с использованием программы `mysqladmin` (раздел 2.4).

Пример 17. Добавление данных. Созданная таблица `productFAM` пуста. Чтобы наполнить ее данными, можно воспользоваться запросом с SQL-командой `INSERT`. После установки соединения с SQL-сервером и выбора базы данных формируется запрос на добавление строки в таблицу, который исполняется при помощи уже знакомой функции `mysql_query()`.

В листинге 45 в таблицу `productFAM` добавляются три строки, содержащие сведения про яблоки, груши и персики.

Листинг 45. Добавление данных в таблицу (`insert_table.php`)

```
<HEAD>
<TITLE>
Добавление данных в таблицу
</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>Добавление данных в таблицу</H1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* формирование SQL-запроса на добавление записей в таблицу */
$query="INSERT INTO productFAM VALUES
```

```
(0,'яблоки','Молдавия',100,3100),(0,'груши','Ита-лия',50,
4500),(0,'персики','Испания',120,5600)";
// в переменную result заносится результат запроса
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
echo "Данные успешно добавлены";
// закрытие соединения
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>
```

Примечание – Добавление данных в таблицу удобнее выполнять из HTML-форм с передачей параметров методом POST (раздел 3.2).

3.1.6. Функции для отображения результатов запросов к базе данных

В предыдущем разделе была создана таблица productFAM, содержащая три строки. Чтобы выполнить выборку всех записей из этой таблицы, выполним SQL-запрос SELECT. В результате выполнения этого запроса сформируется массив*, содержащий все строки требуемой таблицы:

```
$query="SELECT * FROM productFAM";
$result=mysql_query($query)
ordie ("Ошибка при выполнении запроса: ".mysql_error());
```

Для организации последовательного отображения всех строк таблицы пользователю используется цикл while в сочетании с функцией mysql_fetch_array(). Эта функция возвращает одну очередную строку из результирующего набора данных в виде массива, проиндексированного именами полей таблицы. Синтаксис данной функции:

```
mysql_fetch_array(результат[, вид_результата])
```

В качестве первого аргумента результат функция принимает дескриптор запроса, возвращаемый функцией mysql_query(). Второй необязательный параметр вид_результата может принимать три значения:

- mysql_assoc – возврат результата работы в виде ассоциативного массива;
- mysql_num – возврат результата работы в виде численного неассоциативного массива;
- mysql_both – возврат результата работы в виде массива, содержащего численные и ассоциативные значения.

Возвращаемый массив содержит значения, идентифицируемые как по числовым индексам столбцов (численный массив), так и именами столбцов (ассоциативный массив). Другими словами, доступ к значению каждого столбца можно получить по числовому индексу и по названию столбца. Предположим, результат запроса хранится в массиве с именем \$row, тогда доступ к его элементам можно получить следующим образом:

```
$row[1]
$row[4]
$row["name"]
$row["cena"]
```

Здесь [1],[4] – числовые индексы столбцов name и cena;
["name"],["cena"] – названия столбцов.

Тип массива (численный или ассоциативный, т. е. символьный) задается в виде аргумента вид_результата функции mysql_fetch_array().

Примечание – По умолчанию второй аргумент функции mysql_fetch_array() принимает значение mysql_both.

Пример 18. Отображение всех строк таблицы productFAM (листинг 46).

* Массив представляет собой набор данных, объединенных под одним именем.

Листинг 46. Извлечение всех строк из таблицы (view_table.php)

```
<HTML>
<HEAD>
<TITLE>
Отображение данных
</TITLE >
</HEAD>
<BODY>
<CENTER>
<H1>Отображение данных</H1>

<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* получение в переменную result результата SQL-запроса на выборку всех
записей */
$result=mysql_query("SELECT * FROM productFAM");
// проверка успешности выполнения SQL-запроса
/* если не результат (отрицательный результат) - выход */
if(!$result) exit(mysql_error());
/* открытие тега таблицы для вывода результата запроса */
echo "<TABLE>";
// цикл для вывода строк результата запроса
while($row=mysql_fetch_array($result))
{
// открыть тег строки таблицы
echo "<TR>";
/* открыть тег поля таблицы и вывести содержимое столбцов */
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[1]."</TD><TD>".$row[2].
"</td><td>".$row[3]."</td><td>".$row[4]."</td>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>
```

Пример 19. Отображение строк таблицы, удовлетворяющих определенному условию. Напишем листинг для извлечения из таблицы productFAM записей по поставкам яблок (листинг 47).

Листинг 47. Выборка записей из таблицы (select_table.php)

```
<HTML>
<HEAD>
<TITLE>
Выборка записей
</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>Выборка записей</H1>
<?php
```

```

// подключение к серверу и открытие базы данных
include "config1.php";
/* получение в переменную result результата SQL-запроса на выборку записей
по условию */
$result=mysql_query("SELECT * FROM productFAM WHERE name LIKE 'яблоки'");
// проверка успешности выполнения SQL-запроса
/* если не результат (отрицательный результат) - выход */
if(!$result) exit(mysql_error());
/* открытие тега таблицы для вывода результата запроса */
echo "<TABLE>";
// цикл для вывода строк результата запроса
while($row=mysql_fetch_array($result))
{
// открыть тег строки таблицы
echo "<TR>";
/* открыть тег поля таблицы и вывести содержимое столбцов */
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[1]."</TD><TD>".$row[2].
"</TD><TD>".$row[3]."</TD><TD>".$row[4]."</TD>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>

```

3.1.7. Изменение данных

Для изменения данных требуется всего лишь написать соответствующий SQL-запрос. Например, требуется изменить цену яблок с 3 100 на 2 900 в таблице productFAM. Для начала следует установить соединение с базой данных аналогично тому, как это сделано в предыдущих разделах. Потом выполняется SQL-запрос, изменяющий заданную строку в таблице (листинг 48).

Листинг 48. Изменение строки в таблице

```

<?php
$query="UPDATE productFAM SET cena=2900 WHERE name='яблоки'";
$result = mysql_query($query)
or die ("Ошибка при выполнении запроса: ".mysql_error());
?>

```

Сортировка данных

В листинге 46 данные выводятся в том порядке, в каком они были добавлены в таблицу. Но гораздо удобнее видеть данные, упорядоченные по какому-либо критерию. Для этого в запросе SELECT используется конструкция ORDER BY. В ней задается поле, перечень полей или выражение, которое используется для сортировки данных. Необязательный параметр DESC задает сортировку в обратном порядке. Ниже приведен фрагмент кода, который возвращает список фруктов, отсортированный по их наименованиям (листинг 49):

Листинг 49. Сортировка строк таблицы

```

<?php
$query="SELECT * FROM productFAM ORDER BY name";
$result=mysql_query($query)
or die ("Ошибка при выполнении запроса: ".mysql_error());
?>

```

Удаление данных

Для удаления данных из таблицы используется SQL-запрос DELETE. В запросе обычно присутствует условие WHERE, идентифицирующее те данные, которые следует удалить. Если по ошибке опустить условие, то будут удалены все записи из таблицы, так что в этом случае следует быть особенно внимательным. Приведем пример удаления из таблицы productFAM заданного фрукта:

```
$query="DELETE FROM productFAM WHERE name='яблоки'";
$result=mysql_query($query)
or die ("Ошибка при выполнении запроса:".mysql_error());
```

Листинг 50 демонстрирует удаление одной строки в таблице productFAM. Следует обратить внимание на то, что если условию отбора строк для удаления не удовлетворила ни одна запись из таблицы, это не является ошибкой. Такую ситуацию следует диагностировать отдельно при помощи функции mysql_affected_rows() (рассмотрена в подразделе 3.1.8 «Дополнительные функции PHP для обработки результирующих наборов»).

Листинг 50. Удаление данных (delete_data.php)

```
<HTML>
<HEAD>
<TITLE>
Удаление данных
</TITLE>
</HEAD>
<BODY>
<CENTER>

<H1>Удаление данных</H1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
// создаем запрос на удаление строк
$query="DELETE FROM productFAM WHERE name='яблоки'";
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
$query="SELECT * FROM productFAM";
$result=mysql_query($query)
or die ("Ошибка запроса: ".mysql_error());
echo "<TABLE>";
echo "<TR>";
echo "<TH>Наименование</TH><TH>Цена</TH>";
echo "</TR>";
while ($row=mysql_fetch_array($result))
{
echo "<TR>";
echo "<TD>".$row[1]."</TD><TD>".$row[4]."</TD>";
echo "</TR>";
}
echo "</TABLE>";
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>
```

3.1.8. Дополнительные функции PHP для обработки результирующих наборов

Описанные в данном разделе функции используются для организации вывода результатов выполнения запросов, осуществляемых при помощи функции mysql_query() в виде массива.

Дескриптор, возвращаемый функцией mysql_query(), используется далее для получения значений, возвращаемых СУБД. Обычно это осуществляется при помощи следующих функций: mysql_result(), mysql_fetch_array(), mysql_num_rows(), mysql_num_fields() и mysql_affected_rows().

Пусть в базе данных student имеется таблица пользователей avtor, содержащая три поля: первичный ключ (kod_avtor), имя пользователя (name) и его пароль (passw). SQL-запрос, создающий эту таблицу в базе данных student, приведен в листинге 51.

Листинг 51. Создание таблицы avtor

```
$query="CREATE TABLE avtor
(kod_avtor INT NOT NULL AUTO_INCREMENT,
name TEXT,
passw TEXT,
PRIMARY KEY (kod_avtor))
TYPE=MyISAM";
```

Функция отображения одной строки из результирующего набора

Функция `mysql_result()` возвращает значение строки с указанным номером из результирующего набора. Синтаксис функции таков:

```
mysql_result(результат_запроса, №строки[, поле])
```

В качестве первого аргумента `результат_запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Второй аргумент `№строки` представляет собой номер строки, которую необходимо вернуть. Третий необязательный параметр `поле` – это имя поля в строке результата.

Пример 20. Вывод первой записи с именем автора (листинг 52).

Листинг 52. Поиск первой записи с именем автора

```
<?php
// устанавливаем соединение
$conn=mysql_connect("uchserv","student","")
or die ("Ошибка соединения с сервером");
// выбираем базу данных
$db=mysql_select_db("student", $conn)
// выполняем SQL-запрос
$result=mysql_query("SELECT name FROM avtor");
// обрабатываем результаты запроса
if($result) echo mysql_result($result,0, 'name');
else exit(mysql_error());
mysql_close($conn);
?>
```

Примечание – Функция `mysql_result()` работает достаточно медленно, поэтому рекомендуется вместо нее использовать функцию `mysql_fetch_array()`. Для улучшения производительности функции `mysql_result()` рекомендуется задавать цифровой индекс столбца, а не полное его имя. Функция `mysql_fetch_array()` уже рассматривалась выше, в разделе 3.6.

Функция определения количества строк в результирующем наборе

Одной из распространенных задач при выборке из базы данных является определение числа записей, которые возвращает функция `mysql_query()`. Для этого предназначена функция, имеющая следующий синтаксис:

```
mysql_num_rows(результат_запроса)
```

В качестве единственного аргумента `результат_запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Рассмотрим пример использования функции `mysql_num_rows()` для авторизации пользователя. Функция `mysql_query()` в случае успеха всегда возвращает дескриптор соединения, а `false` – только в случае ошибочного синтаксиса SQL-запроса, поэтому проверка соответствия пароля в листинге 53 ошибочна, так как такая проверка будет «срабатывать» в любом случае, даже если число возвращенных записей окажется равным нулю.

Листинг 53. Ошибочная проверка пароля

```
<?php
// устанавливаем соединение
$connect=mysql_connect("uchserv","student","")
or die ("Ошибка соединения с сервером");
// выбираем базу данных
$db=mysql_select_db("student", $connect)
// формируем SQL-запрос
$query="SELECT * FROM avtor WHERE name='имя' AND pass= 'пароль'"
// выполняем SQL-запрос
$result=mysql_query($query);
// проверяем успешность выполнения SQL-запроса
if(!$result)
{
exit("Ошибка авторизации!")
}
else
{
echo("Авторизация пройдена!")
}
?>
```

В этом случае следует использовать функцию `mysql_num_rows()`, как это продемонстрировано в листинге 54.

Листинг 54. Корректная проверка пароля

```
<?php
// устанавливаем соединение
$connect=mysql_connect("uchserv","student","")
or die ("Ошибка соединения с сервером");
// выбираем базу данных
$db=mysql_select_db("student", $connect);
// формируем SQL-запрос
$query="SELECT * FROM avtor
WHERE name='имя' AND pass='пароль'";
// выполняем SQL-запрос
$result=mysql_query($query);
// проверяем успешность выполнения SQL-запроса
if(mysql_num_rows($result)==0)
{
exit("Ошибка авторизации!")
}
else
{echo("Авторизация пройдена!")}
?>
```

Число строк можно так же определить при помощи отдельного SQL-запроса, воспользовавшись встроенной функцией MySQL:

```
SELECT COUNT(*) FROM avtor;
```

Примечание – Функция `mysql_num_rows()` работает только с запросами SELECT. Чтобы получить количество строк, обработанных командами INSERT, UPDATE, DELETE, следует использовать функцию `mysql_affected_rows()`.

Функция `mysql_affected_rows()` возвращает количество строк, измененных последним запросом DELETE, INSERT, REPLACE или UPDATE в заданном соединении. Функция `mysql_affected_rows()` возвращает значение количества строк; значение 0, если ни одна строка не была изменена; значение -1, если имела место ошибка. Синтаксис функции таков:

```
mysql_affected_rows(результат_запроса)
```

В качестве единственного аргумента `результат_запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Обычно для операторов SELECT используется функция `mysql_num_rows()`.

Рассмотрим пример по подсчету записей в таблице `avtor`, измененных в последнем запросе (листинг 55).

Листинг 55. Подсчет записей, измененных последним запросом

```
<?php
// устанавливаем соединение
$connect=mysql_connect("uchserv","student","")
or die ("Ошибка соединения с сервером");
// выбираем базу данных
$db=mysql_select_db("student", $connect)
// формируем SQL-запрос
$query="INSERT INTO avtor (id_avtor, name, passw)
VALUES ('0', 'Маршак', '02-6-3')";
$result=mysql_query($query)
or die ("Запрос не выполнен \n");
// считаем количество введенных строк
$count=mysql_affected_rows($result);
print ("добавлено: ".$count);
?>
```

Функция `mysql_num_fields()` позволяет определить число столбцов в результирующем наборе. Синтаксис функции таков:

```
mysql_num_fields(результат_запроса)
```

В качестве единственного аргумента `результат_запроса` функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Рассмотрим пример использования функции `mysql_num_fields()` для подсчета столбцов результата запроса (листинг 56).

Листинг 56. Подсчет числа столбцов в результирующем наборе

```
<?php
// устанавливаем соединение
$connect=mysql_connect("uchserv","student","")
or die ("Ошибка соединения с сервером");
// выбираем базу данных
$db=mysql_select_db("student", $connect)
// формируем SQL-запрос
$query="SELECT * FROM avtor";
$result=mysql_query($query)
or die ("Запрос не выполнен \n");
// считаем количество выведенных столбцов
$count=mysql_num_fields($result);
print ("Количество столбцов равно: ".$count);
?>
```

Лабораторная работа 6 РАБОТА С SQL-СЕРВЕРОМ СРЕДСТВАМИ PHP

Порядок выполнения работы

1. С помощью программы `mysqladmin` создайте структуру таблицы `productFAM` (`FAM` – ваша фамилия, набранная латинскими буквами) в базе данных `student` на учебном сервере университета (параметры подключения к серверу баз данных уточните у преподавателя). Описание полей таблицы приведено в примере 16.

2. С помощью PHP-редактора (`PHP Expert Editor`) или любого текстового редактора напишите следующие скрипты:

- Скрипт соединения с сервером `MySQL` и открытия базы данных (см. листинг 41) (информацию по параметрам соединения уточните у преподавателя). Сохраните скрипт в файл с именем `config1.php` в свою рабочую папку на компьютере.

- Скрипт на добавление записей в созданную таблицу (см. листинг 46). Сохраните скрипт в файл с именем `insert_table.php` в свою рабочую папку.

- Скрипт на отображение всех записей, имеющих в созданной таблице (см. листинг 46). Сохраните скрипт в файл с именем `view_table.php` в свою рабочую папку.

- Скрипт на выборку записей из таблицы по определенному условию и отображению результата (см. листинг 47). Сохраните скрипт в файл с именем `select_table.php` в свою рабочую папку.

• Скрипт на удаление одной записи из таблицы с заданным условием (см. листинг 50). Сохраните скрипт в файл с именем `delete_table.php` в свою рабочую папку.

Примечание – В каждый из скриптов обязательно включите команды по подключению к серверу и по завершению соединения с сервером баз данных (можно включить в скрипт файл `config1.php` командой `include`).

3. Визуально проверьте правильность написания скриптов.

4. Скопируйте все написанные вами скрипты на учебный сервер университета в папку, указанную преподавателем.

5. Проверьте работу скриптов на учебном сервере. Для этого загрузите браузер Internet Explorer, а затем в адресную строку браузера запишите название PHP сценария и место его нахождения на сервере.

Пример: если PHP-сценарий с именем `config.php` скопирован на учебный сервер в папку `STUD_PHP`, то в адресную строку браузера нужно набрать `http://uchserv/STUD_PHP/config.php`.

В отчет по лабораторной работе необходимо представить рабочую папку пользователя с сохраненными текстами скриптов на PHP и демонстрацию работы скриптов на учебном сервере.

3.2. ИСПОЛЬЗОВАНИЕ HTML-ФОРМ ДЛЯ ПЕРЕДАЧИ ПАРАМЕТРОВ В СКРИПТЫ PHP. РЕАЛИЗАЦИЯ ПЕРЕДАЧИ ПАРАМЕТРОВ МЕТОДОМ POST

Протокол HTTP, лежащий в основе WWW, допускает передачу данных с помощью метода GET или POST. По умолчанию используется метод GET.

Передача данных методом GET не всегда является удобной по следующим причинам:

- пользователь может видеть значение параметров и легко подделывать их в строке запроса (GET-параметры передаются через HTTP-заголовки);

- объем передаваемой информации через GET-параметры ограничен (как правило, 8 кбайт).

Существует еще один способ передачи данных – передача через тело HTML-документа. Для этого предназначен метод POST. Чтобы передать данные из формы обработчику методом POST атрибуту `method` тега `<form>`, необходимо присвоить значение POST.

Рассмотрим применение метода POST на конкретном примере.

Пример 21. Реализация передачи параметров при работе с таблицей `productFAM`. Разработаем форму главного меню, которая будет содержать перечень основных режимов работы с таблицей, оформленных в виде гиперссылок (листинг 57, рисунок 34).

Листинг 57. HTML-форма главного меню работы с таблицей (`index.htm`)

```
<html>
<head>
<title>РАБОТА С ТАБЛИЦЕЙ ПРОДУКТЫ</title>
</head>
<body>
<p align="center"><b>РАБОТА С ТАБЛИЦЕЙ ПРОДУКТЫ </b></p>
<!--гиперссылка открывает страницу с именем 1.htm с формой для заполнения
полей таблицы-->
<p><a href="1.htm">ВВОД ДАННЫХ</a></p>
<!--гиперссылка открывает скрипт php, который выполняет просмотр записей
таблицы-->
<p><a href="view_table.php">ПРОСМОТР ДАННЫХ</a> </p>
<!--гиперссылка открывает страницу с именем 3.htm с формой для задания
условий поиска записей таблицы-->
<p><a href="3.htm">ВЫБОРКА ДАННЫХ</a></p>
</body>
</html>
```


В связи с тем, что данные для заполнения таблицы передаются из HTML-формы 1.htm в файл insert_table.php методом POST, внесем некоторые изменения в исходный файл insert_table.php (листинг 59).

Примечание – Файл insert_table.php описан в листинге 45.

Листинг 59. Откорректированный файл insert_table.php

```
<HTML>
<HEAD>
<TITLE>
Добавление данных в таблицу
</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>Добавление данных в таблицу</H1>
</CENTER>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* в переменные сохраняются параметры, переданные методом POST из формы
1.html */
$name=$_POST["name"];
$post=$_POST["post"];
$kol=$_POST["kol"];
$cena=$_POST["cena"];
/* формирование SQL-запроса на добавление записей в таблицу */
$query="INSERT INTO productFAM VALUES (0, '$name', '$post', '$kol', '$cena)";
/* в переменную result заносится результат запроса */
$result=mysql_query($query)
or die ("Ошибка выполнения запроса: ".mysql_error());
echo "Данные успешно добавлены";
// закрытие соединения
mysql_close($connect);
?>
</BODY>
</HTML>
```

Осталось разработать форму, в которую пользователь сможет ввести критерии выбора записей из таблицы. Назовем ее 3.htm (третий пункт главного меню). Введенные данные методом POST будут передаваться в скрипт select_table.php (листинг 60, рисунок 36).

Листинг 60. Форма для третьего пункта меню Выборка данных (3.htm)

```
<html>
<head>
<title>Выборка данных</title>
</head>
<body>
<p><b>ВЫБОРКА ДАННЫХ ИЗ ТАБЛИЦЫ</b></p>
<form method="POST" action="select_table.php">
<p>Введите значения полей для поиска записей:</p>
<p>наименование продукта  <input type="text" name="name" size="25"></p>
<p>цена<input type="text" name="cena" size="16"> </p>
<p><input type="submit" value="найти" name="OK"> </p>
</form>
<p><a href="INDEX.htm">на главную</a></p>
</body>
</html>
```

ВЫБОРКА ДАННЫХ ИЗ ТАБЛИЦЫ

Введите значения полей для поиска записей

наименование продукта

цена

[на главную](#)

Рисунок 36 – Окно формы пункта меню **Выборка данных**

В связи с тем, что данные для заполнения таблицы передаются из HTML-формы 3.htm в файл `select_table.php` методом POST, внесем некоторые изменения в исходный файл `select_table` (листинг 61).

Примечание – Файл `select_table.php` уже создан ранее на основе листинга 47.

Листинг 61. Откорректированный файл `select_table.php`

```

<HTML>
<BODY>
<CENTER>
<H1>Выборка записей</H1>
<?php
// подключение к серверу и открытие базы данных
include "config1.php";
/* в переменные сохраняются параметры, переданные методом POST из формы 3.html
*/
$name=$_POST["name"];
$cena=$_POST["cena"];
/* получение в переменную result результата SQL-запроса на выборку записей по
условию */
$query="SELECT * FROM productFAM WHERE name LIKE '$name' and cena LIKE '$cena'";
// проверка успешности выполнения SQL-запроса
/* если не результат (отрицательный результат) - выход */
if (!$result) exit (mysql_error());
/* открытие тега таблицы для вывода результата запроса */
echo "<TABLE>";
// цикл для вывода строк результата запроса
while ($row=mysql_fetch_array($result))
{
echo "<TR>";
// открыть тег строки таблицы
/* открыть тег поля таблицы и вывести содержимое столбцов */
// нумерация столбцов ведется с нуля!
echo "<TD>".$row[1]."</TD><TD>".$row[2].
"</TD><TD>".$row[3]."</TD><TD>".$row[4]."</TD>";
// закрыть тег строки таблицы
echo "</TR>";
}
// закрытие тега таблицы
echo "</TABLE>";
// закрытие соединения с сервером баз данных
mysql_close($connect);
?>
</CENTER>
</BODY>
</HTML>

```

Лабораторная работа 7 ИСПОЛЬЗОВАНИЕ HTML-ФОРМ ДЛЯ ПЕРЕДАЧИ ПАРАМЕТРОВ В СКРИПТЫ PHP

Порядок выполнения работы

1. Создайте HTML-форму главного меню работы с таблицей (`index.htm`), форму для ввода значений полей в таблицу (`1.htm`) и форму для указания условий отбора полей из таблицы (`3.htm`).

Формы можно создавать в редакторе FrontPage или в любом текстовом редакторе. При создании форм используйте листинги 57, 58, 60.

2. С помощью PHP-редактора (PHP Expert Editor) или текстового редактора Notepad (Блокнот) откорректируйте ранее написанные скрипты `insert_table.php`, `select_table.php`. Для корректировки воспользуйтесь листингами 59 и 61.

3. Самостоятельно создайте форму для удаления записей из таблицы по определенному условию. Откорректируйте ранее написанный скрипт `delete_table.php`.

4. Визуально проверьте правильность скриптов.

5. Скопируйте все написанные вами файлы на учебный сервер университета в папку, указанную преподавателем.

6. Проверьте выполнение скриптов на учебном сервере. Для этого загрузите браузер Internet Explorer и в адресной строке браузера напечатайте название главной страницы `index.htm` и место ее нахождения на сервере.

Схема взаимодействия HTML-форм с PHP-файлами их обработки представлена на рисунке 37.

Примечание – Если главная страница `index.htm` скопирована на учебный сервер в папку `STUD_PHP`, то в адресной строке браузера нужно напечатать:

`http://uchserv/STUD_PHP/index.htm.`

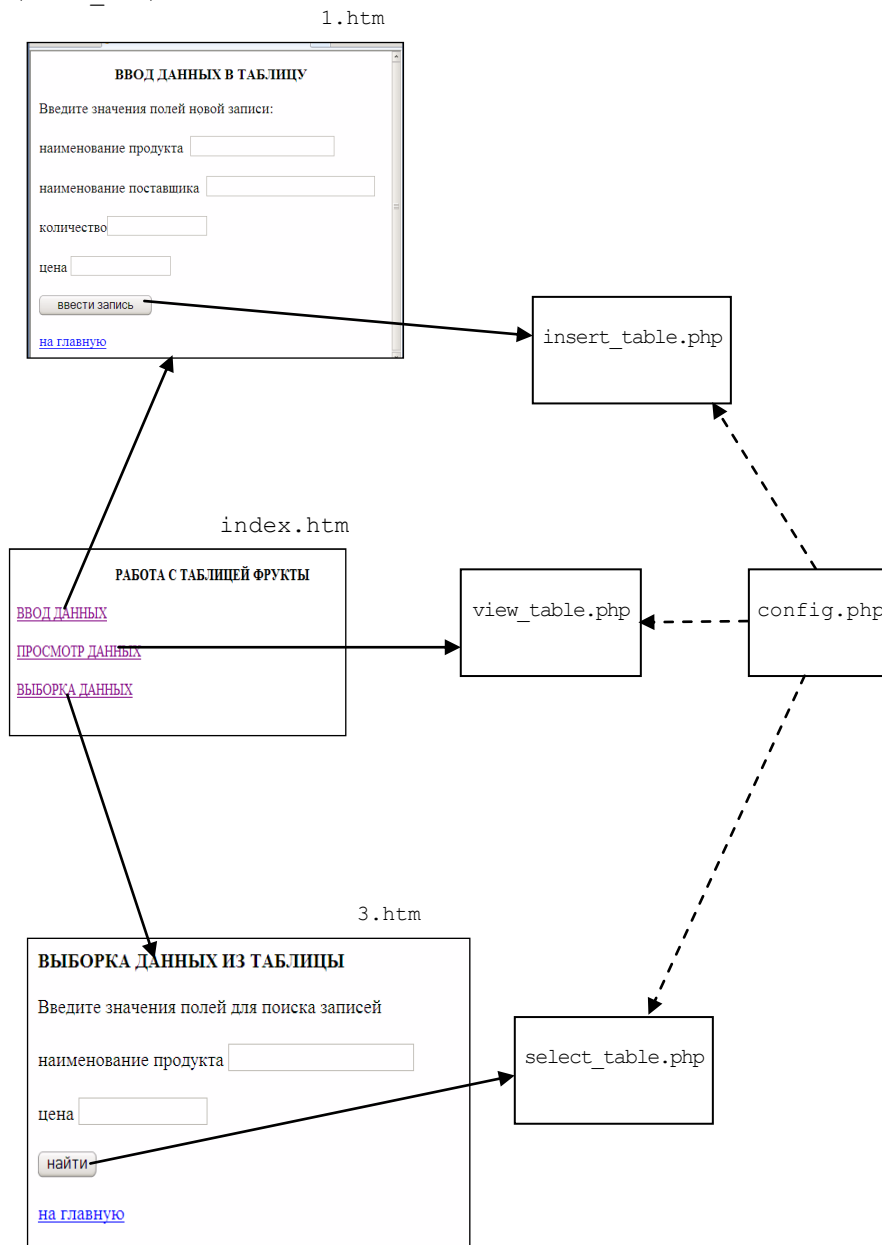


Рисунок 37 – Схема взаимодействия HTML-форм с PHP-файлами

3.3. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОЙ БАЗЫ ДАННЫХ С ВЕБ-ИНТЕРФЕЙСОМ

Рассмотрим пример создания новой базы данных. В проектировании базы данных первое, что нужно сделать – это определить ее структуру: количество таблиц, количество и наименования столбцов в таблице, реляционные связи между таблицами, а также определить основные режимы работы с базой.

Дадим описание требований к создаваемому в данном разделе информационному ресурсу.

Постановка задачи

Создать пакет программ, включающих программы по созданию базы данных *Библиотека*, вводу записей в таблицы базы, поиску записей по заданным условиям, удалению и редактированию записей. Пакет программ должен содержать общее меню по выполняемым действиям, которое открывается в виде HTML-документа с гиперссылками, позволяющими переходить на другие HTML-страницы библиотеки. Все программы должны быть написаны на языке PHP.

Решение

Описание структуры базы данных *Библиотека*

В ходе логического проектирования была определена следующая модель создаваемой базы данных, состоящая из трех таблиц:

1. Таблица Книги (*Book*), состоящая из полей:
 - идентификатор книги (*book_id*);
 - ISBN (*isbn*);
 - автор (*author*);
 - название книги (*name_book*);
 - цена (*price*).
2. Таблица Читатели (*Reader*), состоящая из полей:
 - идентификатор читателя (*reader_id*);
 - фамилия читателя (*fam_reader*);
 - имя читателя (*name_reader*).
3. Таблица Учет (*Uchet*), состоящая из полей:
 - дата (*data*);
 - идентификатор читателя (*reader_id*);
 - идентификатор книги (*name_reader*);

Структура таблиц и реляционные связи между таблицами представлены на рисунке 38.

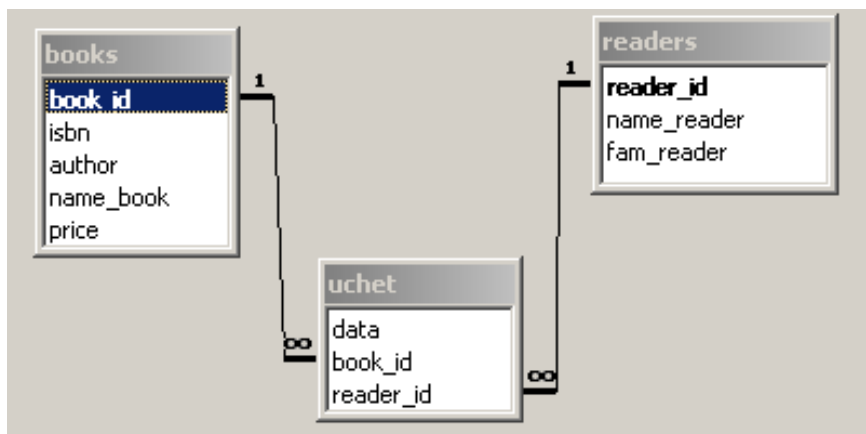


Рисунок 38 – Схема базы данных *Библиотека*

Определение основных режимов работы с базой данных *Библиотека*

Изучив предметную область данной задачи и требования, изложенные в постановке, можно определить следующие режимы проекта *Библиотека* (таблица 16).

Таблица 16 – Перечень режимов работы с базой данных Библиотека

Режимы главного меню	Реализуемые функции
Работа с книгами	<ul style="list-style-type: none"> • Отображение списка всех книг • Поиск книги по названию или (и) автору • Добавление новой книги
Работа с читателями	<ul style="list-style-type: none"> • Отображение списка читателей • Поиск читателя по фамилии • Добавление нового читателя
Учет выдачи книг	<ul style="list-style-type: none"> • Оформление выдачи книги • Оформление возврата книги • Отображение списка всех выданных книг читателям

Описание структуры таблиц базы данных

Как говорилось ранее, создание таблиц удобнее выполнять с использованием программы `mysqladmin`. Но прежде чем воспользоваться этой программой, приведем описание полей и их типов для каждой таблицы базы данных Библиотека (таблицы 17–19).

Таблица 17 – Описание таблицы **Book**

Наименование поля	Свойства поля
<code>book_id</code>	INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
<code>isbn</code>	VARCHAR(20) NOT NULL
<code>author</code>	VARCHAR(25) NOT NULL
<code>name_book</code>	VARCHAR(40) NOT NULL
<code>price</code>	INTEGER(8) UNSIGNED NOT NULL

Таблица 18 – Описание таблицы **Reader**

Наименование поля	Свойства поля
<code>reader_id</code>	INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
<code>fam_reader</code>	VARCHAR(25) NOT NULL
<code>name_reader</code>	VARCHAR(15) NOT NULL

Таблица 19 – Описание таблицы **Uchet**

Наименование поля	Свойства поля
<code>reader_id</code>	INTEGER(10) UNSIGNED NOT NULL
<code>book_id</code>	INTEGER(10) UNSIGNED NOT NULL
<code>data</code>	DATE NOT NULL

Создание таблиц в базе данных

Запустите программу `mysqladmin` (раздел 2.4), установите соединение с сервером баз данных на учебном сервере (параметры уточните у преподавателя) и создайте 3 таблицы. Структура таблиц приведена в таблицах 17–19.

Создание HTML-форм, реализующих основные режимы работы с базой данных

Формы можно создавать в редакторе `FrontPage` или в любом текстовом редакторе. В качестве имен файлов можно использовать имена с расширением `*.htm`, которые в тексте пособия приводятся в круглых скобках. Файлы сохраняются на учебный сервер в папку, указанную преподавателем.

1. HTML-форма для отображения главного меню (`index.htm`):

```
<html>
<head>
<title>Библиотека</title>
</head>
<body>
<b><center>БИБЛИОТЕКА</center></b><br><br>
<a href="book.htm">Работа с книгами</a><br>
<a href="reader.htm">Работа с читателями</a><br>
<a href="admin.htm">Учет выдачи книг</a><br>
</body>
</html>
```

Отображение HTML-формы главного меню в браузере Internet Explorer представлено на рисунке 39.

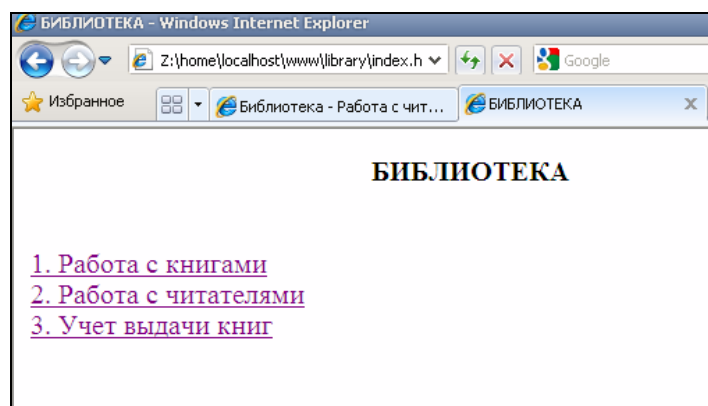


Рисунок 39 – Главное меню работы с базой данных Библиотека

2. Листинг HTML-формы для отображения содержания режима Работа с книгами (book.htm) :

```
<html>
<head>
<title>Библиотека - Работа с книгами</title>
</head>
<body>
<b>Библиотека - Работа с книгами</b><br>
<br>
<form method "POST" action="all_book.php">
Список всех книг
<input type="submit" value="Показать">
</form>
<form method="POST" action="search_book.php">
Поиск книги <br>
по названию
<input type="text" name="name_book">
или по автору
<input type="text" name="author">
<br>
<input type="submit" value="Искать">
<input type="reset" value="Отмена">
</form>
Добавить новую книгу <br>
<form method="POST" action="add_book.php">
ISBN :<input type="text" name="isbn">
Автор :<input type="text" name="author"><br>
Название книги :<input type="text" name="name_book">
Цена :<input type="text" name="price">
<input type="submit" value="Добавить">
</form>
<br>
<a href="index.htm">На главную</a><br>
</body>
</html>
```

Отображение HTML-формы режима Работа с книгами в браузере Internet Explorer представлено на рисунке 40.

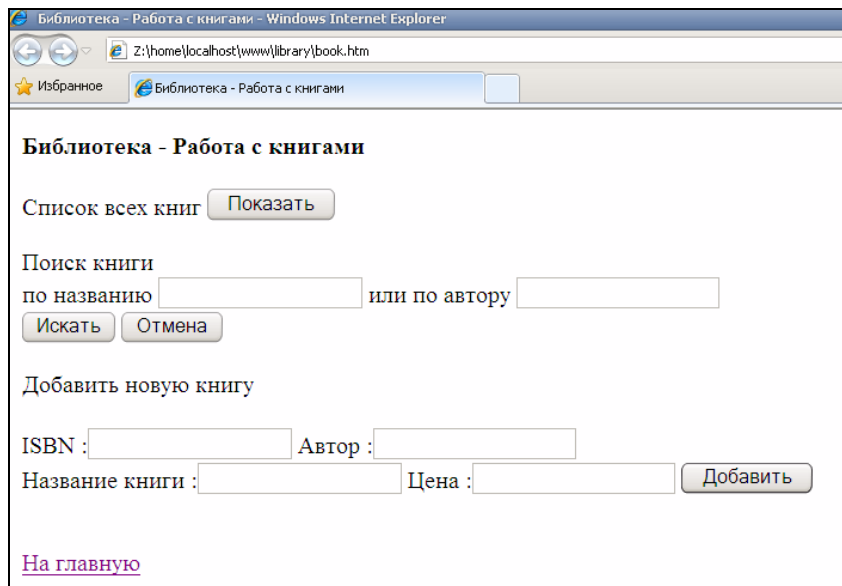


Рисунок 40 – Содержание режима Работа с книгами

3. Листинг HTML-формы для отображения содержания режима Работа с читателями (reader.htm):

```

<html>
<head>
<title>Библиотека - Работа с читателями</title>
</head>
<body>
<b>Библиотека - Работа с читателями</b><br><br>
<br>
<form method="POST" action="all_reader.php">
Список всех читателей <input type="submit" value="Показать">
<br>
</form>
<form method="POST" action="search_reader.php">
Поиск читателя по фамилии <input type="text" name="fam_reader">
<input type="submit" value="Искать">
<br>
</form>
Добавить нового читателя <br>
<form method="POST" action="add_reader.php">
Фамилия :<input type="text" name="fam_reader">
Имя :<input type="text" name="name_reader">
<input type="submit" value="Добавить">
</form>
<br>
<a href="index.htm">На главную</a>
</body>
</html>

```

Отображение HTML-формы режима Работа с читателями в браузере Internet Explorer представлено на рисунке 41.

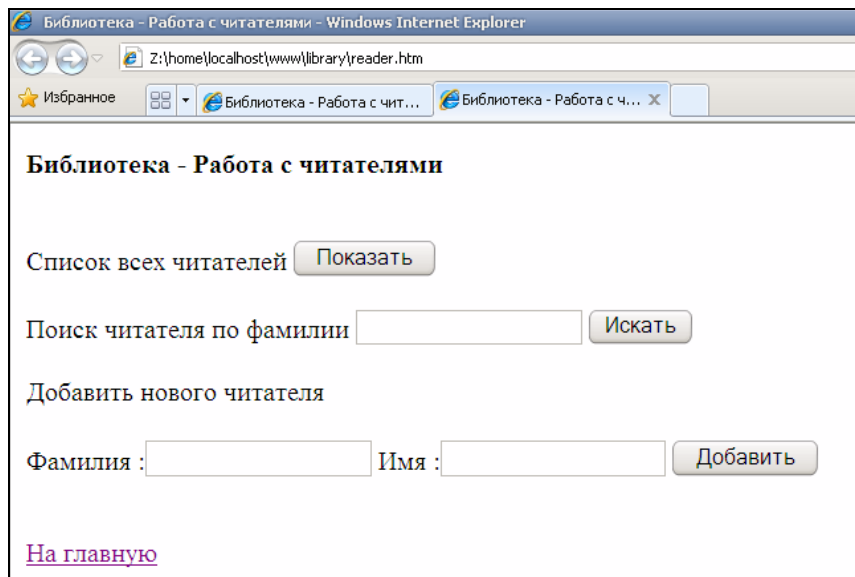


Рисунок 41 – Содержание режима Работа с читателями

4. Листинг HTML-формы для отображения содержания режима Учет выдачи книг (admin.htm) :

```

<html>
<head>
<title>Библиотека - Учет выдачи книг</title>
</head>
<body>
<b>Библиотека - Учет выдачи книг</b><br><br>
<br>
<a href="all_uchet.php">Выданные книги</a><br>
<form method="POST" action="add_uchet.php">
Читатель (код читателя)
<input type="text" name="reader_id"><br>
берет книгу (код книги)
<input type="text" name="book_id">
<input type="submit" value="Выдать книгу">
<br>
</form>
<form method="POST" action="del_uchet.php">
Читатель (код читателя)
<input type="text" name="reader_id"><br>
возвращает книгу (код книги)
<input type="text" name="book_id">
<input type="submit" value="Принять книгу">
<br>
</form>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

Отображение HTML-формы режима Работа с читателями в браузере Internet Explorer представлено на рисунке 42.

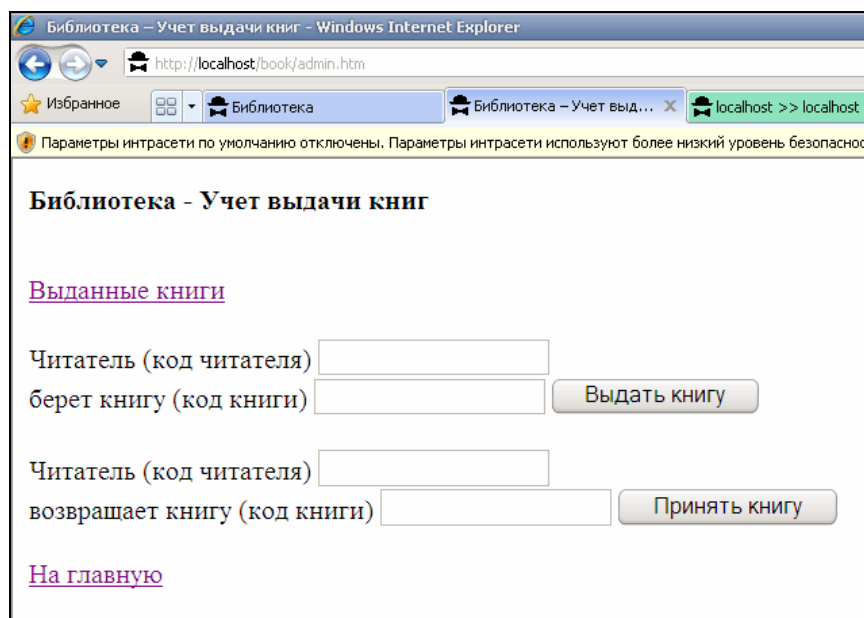


Рисунок 42 – Содержание режима **Учет выдачи книг**

Создание PHP-скриптов, реализующих основные режимы работы с базой данных

Скрипты можно писать в редакторе PHP Expert Editor или в любом текстовом редакторе. В качестве имен файлов можно использовать имена, которые приводятся в круглых скобках в тексте пособия. Файлы *.php сохраняются на учебный сервер в папку, указанную преподавателем.

5. PHP-сценарий для подключения к серверу баз данных и указания открываемой базы (config.php):

```
<?php
// соединение с сервером баз данных
$connect=mysql_connect("uchserv","student","");
// запрос на открытие базы student
mysql_select_db("student",$connect);
// задается кодировка по умолчанию
mysql_query("set names cp1251");
?>
```

6. PHP-сценарий для отображения всех книг в режиме Работа с книгами (all_book.php):

```
<html>
<head>
<title>Библиотека - Каталог</title>
</head>
<body>
<a href="index.htm">На главную</a>
<b>Библиотека - Каталог</b><br><br>
<br>
В библиотеке имеются книги:<br>
<br>
<?php
// включаем в листинг файл конфигурации
include "config.php";
// записываем в переменную запрос
$query="SELECT * FROM book";
// записываем результат запроса в переменную
```

```

$result=mysql_query($query) or die ("Ошибка: " .mysql_error());
/* организуем цикл чтения строк из массива результата */
// форматируем вывод строки в виде таблицы
echo "<table>";
/* при каждом обращении к mysql_fetch_array выдается следующая строка массива
результата */
while ($row=mysql_fetch_array($result))
/* каждая строка содержит значения всех полей таблицы */
{
echo "<tr>";
echo "<td>".$row[0]."</td><td>".$row[1].
"</td> <td>".$row[2]."</td><td>".$row[3]. "</td><td>".$row[4]."</td>";
echo "</tr>";
}
echo "</table>";
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

7. PHP-сценарий для поиска книги по названию или автору в режиме Работа с книгами (search_book.php):

```

<html>
<head>
<title>Библиотека - Поиск книг</title>
</head>
<body>
<a href="index.htm">На главную</a>
<br>
<br>
<b>Библиотека - Поиск книг</b><br><br>
<br>
Ответ на ваш запрос по книгам:<br>
<br>
<?php
// включаем в листинг файл конфигурации
include "config.php";
/* записываем в переменные значения, переданные методом POST из формы (файл
book.htm) */
$name_book= $_POST["name_book"];
$author=$_POST["author"];
// записываем в переменную SQL-запрос
$query = "SELECT * FROM book WHERE name_book LIKE '$name_book' or author LIKE
'$author'";
// вывод строки запроса для проверки синтаксиса
echo "Запрос $query <br>";
$result=mysql_query($query) or die ("Ошибка: " .mysql_error());
echo "<table>";
// выбираем каждую строку из результирующего набора
while ($row=mysql_fetch_array($result))
{
echo "<tr>";
echo "<td>".$row[0]."</td><td>".$row[1].
"</td><td>".$row[2]."</td><td>".$row[3]. "</td><td>".$row[4]."</td>";
echo "</tr>";
}
echo "</table>";

```

```
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>
```

8. PHP-сценарий для добавления новой книги в режиме Работа с книгами (add_book.php):

```
<html>
<head>
<title>Библиотека - Добавить книгу</title>
</head>
<body>
<b>Библиотека - Добавить книгу</b><br><br>
<br>
<a href="index.htm">В библиотеку</a>
<br>
<?php
// включаем в листинг файл конфигурации
include "config.php";
/* записываем в переменные значения, переданные методом POST из формы (файл
book.htm) */
$isbn=$_POST["isbn"];
$author=$_POST["author"]; $name_book=$_POST["name_book"];
$price=$_POST["price"];
echo "Регистрируется новая книга: ISBN:". $isbn. "Автор". $author. "
название". $name_book. "цена ". $price. "<br>";
// записываем в переменную SQL-запрос
$query="INSERT INTO book VALUES (0, '$isbn', '$author', '$name_book',
'$price')";
echo $query;
$result=mysql_query($query) or die ("Ошибка: ".mysql_error());
// отображение результата запроса
echo $result;
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>
```

9. PHP-сценарий для отображения списка всех читателей библиотеки в режиме Работа с читателями (all_reader.php):

```
<html>
<head>
<title>Библиотека - Все читатели</title>
</head>
<body>
<b>Библиотека - Все читатели</b><br><br>
<br>
<a href="index.htm">На главную</a><br>
В библиотеке зарегистрированы читатели:<br>
<br>
<?php
include "config.php";
$query="SELECT * FROM reader";
$result=mysql_query($query) or die ("Ошибка: ".mysql_error());
```

```

echo "<table>";
while ($row=mysql_fetch_array($result))
{
echo "<tr>";
for ($i=0; $i<mysql_num_fields($result); $i++)
{
echo "<td>$row[$i]</td>";
}
echo "</tr>";
}
echo "</table>";
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

10. PHP-сценарий для поиска читателя по фамилии в режиме Работа с читателями (search_reader.php):

```

<html>
<head>
<title>Библиотека - Поиск читателя</title>
</head>
<body>
<a href="index.htm">На главную</a>
<br>
<b>Библиотека - Поиск читателя</b><br><br>
<br>
<?php
include "config.php";
$fam_reader=$_POST["fam_reader"];
$query="SELECT * FROM reader WHERE fam_reader LIKE '$fam_reader'";
$result=mysql_query($query) or die("Ошибка: ".mysql_error());
echo "<table>";
while ($row=mysql_fetch_array($result))
{
echo "<tr>";
/* функция mysql_num_fields считает количество полей в одной строке результи-
рующего набора */
for ($i=0; $i<mysql_num_fields($result); $i++)
{
echo "<td>$row[$i]</td>";
}
echo "</tr>";
}
echo "</table>";
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

11. PHP-сценарий для добавления нового читателя в режиме Работа с читателями (add_reader.php):

```

<html>

```

```

<head>
<title>Библиотека - Добавить читателя</title>
</head>
<body>
<b>Библиотека - Добавить читателя</b><br><br>
<br>
<a href="index.htm">На главную</a>
<br>
<?php

include "config.php";
$fam_reader=$_POST["fam_reader"]; $name_reader=$_POST["name_reader"];
echo "Регистрируется новый читатель: Фамилия".$fam_reader."Имя
".$name_reader." <br>";
$query="INSERT INTO reader VALUES (0,'$fam_rea-der', '$name_reader')";
echo $query;
$result = mysql_query($query) or die("Ошибка: " .mysql_error());
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

12. PHP-сценарий для отображения списка всех выданных книг читателям в режиме Учет выдачи книг (all_uchet.php):

```

<html>
<head>
<title>Библиотека - Каталог</title>
</head>
<body>
<a href="index.htm" >На главную</a>
<b>Библиотека - Каталог</b><br><br>
<br>
В библиотеке читателям выданы книги:<br>
<br>
<?php
// включаем в листинг файл конфигурации
include "config.php";
// записываем в переменную запрос
$query="SELECT reader.reader_id, reader.fam_rea-der, reader.name_reader,
book.book_id, book.author, book.name_book, book.isbn, uchet.data FROM book,
reader, uchet WHERE reader.reader_id=uchet.reader_id AND
book.book_id=uchet.book_id";
// записываем результат запроса в переменную
$result=mysql_query($query) or die ("Ошибка: " .mysql_error());
// форматируем вывод в виде таблицы
echo "<table border=1>";
echo "<tr>";
// заголовок таблицы
echo "<td>Код читателя</td><td>Фамилия</td> <td>Имя</td><td>Код кни-
ги</td><td>Автор</td><td> Название</td><td>ISBN</td><td>Дата</td>";
echo "<tr>";
/* организуем цикл чтения строк из массива результата */
while ($row = mysql_fetch_row($result))
/* при каждом обращении к mysql_fetch_row выдается следующая строка массива ре-
зультата */

```

```

{
print "<tr>";
for ($i = 0; $i < mysql_num_fields($result); $i++)
{
print "<td>$row[$i]</td>";
}
print "</tr>";
}
echo "</table>";
// закрытие соединения с сервером
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

13. PHP-сценарий для оформления выдачи книги на руки в режиме Учет выдачи книг (add_uchet.php):

```

<html>
<head>
<title>Библиотека - Выдать книгу</title>
</head>
<body>
<b>Библиотека - Выдать книгу</b><br><br>
<br>
<a href="index.htm">На главную</a>
<br>
<?php
include "config.php";
$reader_id=$_POST["reader_id"]; $book_id=$_POST["book_id"];
$query1="SELECT name_book FROM book WHERE book_id LIKE '$book_id'";
$book=mysql_result(mysql_query($query1), 0);
$query2="SELECT fam_reader FROM reader WHERE reader_id LIKE '$reader_id'";
$reader=mysql_result(mysql_query($query2), 0);
echo "Выдается книга: код: ".$book_id." название: " . $book."читателю с
кодом ".$reader_id." фамилией " . $reader." <br>";
$query="INSERT INTO uchet VALUES ('$reader_id', '$book_id', CURDATE())";
echo $query;
$result=mysql_query($query) or die ("Ошибка: " .mysql_error());
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

14. PHP-сценарий для оформления возврата книги в режиме Учет выдачи книг (del_uchet.php):

```

<html>
<head>
<title>Библиотека - Принять книгу</title>
</head>
<body>
<b>Библиотека - Принять книгу</b><br><br>

```



```

<br>
<a href="index.htm">На главную</a>
<br>
<?php
include "config.php";
$reader_id=$_POST["reader_id"]; $book_id=$_POST["book_id"];
$query1="SELECT name_book from book where book_id LIKE '$book_id'";
$book=mysql_result(mysql_query($query1), 0);
$query2="SELECT fam_reader FROM reader WHERE reader_id LIKE '$reader_id'";
$reader=mysql_result(mysql_query($query2), 0);
echo "Принята книга: код: ".$book_id." название: ".$book." у читателя
".$reader." код читателя ".$reader_id."<br>";
$query="DELETE FROM uchet WHERE reader_id='$reader_id' and
book_id='$book_id'";
$result=mysql_query($query) or die ("Ошибка: ".mysql_error());
echo $result;
mysql_close($connect);
?>
<a href="index.htm">На главную</a><br>
</body>
</html>

```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Создание проекта по работе с базами данных в веб-интерфейсе

В соответствии с вариантом задания, указанным ниже, создайте базу данных, таблицы с информацией, HTML-форму для отображения главного меню (основные режимы работы), а также HTML-формы для добавления, удаления, поиска и отображения записей таблицы базы данных. Напишите скрипты на PHP для обработки этих форм.

Результатом самостоятельной работы являются работающие страницы на сервере.

Вариант 1

Таблица с информацией о студентах и их распределении по местам практики: фамилия, год рождения, пол, группа, факультет, наименование предприятия (организации), город.

Вариант 2

Таблица с информацией об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

Вариант 3

Таблица с информацией о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Вариант 4

Таблица с информацией о пропусках занятий студентами в течение семестра: фамилия, группа, факультет, дата, количество часов пропусков, причина пропусков (уваж., неуваж.).

Вариант 5

Таблица с информацией о сотрудниках, имеющих компьютер: название отдела, фамилия сотрудника, данные о компьютере (тип процессора, объем ОП, установленная ОС).

Вариант 6

Таблица с информацией о заказах, оформленных сотрудниками фирмы: фамилия сотрудника, оформившего сделку; наименование товара; сумма заказа; название фирмы-клиента; фамилия заказчика.

Вариант 7

Таблица с информацией об оценках, полученных студентами на экзаменах: фамилия, группа, наименование дисциплины, номер билета, оценка, преподаватель.

Вариант 8

Таблица с информацией о преподавателях кафедр: название кафедры, фамилия преподавателя, должность, ученая степень, читаемые курсы.

Вариант 9

Таблица с информацией об авторах веб-сайтов и их статьях: имя, учетная запись, пароль, тематика, заголовки статьи.

Вариант 10

Таблица с информацией о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков.

Вариант 11

Таблица с информацией о специальностях, на которых обучаются студенты: имя, фамилия, название специальности, курс, группа.

Вариант 12

Таблица стола заказов ресторана: дата проведения мероприятия, ФИО заказчика, общая сумма заказа, количество персон, вид торжества.

Вариант 13

Таблица с информацией о поставках продукции: дата, наименование продукции, единица измерения, количество, цена, номер накладной.

Вариант 14

Таблица учета валютных операций: дата, наименование валюты, курс, количество единиц, признак операции (покупка или продажа), общая стоимость.

Вариант 15

Таблица с информацией о таможенных операциях: дата, признак операции (ввоз или вывоз), группа товара, наименование товара, количество единиц, стоимость товара, сумма таможенной пошлины.

Вариант 16

Таблица с информацией о вкладах в банк: дата, ФИО, срок вклада, процентная ставка, сумма вклада.

Вариант 17

Таблица с информацией о реализации товаров: наименование товара, дата продажи, количество проданного товара, розничная цена, оптовая цена.

Вариант 18

Таблица учета командировок сотрудников: ФИО сотрудника, должность, дата начала командировки, количество дней, расходы на транспорт, прочие расходы, аванс.

Вариант 19

Таблица учета дежурств сотрудников: ФИО сотрудника, должность, подразделение (отдел), дата дежурства, время дежурства.

Вариант 20

Ведомость расчета заработной платы сотрудников: отдел, ФИО сотрудника, должность, оклад, надбавки, премия.

Вариант 21

Таблица учета товарно-материальных ценностей (ТМЦ): наименование ТМЦ, балансовая стоимость, количество (на балансе), фактическое количество, наименование отдела.

Вариант 22

Таблица учета оплаты коммунальных услуг: дата оплаты, ФИО квартиросъемщика, наименование услуги, сумма оплаты, пеня.

Вариант 23

Таблица по учету выдачи кредитов юридическим лицам: дата оформления, номер договора, наименование юридического лица, адрес, сумма кредита, срок, годовая процентная ставка.

Вариант 24

Таблица штатного расписания организации: наименование отдела или подразделения, наименование должности, количество штатных единиц, оклад.

Вариант 25

Таблица с информацией учебных планов вуза: наименование изучаемой дисциплины, наименование кафедры, номер семестра, форма контроля знаний (экзамен или зачет), количество часов.

Вариант 26

Таблица с информацией туристических фирм: наименование турфирмы (туроператора), адрес, номер тура, страна, количество дней отдыха, стоимость, периодичность тура.

Вариант 27

Таблица с информацией о репертуаре кинотеатров: наименование кинотеатра, наименование фильма, период проката, признак (отечественный или зарубежный), длительность фильма, цена билета.

Вариант 28

Таблица с информацией о маршрутах поездов пригородного сообщения: наименование маршрута, номер поезда, периодичность маршрута, время отправления поезда, цена билета.

Вариант 29

Таблица с информацией об оформленных заказах в магазине «Компьютер-маркет»: дата оформления заказа, номер заказа, ФИО заказчика, домашний адрес, стоимость покупки, номер счета-фактуры (перечень приобретенных комплектующих).

Вариант 30

Таблица с информацией учета больничных листов в поликлинике: номер больничного листа, дата начала больничного, дата окончания больничного, ФИО врача, ФИО больного, домашний адрес, место работы.

Вариант 31

Таблица с информацией учета услуг стоматолога: дата оказания услуги, ФИО пациента, домашний адрес, вид услуги (стоматология или протезирование), наименование услуги, стоимость услуги.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Дюбуа, П. MySQL. Полное и исчерпывающее руководство по применению и администрированию баз данных MySQL 4, а также программированию приложений : [пер. с англ.] / П. Дюбуа. – М. : Вильямс, 2004. – 1056 с.

Дюбуа, П. MySQL. Сборник рецептов : [пер. с англ.] / П. Дюбуа. –СПб. : Символ-Плюс, 2004. – 1056 с.

Хольцнер, С. PHP в примерах : [пер. с англ.] / С. Хольцнер – М. : ООО «Бином-Пресс», 2007. – 352 с.

Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. А. Прохоренок. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2009. – 880 с.

Мотев, А. А. Уроки MySQL. Самоучитель / А. А. Мотев. – СПб. : БХВ-Петербург, 2006. – 208 с.

Документация по MySQL // База данных MySQL [Электронный ресурс]. – М., 2000. – Режим доступа : <http://www.mysql.ru/docs/>.

PHP manual // PHP : Hypertext Preprocessor [Electronic resource]. – US, 1997. – Mode of access : <http://www.php.net/manual/ru/>.

СОДЕРЖАНИЕ

Пояснительная записка	3
1. Основные сведения о СУБД MySQL	4
1.1. Преимущества хранения данных в базах	4
1.2. Состав дистрибутива СУБД MySQL	5
1.3. Основные преимущества СУБД MySQL	6
1.4. Терминология СУБД	7
1.5. Понятие реляционной базы данных	8
1.6. Терминология языка запросов SQL	12
2. Основные приемы работы в СУБД MySQL	12
2.1. Использование командной строки для работы с БД MySQL	12
2.1.1. Варианты подключения к серверу MySQL	12
2.1.2. Запуск программы-клиента <code>mysql</code> , установка соединения с сервером MySQL	13
2.1.3. Ввод запросов (SQL-команд)	18
2.1.4. Правила редактирования команд в окне командной строки	21
2.1.5. Сохранение запросов и результатов их выполнения в файл текстового редактора	22
2.1.6. Синтаксис комментариев	23
2.1.7. Правила присвоения имен в СУБД MySQL	23
Лабораторная работа 1. Работа с БД MySQL через командную строку	24
2.2. Создание базы данных	25
2.2.1. Команда создания базы данных	25
2.2.2. Просмотр списка баз данных	25
2.2.3. Команда активизации (выбора) базы данных	26
2.3. Создание таблиц в базе данных	27
2.3.1. Команда создания таблиц в базе данных	27
2.3.2. Типы данных в столбцах таблицы	31
2.3.3. Выбор типа данных для поля	36
2.3.4. Ключи и индексы	38
2.3.5. Пример создания таблицы в базе данных	40
2.3.6. Просмотр структуры созданных таблиц	41

2.3.7. Команды удаления таблиц и баз данных	43
Лабораторная работа 2. Создание таблиц в базе данных	25
2.4. Технологии работы с таблицами.....	26
2.4.1. Команда изменения структуры таблиц.....	26
2.4.2. Команда добавления записей в таблицу.....	27
2.4.3. Команда удаления записей из таблицы	28
Лабораторная работа 3. Технологии работы с таблицами в базе данных	29
2.4.4. Команда выборки записей из таблицы по заданным критериям	29
2.4.5. Команда обновления значений столбцов	34
Лабораторная работа 4. Выборка данных из таблиц.....	35
2.5. Создание структуры таблицы с помощью программы mysqldadmin. Порядок действий по подключению к серверу баз данных MySQL и созданию таблицы.....	36
Лабораторная работа 5. Создание таблиц с помощью программы mysqldadmin.....	38
3. Приемы работы с SQL-сервером средствами PHP.....	38
3.1. Основные PHP-функции для работы с сервером баз данных.....	38
3.1.1. Принципы работы с базой данных через веб-интерфейс.....	38
3.1.2. Функция установки соединения с сервером баз данных	39
3.1.3. Функция закрытия соединения с SQL-сервером	40
3.1.4. Функция указания имени открываемой базы данных.....	40
3.1.5. Создание запросов к базе данных	41
3.1.6. Функции для отображения результатов запросов к базе данных	43
3.1.7. Изменение данных.....	45
3.1.8. Дополнительные функции PHP для обработки результатирующих наборов	46
Лабораторная работа 6. Работа с SQL-сервером средствами PHP.....	49
3.2. Использование HTML-форм для передачи параметров в скрипты PHP. Реализация передачи параметров методом POST.....	50
Лабораторная работа 7. Использование HTML-форм для передачи параметров в скрипты PHP	54
3.3. Проектирование пользовательской базы данных с веб-интерфейсом.....	55
Задания для самостоятельной работы	116
Список рекомендуемой литературы	120

Учебное издание

ИНФОРМАЦИОННЫЕ РЕСУРСЫ

**Практикум
к лабораторным занятиям для студентов специальности
1-26 03 01 «Управление информационными ресурсами»**

В двух частях

Часть 2

**Технологии работы с клиент-серверными СУБД
(на примере СУБД MySQL)**

Авторы-составители:

Зяц Татьяна Александровна

Зяц Вячеслав Михайлович

Редактор М. П. Герасенко
Технический редактор Н. Н. Короедова
Компьютерная верстка И. А. Козлова

Подписано в печать 03.01.11. Бумага типографская № 1.
Формат 60 × 84 ¹/₁₆. Гарнитура Таймс. Ризография.
Усл. печ. л. 7,21. Уч.-изд. л. 7,10. Тираж 130 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический университет
потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0494302 от 04.03.2009 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический университет
потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.