Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пермский государственный аграрно-технологический университет
имени академика Д.Н. Прянишникова»

УТВЕРЖДАЮ
Заведующий кафедрой
Информационных технологий и программной инженерии
_____/Е.А. Муратова/
«___» _____20__ г.

МЕТОДИЧЕСКОЕ ПОСОБИЕ
по выполнению курсового проекта

Учебная дисциплина
Программная инженерия объектно-ориентированного программирования

Направление подготовки (специальность)
09.03.04 Программная инженерия

Профиль подготовки
Программная инженерия

Квалификация (степень) выпускника
бакалавр

Пермь 2020

Содержание

1. Задание на курсовое проектирование 3
2. Требования к оформлению 4
3. Порядок подготовки и защиты курсового проекта 5
4. Методические указания по разработке проекта 6
5. Приложение 1. Образец оформления титульного листа 46
6. Приложение 2. Образец оформления листа рецензии 47
7. Приложение 3. Порядок работы с хостингом университета 48
   1. Как подключаться к БД через панель phpMyAdmin.
   2. Как подключаться к БД через редактор MySQL WorkBench.
   3. Как наладить FK для связи таблиц в phpMyAdmin.
   4. Как экспортировать БД из phpMyAdmin и добавить в WorkBench.

1. Задание на курсовое проектирование.

Разработка клиент-серверного приложения.

Необходимо разработать приложение по управлению базой данных. База данных размещена на хостинге в сети интернет. База данных содержит несколько связанных таблиц. Поля таблиц содержат разнородную информацию: текст, числовые данные, данные типа дата/время, рисунки. В приложении должно быть несколько форм, обеспечивающих интуитивно понятное обращение к полям таблиц базы данных. Должен быть разработан пользовательский интерфейс, включающий обработку клавиатурных событий и событий мыши. Приложение должно решать задачи отбора данных по критериям, выбираемых пользователем, фильтрации, сортировки данных.

Предметную область базы данных выбрать в соответствии с последней цифрой в шифре в зачетной книжке:

1. Сеть магазинов бытовой техники.
2. Сеть автосалонов.
3. Отдел кадров (подразделения фирмы, должности, количество ставок, зарплата, ФИО сотрудников, адреса, телефоны, возраст, стаж).
4. Афиша сети кинотеатров.
5. Сеть магазинов «Туризм, охота и рыбалка».
6. Сеть продуктовых магазинов.
7. Сеть книжных магазинов.
8. Рейтинг студентов.
9. Сеть салонов красоты.
10. Сеть зоомагазинов.

Для сети (магазинов и т.п.) поля таблиц (выбирать к себе в проект по необходимости или добавить недостающие поля): адреса, график работы, телефоны, название товара (услуги), наличие (текущее), количество (шт./кг), цена, производитель, год выпуска, характеристики, фото.

С разрешения преподавателя тему курсового проекта можно сменить.

2. Требования к оформлению.

Объем реферата 20-30 страниц (листы А4 формата, сброшюрованы).

1) Титульный лист – 1 стр.,

2) Содержание – 1 стр.,

3) Постановка задачи на проектирование – 1-2 стр.

Следует описать решаемую приложением задачу, можно с иллюстрациями. Следует обосновать выбор технологий для реализации проекта.

4) Разработка структуры базы данных.

Следует описать порядок создания базы данных со скриншотами входящих в базу таблиц. Следует раскрыть структуру каждой таблицы и связи между полями разных таблиц. Следует приложить модель базы данных со связами между таблицами.

5) Разработка приложения.

Разработка интерфейса пользователя.

Организация избирательного отображения данных (SQL).

Настройка внешнего вида отображения необходимых отдельных данных или таблиц, обеспечение возможности сортировки и фильтрации данных.

6) Инструкция пользователя.

Краткое описание порядка работы с программой со скриншотами внешнего вида главной формы приложения и иных необходимых элементов интерфейса. Возможно демонстрация работы некоторых элементов работы меню или кнопок сортировки, редактирования и т.п.

7) Заключение.

Краткое описание проделанной работы, возможностей и достоинств разработанного приложения, затруднений в использовании технологий и перспектив к его совершенствованию. Возможно итоги проведённых испытаний приложения.

8) Список литературы – 1 стр.

9) Приложения – 5-15 стр.

1. Текст программы.
2. Тексты SQL-запросов.

3. Порядок подготовки и защиты курсового проекта.

Для студентов заочного отделения за две недели до сессии необходимо прислать в секретариат заочного отделения (для кафедры Информационных технологий и программной инженерии ПГАТУ) проект в бумажном варианте.

Для студентов очного отделения в установленный преподавателем срок необходимо сдать на кафедру Информационных технологий и программной инженерии проект в бумажном варианте.

Все студенты (очники, заочники) в момент защиты обязаны иметь при себе электронный вариант курсового проекта. Электронный вариант должен содержать папку с файлами:

1) текст реферата;
2) все файлы проекта, включая файл и базы данных.

Общие требования к дизайну программы (отсутствие или наличие должно быть оправданным):

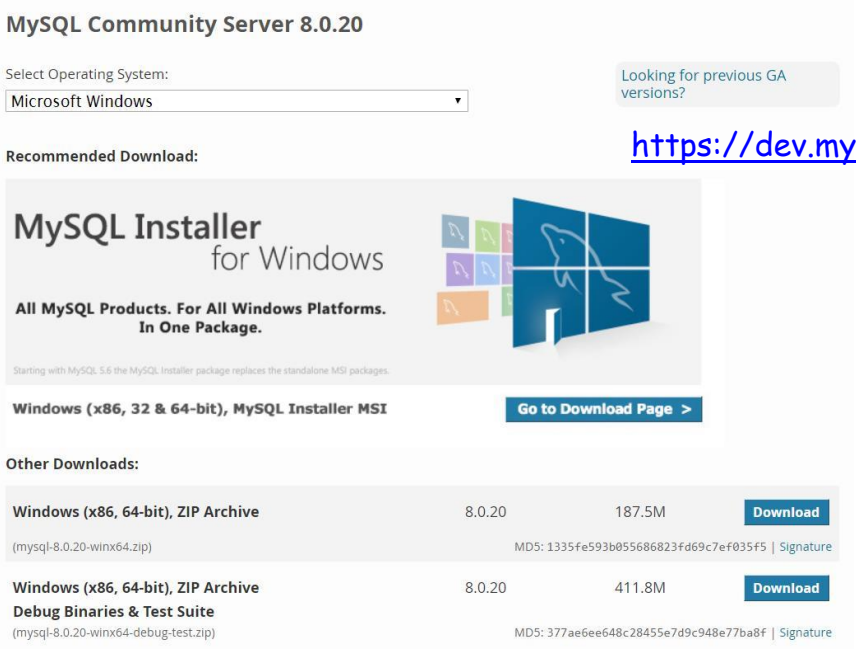
1) наличие в программе информации об авторе и версии программы;
2) наличие уникальной иконки приложения;
3) наличие главного и контекстного меню;
4) наличие всплывающих подсказок;
5) оптимизированные по размеру и выполненные в гармоничной цветовой гамме формы, аккуратное размещение компонентов на форме приложения;
6) контекстное изменение вида приложения путем динамического изменения свойств компонентов: видимости, доступности, размера, положения, цвета и т.п.;
7) разнообразное использование обработчиков событий клавиатуры и мыши;
8) наличие в приложении панели статуса (строки состояния) с текущей информацией;
9) наличие в процедурах проверок на возможные ошибки ввода/вывода, корректности данных и т.п.;
10) наличие в программе справки о функциях программы.

4. Методические указания по разработке приложения.

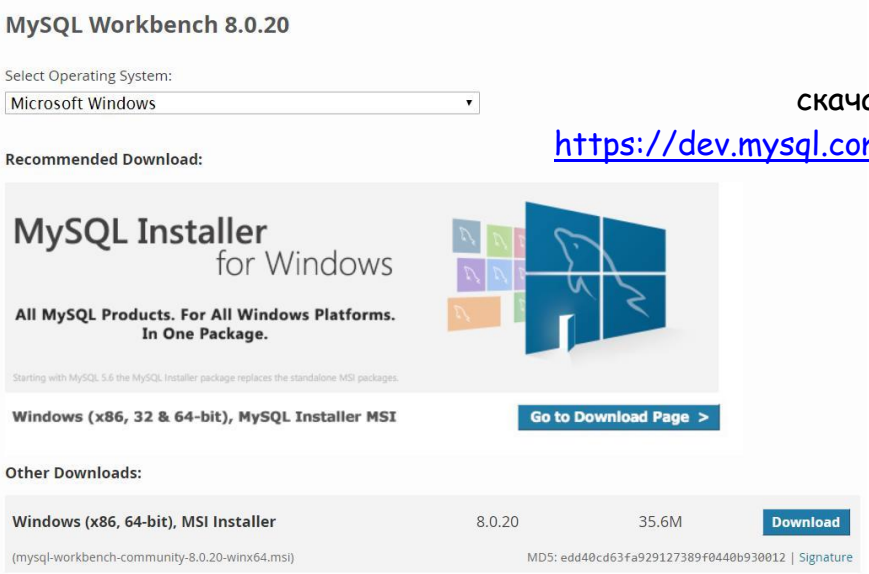
Клиент-серверное приложение с базой данных MySQL

[Введение. Об архитектурах серверных приложений]
[Часть 1. Когда БД уже есть. Связываем MySQL + C#]
[Часть 2. Создаём свою БД, подсоединяем её и используем]
[Часть 3. Некоторые примеры структуры базы данных и запросов к ней]

Для справки – откуда грузить:



скачать сервер MySQL:
https://dev.mysql.com/downloads/mysql/



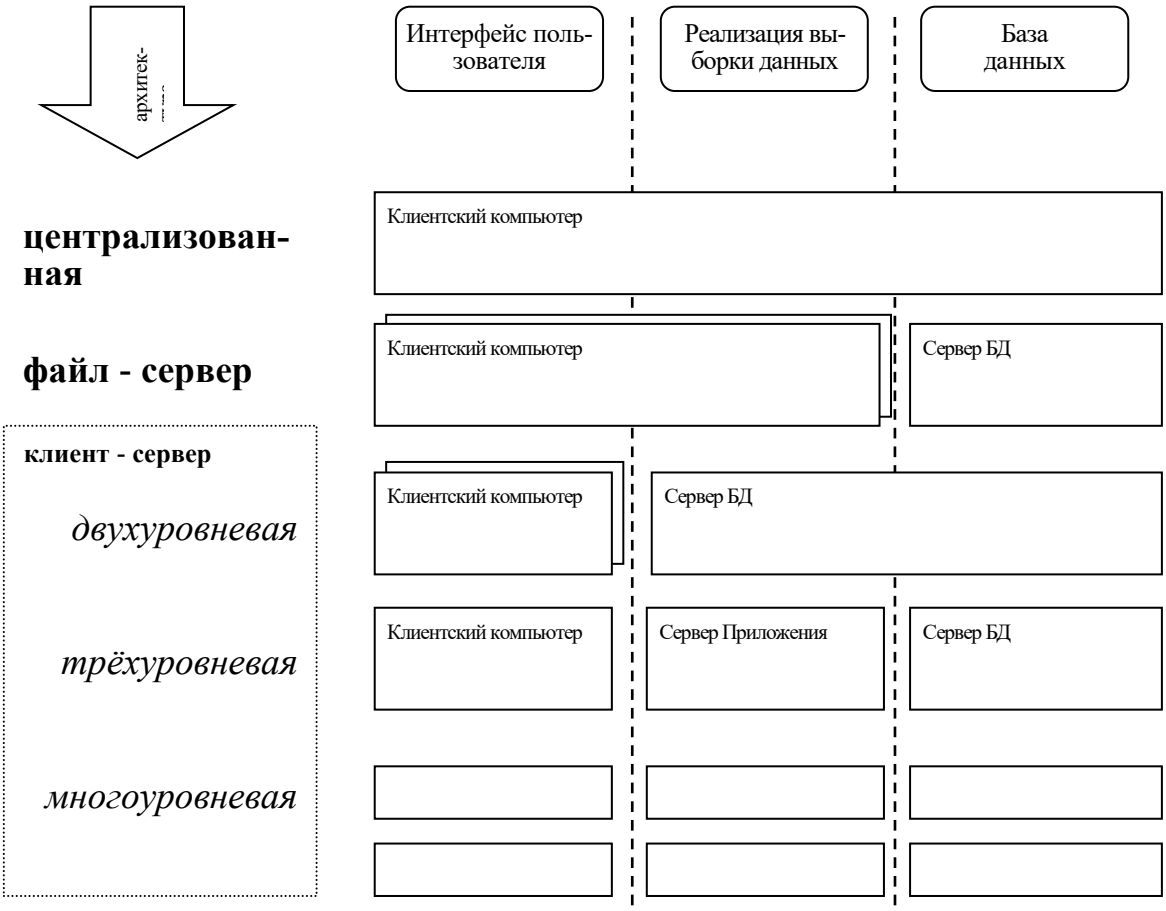
скачать редактор Workbench:
https://dev.mysql.com/downloads/workbench/

Введение. Об архитектурах серверных приложений

Разновидности архитектур в общем виде:

- централизованная система;
- архитектура «файл-сервер»;
- архитектура «клиент-сервер»;
  - двухуровневая,
  - трёхуровневая,
  - многоуровневая.

Схема в помощь:



Поясняющий текст:

<u>Централизованная система</u> – приложение и БД на одном компьютере, например, связка Lazarus (Delphi) + БД Access.

<u>Архитектура «файл-сервер»</u> – на удалённом сервере хранится БД, а приложение на клиентском компьютере. Нагрузка на сеть повышена, так как во время работы на «клиента» загружается не выборка из БД, а сама БД.

Двухуровневая архитектура «клиент-сервер» – на удалённом сервере хранится БД и установлена СУБД, а на клиенте только интерфейс пользователя, через который формируются SQL-запросы – они отправляются через сеть к СУБД, она их обрабатывает и возвращает через сеть к клиенту только выборку из БД (например, C# + MySQL).

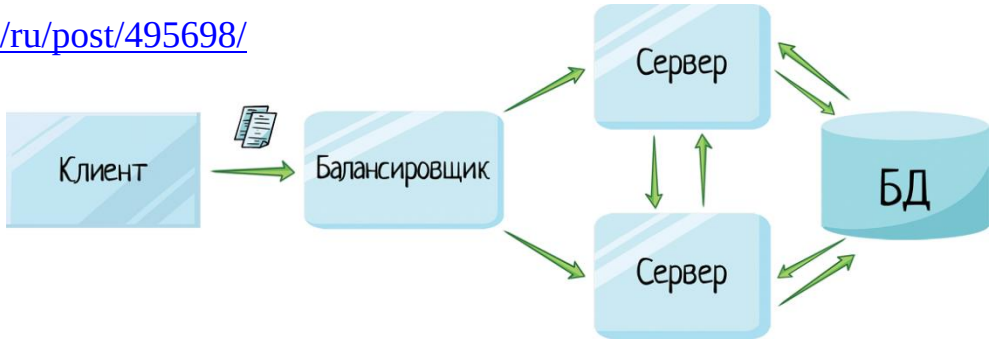
Трёхуровневая архитектура «клиент-сервер» – на одном удалённом сервере хранится БД, на клиенте реализован интерфейс пользователя, но нет непосредственной связи между этими уровнями – между ними есть промежуточное звено: *сервер приложения*, где и реализована бизнес-логика, именно там (тоже удалённо от клиента) формируются запросы к БД и принимаются выборки данных от БД (например, php + MySQL). Преимущества:

- функции обработки данных можно использовать в нескольких проектах – нет дублирования кода, экономия ресурсов;
- независимость сервера БД и сервера Приложения;
- можно дублировать ПО для одного проекта на нескольких серверах Приложений – возрастает надёжность и скорость работы при перегрузках.

Многоуровневая архитектура «клиент-сервер» – используется для крупных и территориально-распределённых предприятий. Есть несколько филиалов предприятия с локализованными трёхуровневыми архитектурами, объединёнными в общую систему. Ведутся полная БД для всего предприятия и копии БД, адаптированные под функционал филиалов. Репликация данных может проходить синхронно (на всех серверах одновременно) и асинхронно (периодически, по расписанию).

Замечательный вариант статьи про клиент-серверную архитектуру читайте тут:

https://habr.com/ru/post/495698/



Чтобы понять, как это работает и, особенно, чтобы понять почему что-то из написанного не работает, нужно всё это проделать своими руками…

Часть 1. Когда БД уже есть. Связываем MySQL + C#

Этап 1 – настройка среды Visual Studio.

Что нужно для связки MySQL + C#? Нужно чтобы в Visual Studio появилась возможность (по умолчанию нет её) подключать через «Обозреватель решений» ссылку на DLL-библиотеку, которая имеет классы для работы с MySQL. Возможны два варианта подключения. Саму библиотеку (`MySql.Data.dll`) можно заранее скачать из сети с учётом версии фреймворка (.NET) того компьютера, на котором собираетесь работать с БД (можно использовать версию библиотеки ниже, чем фреймворк на компьютере, но не выше), например, версии 4.0 и 4.5 можете взять по прямой ссылке:

`https://pcoding.ru/dll/MySQL/net40/MySql.Data.dll`
`https://pcoding.ru/dll/MySQL/net45/MySql.Data.dll`

или закачать самую последнюю версию, подходящую именно для вашей системы, используя возможности самого Visual Studio через NuGet:

– создайте обычное приложение с одной формой,
– в обозревателе решений нажмите правой клавишей мыши по опции «Ссылки»,
– в контекстном меню выберите «Управление пакетами NuGet»,
– в открывшемся слева окне перейдите на вкладку «Обзор»
– в поисковой строке наберите mySql (рис.1),
– после поиска выберите версию MySql.data,
– и нажмите «Установить» (рис.2).

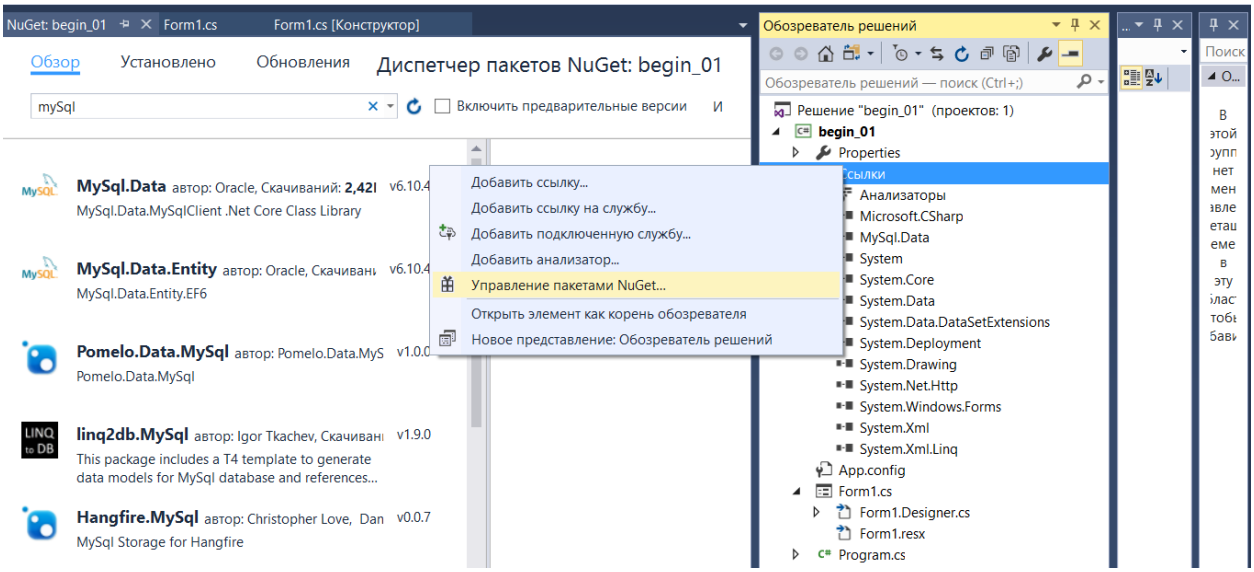


Рис.1.

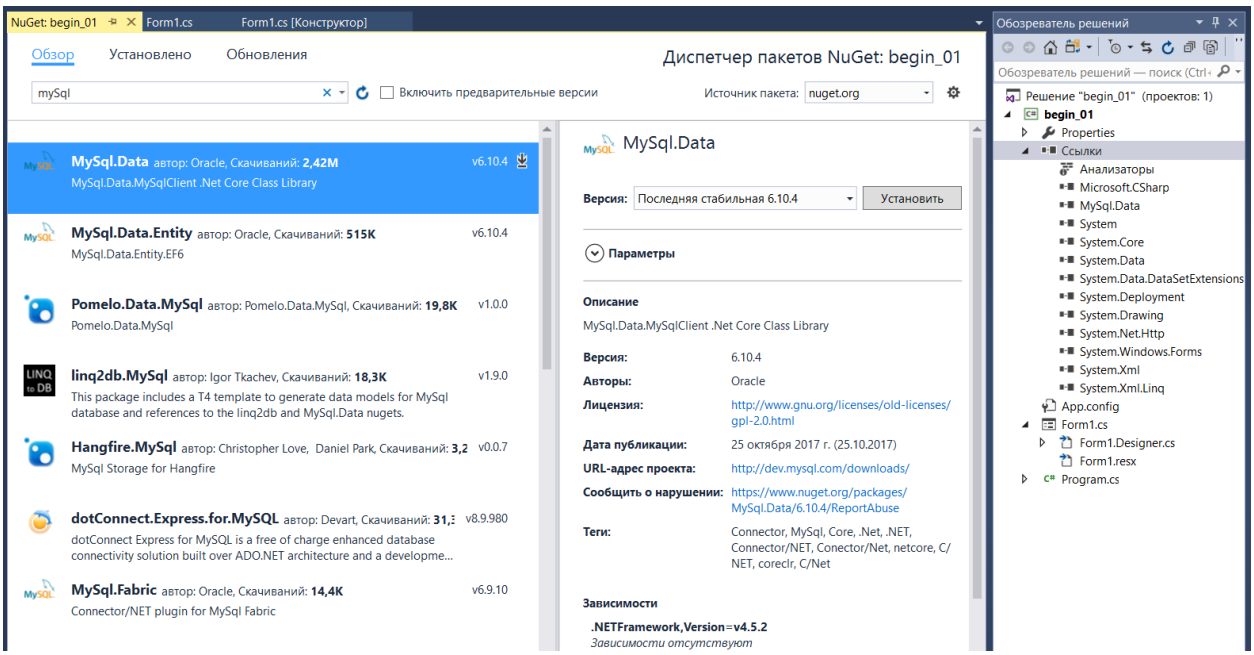


Рис.2.

Если у вас уже есть скачанная ранее библиотека (`MySql.Data.dll`), то можно просто её подключить:

– в «Обозревателе решений» правой клавишей мыши кликаете по опции «Ссылки»,

– выбираете «Добавить ссылку» (рис.3),

– далее жмёте клавишу «Обзор», выбираете путь и сам файл (рис.4), нажимаете «ОК».

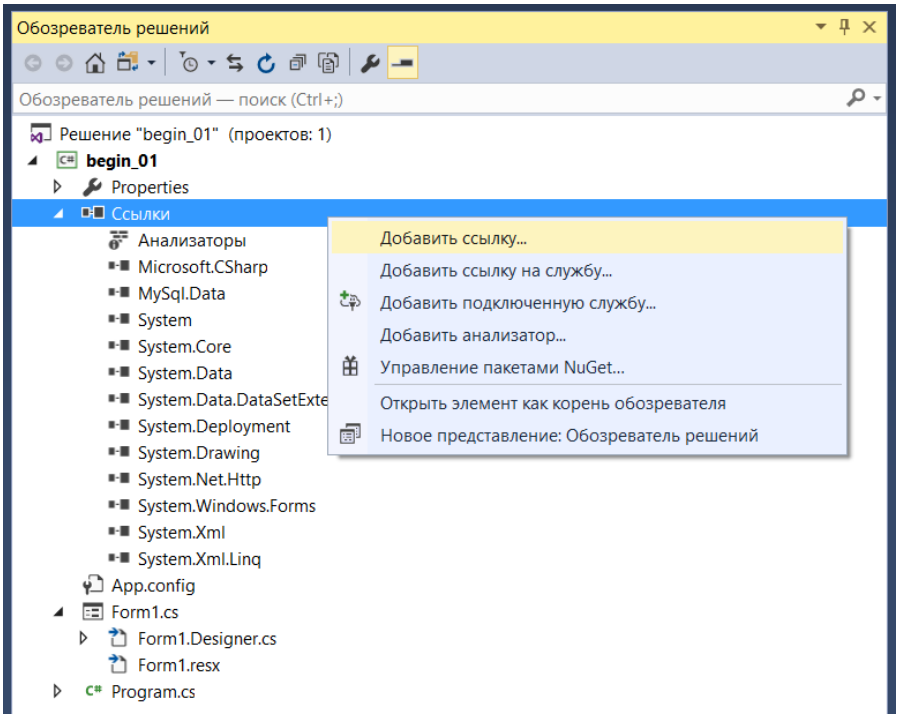


Рис.3

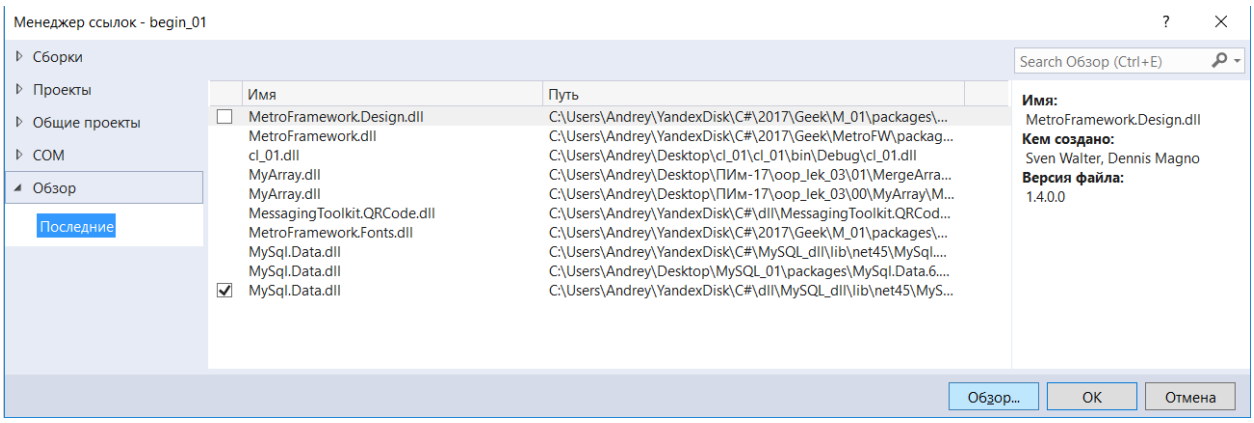


Рис.4.

Когда вы проделаете один из выше приведённых вариантов подключения библиотеки, то в «Обозревателе решений» вы увидите новую строчку – MySql.Data (см. на рис.3), но для использования классов из этой библиотеки в тексте кода (в самом верху) каждого создаваемого модуля к формам вашего приложения ещё нужно будет добавить строчку:

```
using MySql.Data.MySqlClient;
```

Этап первичной подготовки приложения на этом закончен.

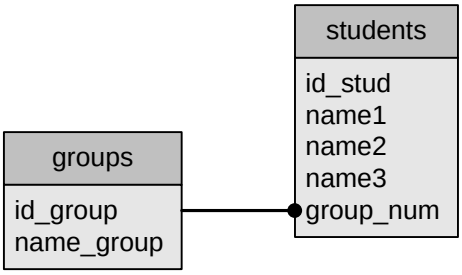
Этап 2 – подсоединимся к БД, считаем данные и отобразим на форме.

После подключения библиотеки можно уже приступать работать с самой базой данных MySQL (если, конечно, её кто-то ранее создал). Для начала так и поступим. Для реализации этой задачи у меня есть специальная пробная база данных – она состоит из двух таблиц:

– учебные группы – groups,

– студенты – students,

связанных отношением «один-ко-многим» (id_group – group_num):



Чтобы подключить программу на C# к сетевой базе данных, нужны такие параметры:

– адрес (хостинг) БД в сети,

– идентификатор (имя) БД,

– имя пользователя,

– пароль пользователя

– кодировка (это необязательно).

Разместите на форме кнопку, сгенерируйте для неё обработчик события «клик мышкой» и в нём создайте объект подключения к БД с такими параметрами:

```
MySqlConnectionStringBuilder db;
db = new MySqlConnectionStringBuilder();
db.Server = "mysql95.1gb.ru";   // хостинг БД
db.Database = "gb_psis";      // Имя БД
db.UserID = "gb_psis";        // Имя пользователя БД
db.Password = "ca8484adc89a";   // Пароль пользователя БД
db.CharacterSet = "utf8";       // Кодировка Базы Данных
```

и сгенерируйте строку подключения:

```
MySqlConnection conn;
conn = new MySqlConnection(db.ConnectionString);
```

Чтобы убедиться в корректности подключения, попытаемся открыть БД и уведомим пользователя о результатах.

```
try
{
    conn.Open();
    MessageBox.Show("Подключение к БД установлено");
}
catch (Exception ex)
{
    MessageBox.Show("Проблемы с подключением к БД \n\r" + ex.ToString());
}
```

Все эти строчки следует разместить в обработчике события клик мыши по клавише:

```
private void button1_Click(object sender, EventArgs e)
```

Апробируйте работу программы по подключению. Обратите внимание, что должно быть в наличии подключение к сети интернет и, после нажатия на клавишу Button1, подключение проходит не мгновенно, а занимает пару секунд. Если удалось установить подключение, то можно подумать уже о начале работы с данными.

Что же можно сделать для начала? Можно, например, вывести что-нибудь из таблицы groups – это будет просто список групп.

Пришло время уточнить, что некоторые переменные следует объявлять, как глобальные, чтобы они были доступны в каждой из функций модуля, поэтому две ранее объявленные переменные (db и conn) перенесите и две новые напишите в самом начале объявления класса формы, а процесс настройки параметров подключения БД перенесите в обработчик загрузки формы:

```
public partial class Form1 : Form
```

```
{
    MySqlConnectionStringBuilder db;
    MySqlConnection conn;
    MySqlCommand cmd;
    string sql;

    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        db = new MySqlConnectionStringBuilder();
        db.Server = "mysql95.1gb.ru";   // хостинг БД
        db.Database = "gb_psis";      // Имя БД
        db.UserID = "gb_psis";        // Имя пользователя БД
        db.Password = "ca8484adc89a";   // Пароль пользователя БД
        db.CharacterSet = "utf8";       // Кодировка Базы Данных
        conn = new MySqlConnection(db.ConnectionString);
    }
```

Далее создайте на форме вторую кнопку и сгенерируйте для неё обработчик (клик мышкой). Давайте пока, в качестве первой пробы, просто в текстовое поле (не забудьте установить MultiLine=True) выведем список групп:

```
    private void button2_Click(object sender, EventArgs e)
    {
        sql = "SELECT name_group FROM groups";
        cmd = new MySqlCommand(sql, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        textBox1.Clear();
        while (reader.Read())
        {
            textBox1.Text += reader[0].ToString() + Environment.NewLine;
        }
        reader.Close();
    }
```

В этом обработчике сначала мы создаём строку с SQL-командой, затем формируем из неё объект cmd для трансляции команды в БД и, с помощью метода `ExecuteReader` запускаем её на исполнение, результаты возвращаем в переменную `reader`. Далее очищаем текстовое поле и в цикле читаем все записи, полученные по команде SELECT. В каждой записи может быть несколько полей (это как в одной строке таблицы), нумерация полей начинается с нуля. Какие именно поля, их количество и названия вы сами определили в запросе:

```
sql = "SELECT name_group FROM groups";
```

Итак, у нас есть только одно поле – `name_group` , к нему можно обратиться как по порядковому номеру, так и по имени:

```
textBox1.Text += reader["name_group"].ToString() + Environment.NewLine;
```

Апробируйте оба варианта.

В исследуемой таблице (groups) есть ещё один столбец – id_group, он нужен для того, чтобы в связной таблице хранить не полное наименование групп, а только ссылки на них. Но сейчас вы можете попробовать немного доработать свою программу и вывести в текстовое поле оба столбца таблицы, для чего в запросе нужны изменения:

```
sql = "SELECT id_group, name_group FROM groups";
```

и в организации вывода, например, так:

```
while (reader.Read())
{
    textBox1.Text +=
        reader["id_group"].ToString() + '\t' +
        reader["name_group"].ToString() +
        Environment.NewLine;
}
```

Если, в дальнейшем, мы хотим организовать вывод студентов из определённой группы по выбору пользователя, то имеет смысл список групп загружать в подходящий для этой цели визуальный компонент, например, в ComboBox (добавьте его на форму). Перепишите обработчик второй клавиши следующим образом:

```
sql = "SELECT name_group FROM groups";
cmd = new MySqlCommand(sql, conn);
MySqlDataReader reader = cmd.ExecuteReader();
comboBox1.Items.Clear();
while (reader.Read())
{
    comboBox1.Items.Add(reader["name_group"].ToString());
}
reader.Close();
comboBox1.Text = "Список групп";
```

Теперь уже можно сделать обработчик выбора опции в выпадающем списке (`comboBox1`), который и будет в текстовое поле `textBox1` выводить список фамилий (name1) из выбранной группы (пока номер группы равен 1):

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    sql = "SELECT name1 FROM students WHERE group = 1";
    cmd = new MySqlCommand(sql, conn);
    MySqlDataReader reader = cmd.ExecuteReader();
    textBox1.Clear();
    while (reader.Read())
    {
        textBox1.Text +=
            reader["name1"].ToString() + Environment.NewLine;
    }
    reader.Close();
}
```

Апробируйте работоспособность этого кода – вполне возможно, что в текстовое поле не будет загружено ни одной фамилии, так как пользователи этой

открытой БД (такие же студенты, как и вы) могли удалить все записи из неё. Проследить это обстоятельство совсем несложно. Если программа не зависает, работает корректно, но в текстовом поле нет записей, то внесите в программу следующие изменения – добавьте функцию получения количества записей в таблице и в сам обработчик добавьте несколько строчек:

```
        int getCount(string nameTable)
        {
            sql = "SELECT COUNT(*) FROM " + nameTable;
            cmd = new MySqlCommand(sql, conn);
            return Convert.ToInt32(cmd.ExecuteScalar().ToString());
        }


        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            sql = "SELECT name1 FROM students WHERE group_num = 1";
            cmd = new MySqlCommand(sql, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            textBox1.Clear();
// тут добавка
            textBox1.Text += "Список группы " +
                comboBox1.SelectedItem + ":"+ Environment.NewLine;
//
            while (reader.Read())
            {
                textBox1.Text +=
                    reader["name1"].ToString() + Environment.NewLine;
            }
            reader.Close();
// тут добавка:
            textBox1.Text +=
                "- - -" + Environment.NewLine +
                "Общее кол-во студентов во всех группах = " +
                getCount("students").ToString() + Environment.NewLine;
        }
    }
```

Я полагаю, что указанные изменения стоит реализовать у себя в программе даже если программа и так работала корректно и, не только потому, что вы сами можете в дальнейшем случайно удалить всех студентов из таблицы, но и для расширения спектра используемых методов работы.

Если всё же в группах есть студенты, то всё равно в таком виде этот код работает не совсем так, как мы задумывали, так как в запросе указано выбирать группу с номером 1 вне зависимости от выбора пользователя. Переработаем обработчик (я использую первый, более короткий вариант, но вы можете оставить более полный вариант), вернув сначала первым запросом номер группы по её имени, а уже затем список студентов выбранной группы:

```
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            MySqlDataReader reader;

            sql = "SELECT id_group FROM groups WHERE name_group = " +
                  "'" + comboBox1.SelectedItem + "'";
```

```
        cmd = new MySqlCommand(sql, conn);
        reader = cmd.ExecuteReader();
        reader.Read();
        int numId = Convert.ToInt32(reader[0]); // номер группы
        reader.Close();

        sql = "SELECT name1 FROM students WHERE group_num = " + numId.ToString();
        cmd = new MySqlCommand(sql, conn);
        reader = cmd.ExecuteReader();
        textBox1.Clear();
        while (reader.Read())
        {
            textBox1.Text +=
                reader["name1"].ToString() + Environment.NewLine;
        }
        reader.Close();
    }
```

Это, конечно, не самое лучшее решение, но оно сделано по аналогии с предыдущими решениями и интуитивно понятно, что достаточно для первой работоспособной программы. Возможные варианты доработки таковы:

– когда мы ранее загружали список групп, то нужно вместе с ними загружать и id групп, хранить их в какой-то структуре и, при необходимости, использовать (это сможете сделать сами);

– можно просто сделать вложенный запрос вместо двух запросов, как было в нашем последнем обработчике, и выглядит это так:

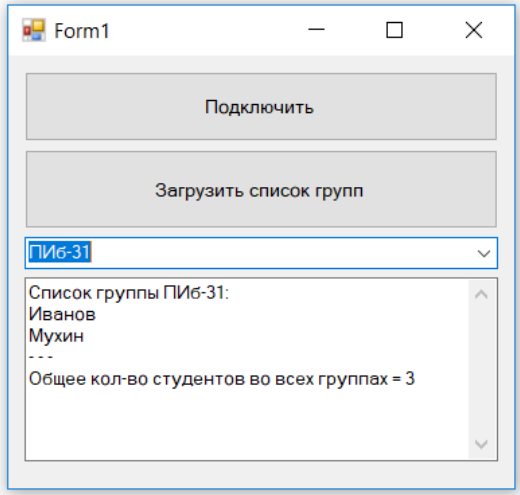
```
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        MySqlDataReader reader;

        sql = "SELECT name1 FROM students WHERE group_num = " +
              "(SELECT id_group FROM groups WHERE name_group = " +
              "'" + comboBox1.SelectedItem + "')";
        cmd = new MySqlCommand(sql, conn);
        reader = cmd.ExecuteReader();
        textBox1.Clear();
        while (reader.Read())
        {
            textBox1.Text +=
                reader["name1"].ToString() + Environment.NewLine;
        }
        reader.Close();
    }
```

Может сложиться такая ситуация (в другой БД), при которой внутренний подзапрос вернёт не одно, а несколько значений, в этом случае необходимо внести изменения в наш запрос, заменив символ сравнения (“=”) на команду проверки вхождения в множество («IN»):

```
        sql = "SELECT name1 FROM students WHERE group_num IN " +
              "(SELECT id_group FROM groups WHERE name_group = " +
              "'" + comboBox1.SelectedItem + "')";
```

Проверьте, что и для нашей БД это работает корректно, ведь множество может состоять и из одного элемента.

Теперь я приведу пример всех уже готовых функций и скриншот экрана программы во время выполнения запроса, чтобы вы смогли сравнить результаты:




```
using System;
using System.Data;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace beginMySQL
{
    public partial class Form1 : Form
    {
        MySqlConnectionStringBuilder db;
        MySqlConnection conn;
        MySqlCommand cmd;
        string sql;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            db = new MySqlConnectionStringBuilder();
            db.Server = "mysql95.1gb.ru";    // хостинг БД
            db.Database = "gb_psis";       // Имя БД
            db.UserID = "gb_psis";         // Имя пользователя БД
            db.Password = "ca8484adc89a";    // Пароль пользователя БД
            db.CharacterSet = "utf8";        // Кодировка Базы Данных
            conn = new MySqlConnection(db.ConnectionString);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (conn.State == ConnectionState.Closed)
                {
                    conn.Open();
                }
                MessageBox.Show("Подключение к БД установлено");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Проблемы с подключением к БД \n\r" + ex.ToString());
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            sql = "SELECT name_group FROM groups";
            cmd = new MySqlCommand(sql, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            comboBox1.Items.Clear();
            while (reader.Read())
            {
                comboBox1.Items.Add(reader["name_group"].ToString());
            }
            reader.Close();
            comboBox1.Text = "Список групп";
```

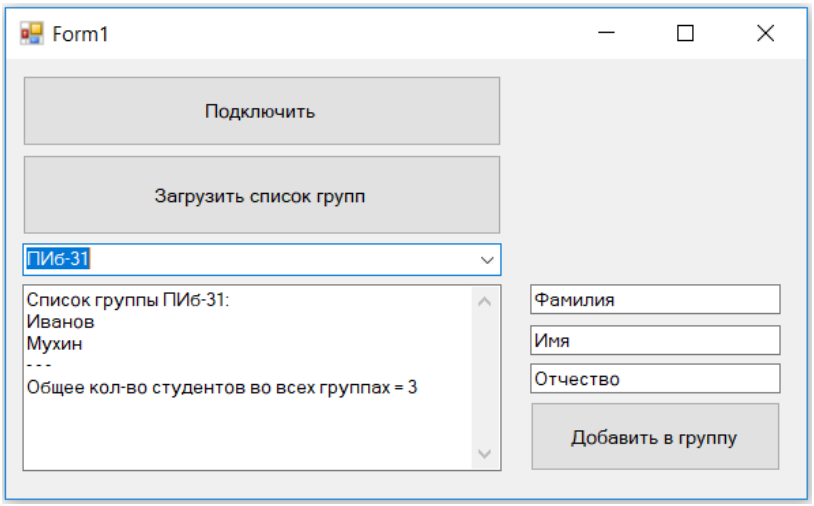
```
        }

        int getCount(string nameTable)
        {
            sql = "SELECT COUNT(*) FROM " + nameTable;
            cmd = new MySqlCommand(sql, conn);
            return Convert.ToInt32(cmd.ExecuteScalar().ToString());
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            MySqlDataReader reader;

            sql = "SELECT name1 FROM students WHERE group_num IN " +
                  "(SELECT id_group FROM groups WHERE name_group = " +
                  "'" + comboBox1.SelectedItem + "')";
            cmd = new MySqlCommand(sql, conn);
            reader = cmd.ExecuteReader();
            textBox1.Clear();
            textBox1.Text += "Список группы " +
                comboBox1.SelectedItem + ":"+ Environment.NewLine;
            while (reader.Read())
            {
                textBox1.Text +=
                    reader["name1"].ToString() + Environment.NewLine;
            }
            reader.Close();
            textBox1.Text +=
                "- - -" + Environment.NewLine +
                "Общее кол-во студентов во всех группах = " +
                getCount("students").ToString() + Environment.NewLine;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            sql = "INSERT INTO students (name1, name2, name3, group_num) ";
            sql += "VALUES(NULL, '" + textBox1.Text + "')";
            cmd = new MySqlCommand(sql, conn);
            cmd.ExecuteNonQuery();
        }
    }
}
```

Этап 3 – добавляем записи в таблицу БД.

Дальше мы будем апробировать команды по *добавлению*, *изменению* и *удалению* строчек из таблицы «Студенты». Давайте сразу договоримся, что вы не трогаете таблицу с учебными группами (groups), так как эта БД используется не только вами (чтобы и остальные смогли воспользоваться этой методичкой). Итак, чтобы работать с таблицей, нужно понимать её структуру, количество и названия полей. Я приведу скриншот структуры и содержания таблицы students:

id_stud	name1	name2	name3	group_num
1	Иванов	Иван	Иванович	1
2	Сидоров	Сидор	Сидорович	4
3	Мухин	Олег	Николаевич	1

Доработайте форму для реализации функции добавления записи в таблицу:



Создайте функцию для определения номера текущей группы и запрос на добавление записи про студента в таблицу students:

```
int getNumGroup()
{
    MySqlDataReader reader;
    sql = "SELECT id_group FROM groups WHERE name_group = " +
          "'" + comboBox1.SelectedItem + "'";
    cmd = new MySqlCommand(sql, conn);
    reader = cmd.ExecuteReader();
    reader.Read();
    int numId = Convert.ToInt32(reader[0]); // номер группы
    reader.Close();
    return numId;
}

private void button3_Click(object sender, EventArgs e)
{
    sql = "INSERT INTO students (id_stud, name1, name2, name3, group_num) ";
    sql += "VALUES(NULL, " +
        "'" + textBox2.Text + "', " +   // Фамилия
        "'" + textBox3.Text + "', " +   // Имя
        "'" + textBox4.Text + "', " +   // Отчество
        getNumGroup().ToString() + ")"; // номер группы
    cmd = new MySqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
}
```

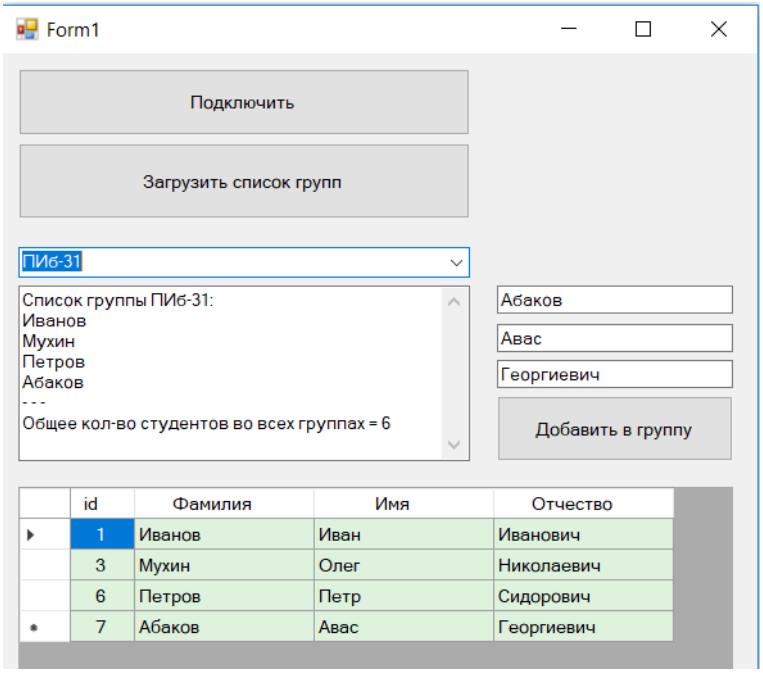
Добавьте несколько студентов в разные группы и проверьте полученный результат.

На этом можно считать, что первые, базовые трудности преодолены, так как вы уже научились связывать сетевую БД с программой, делать запросы по отбору данных из связанных таблиц и отображать полученные результаты на форме приложения.

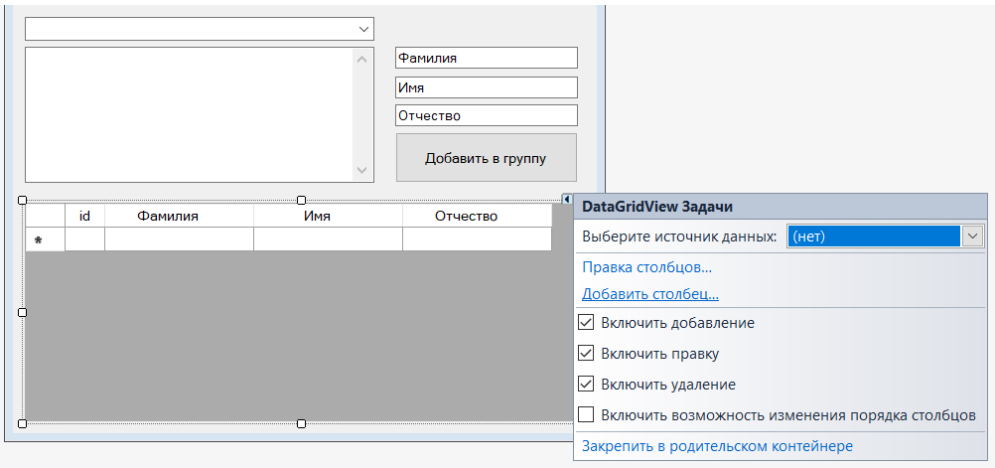
Этап 4 – удаляем и изменяем записи в БД.

Для реализации функций *изменения* и *удаления* записей в БД уже неудобно результаты отображать в обычном, хоть и многострочном, текстовом поле. Пришло время добавить на форму таблицу. Чтобы мне не переписывать полностью программу, я, для реализации функции отбора (SELECT), оставлю на форме многострочное текстовое поле, а для функций редактирования и удаления добавлю DataGridView. После освоения всех предложенных в данной методичке методов вы сами для себя будете выбирать приемлемые подходы.

Настройте внешний вид таблицы по примеру (*уточню, что на рисунке в таблице первая строка является фиксированной, а фиксированных столбцов нет совсем*):



Напомню, что во вновь размещённую таблицу dataGridView можно добавить и настроить столбцы через контекстное меню, вызываемое по нажатию на треугольник в правом верхнем углу таблицы:



Итак, приступим к кодированию функционала. Для начала добавим три кнопки: «Обновить таблицу», «Удалить запись» и «Обновить запись». Сгенерируйте обработчик клика мышки по клавише «Обновить таблицу» и добавьте туда одну строчку кода – вызов функции заполнения таблицы:

```
fillTable();
```

Это вызов функции по выводу списка студентов в таблицу на форме. Сама эта функция пока не описана в коде, но прямо сейчас займёмся этим. Функция должна выполнять две задачи:

– сначала загружаем данные студенческой группы в список,

– затем отображаем этот список в таблице.

Конечно, вы можете обойтись и без списка и сразу после чтения из БД данные размещать в таблице на форме, как мы ранее делали с многострочным текстовым полем. Однако хранить данные в специально созданной для этого структуре и использовать их по необходимости может оказаться удобным. Итак, добавьте описание *структуры* в части, где мы объявляли глобальные переменные:

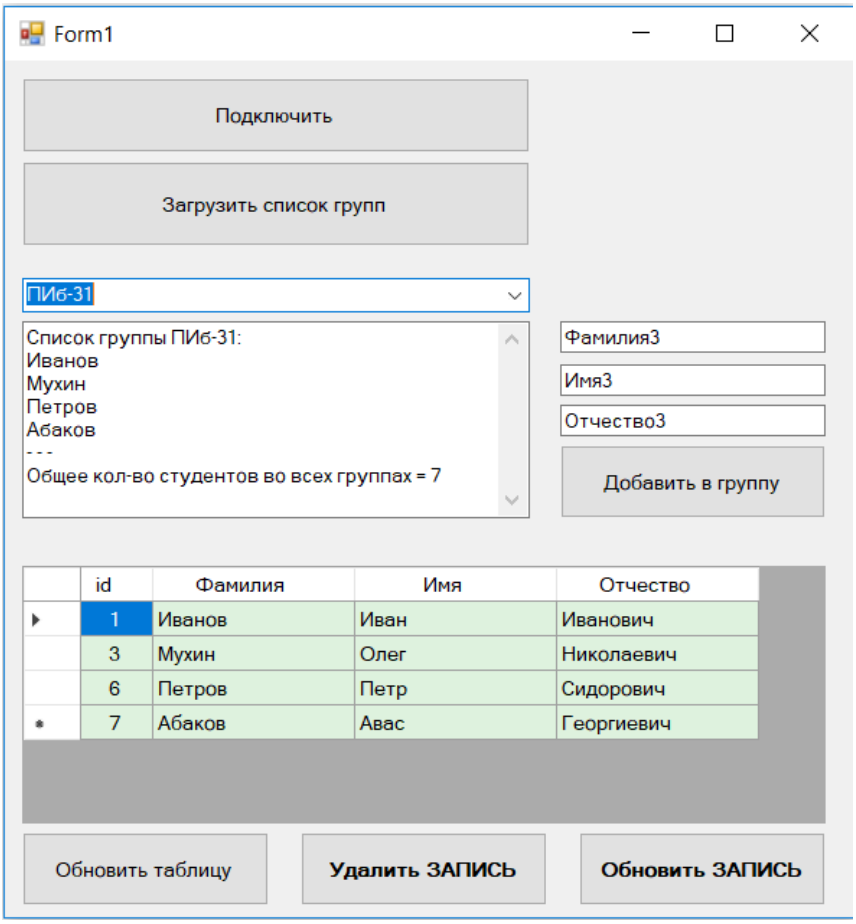
```
struct tableStud
{
    public int tsId;
    public string tsName1, tsName2, tsName3;
}
```

Структура нужна нам для более удобного (структурированного) хранения записей из таблицы студенты. Далее опишите две функции (функция `getTable()` вызывается из функции `fillTable()`) – получить записи и отобразить их на форме в таблице:

```
List<tableStud> getTable()
{
    List<tableStud> tbStud = new List<tableStud>();
    tableStud tmp; // для хранения текущей считанной записи
    tbStud.Clear(); // очистим список
    MySqlDataReader reader; // объект для чтения записей

    sql = "SELECT id_stud,name1,name2,name3 FROM students WHERE group_num IN " +
          "(SELECT id_group FROM groups WHERE name_group = " +
          "'" + comboBox1.SelectedItem + "')";
    cmd = new MySqlCommand(sql, conn);
    reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        tmp.tsId = Convert.ToInt32(reader["id_stud"].ToString());
        tmp.tsName1 = reader["name1"].ToString();
        tmp.tsName2 = reader["name2"].ToString();
        tmp.tsName3 = reader["name3"].ToString();
        tbStud.Add(tmp); // добавим текущую запись в список
    }
    reader.Close();
    return tbStud; // вернём список записей из таблицы
}

void fillTable() // данные из списка переносим в таблицу
```

```
{
    List<tableStud> tbStud = getTable(); // получим список студентов

    // начальные настройки для таблицы
    dataGridView1.Rows.Clear();
    dataGridView1.DefaultCellStyle.BackColor = Color.FromArgb(222, 242, 222);
    dataGridView1.Columns[0].Width = 50;
    dataGridView1.Columns[1].Width = 140;
    dataGridView1.Columns[2].Width = 140;
    dataGridView1.Columns[3].Width = 140;

    dataGridView1.RowCount = tbStud.Count;
    for (int i = 0; i < tbStud.Count; i++)
    {
        dataGridView1.Rows[i].Cells[0].Value = tbStud[i].tsId;
        dataGridView1.Rows[i].Cells[1].Value = tbStud[i].tsName1;
        dataGridView1.Rows[i].Cells[2].Value = tbStud[i].tsName2;
        dataGridView1.Rows[i].Cells[3].Value = tbStud[i].tsName3;
    }
}
```

Если всё было прописано корректно, то вы сможете получить примерно такой результат:



Теперь уже будет гораздо удобнее выполнять задачи по удалению или изменению записей. Например, можно сделать так: удалять запись, если был клик по ней средней клавишей мыши или просто удалять текущую запись по нажатию на специально установленную на форму клавишу удаления:

```
private void button5_Click(object sender, EventArgs e)
{
    int indRow = dataGridView1.CurrentRow.Index; // узнаём текущую строку
    int idStud = Convert.ToInt32(dataGridView1.Rows[indRow].Cells[0].Value);
    sql = "DELETE FROM students WHERE id_stud = '" + idStud.ToString() + "'";
    cmd = new MySqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
}
```

В данном обработчике мы сначала узнаём номер текущей строки в таблице, затем по нему мы определяем идентификатор записи в таблице students, потом формируем SQL-команду на удаление и, собственно, исполняем её. Так как у нас достаточно простой интерфейс пользователя, то после удаления записи следует нажать клавишу «Обновить таблицу» для отображения текущих изменений. Однако, заметно удобнее будет, если вы добавите в последний обработчик в самом конце ещё и строчку:

```
fillTable();
```

В данном варианте сразу после удаления будет происходить обновление отображения данных в таблице.

Завершающая задача данной части изложения – это создание обработчика по внесению изменений в текущую запись. Для простоты описания сделаем это так: у пользователя есть возможность редактировать записи непосредственно в компоненте dataGridView, после редактирования пользователь должен нажать клавишу «Обновить запись» для внесения изменений в сетевую БД (курсор должен находиться в текущей строке таблицы):

```
private void button6_Click(object sender, EventArgs e)
{
    int indRow = dataGridView1.CurrentRow.Index; // узнаём текущую строку
    int idStud = Convert.ToInt32(dataGridView1.Rows[indRow].Cells[0].Value);
    string n1 = dataGridView1.Rows[indRow].Cells[1].Value.ToString();
    string n2 = dataGridView1.Rows[indRow].Cells[2].Value.ToString();
    string n3 = dataGridView1.Rows[indRow].Cells[3].Value.ToString();
    sql = "UPDATE students SET ";
    sql += "name1 = '" + n1 + "', name2 = '" + n2 + "', name3 = '" + n3 + "' ";
    sql += "WHERE id_stud = '" + idStud.ToString() + "'";
    cmd = new MySqlCommand(sql, conn);
    cmd.ExecuteNonQuery();
}
```

Этих знаний вполне достаточно чтобы продумать и разработать полноценное приложение по работе с сетевой базой данных. Осталось дело за малым – научиться самому создавать и размещать базы данных в сети.

Часть 2. Создаём свою БД, подсоединяем её и используем

Что нужно для того, чтобы «создавать и размещать базы данных в сети»? Чтобы создавать (быстро и удобно) – нужен специализированный визуальный редактор (типа MySQL **Workbench**) или имеющаяся на всех хостингах панель для управления базами данных **phpMyAdmin**. В первом варианте после создания базы данных можно её использовать и без хостинга в интернете, а прямо у себя на компьютере, как локальную БД. Это может быть удобно для редактирования и тестирования программы, а строка подключения к локально размещённой БД выглядит крайне просто:

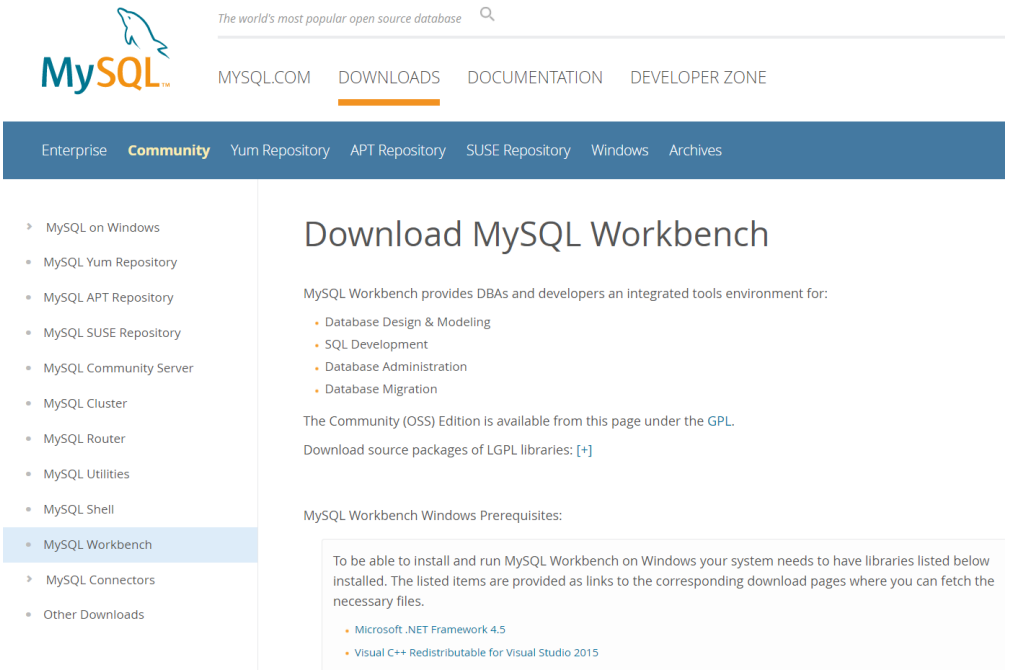
```
string connStr = "server=localhost;user=root;database=students;password=0000;";
```

Конечно, логин и пароль индивидуальны. Однако, потом, тем не менее, необходимо будет потратить время на поиск хостинга и загрузку (может быть и настройку) базы данных. По этой причине, для несложных задач имеет смысл сразу создавать структуру БД непосредственно в сети на хостинге через панель phpMyAdmin.

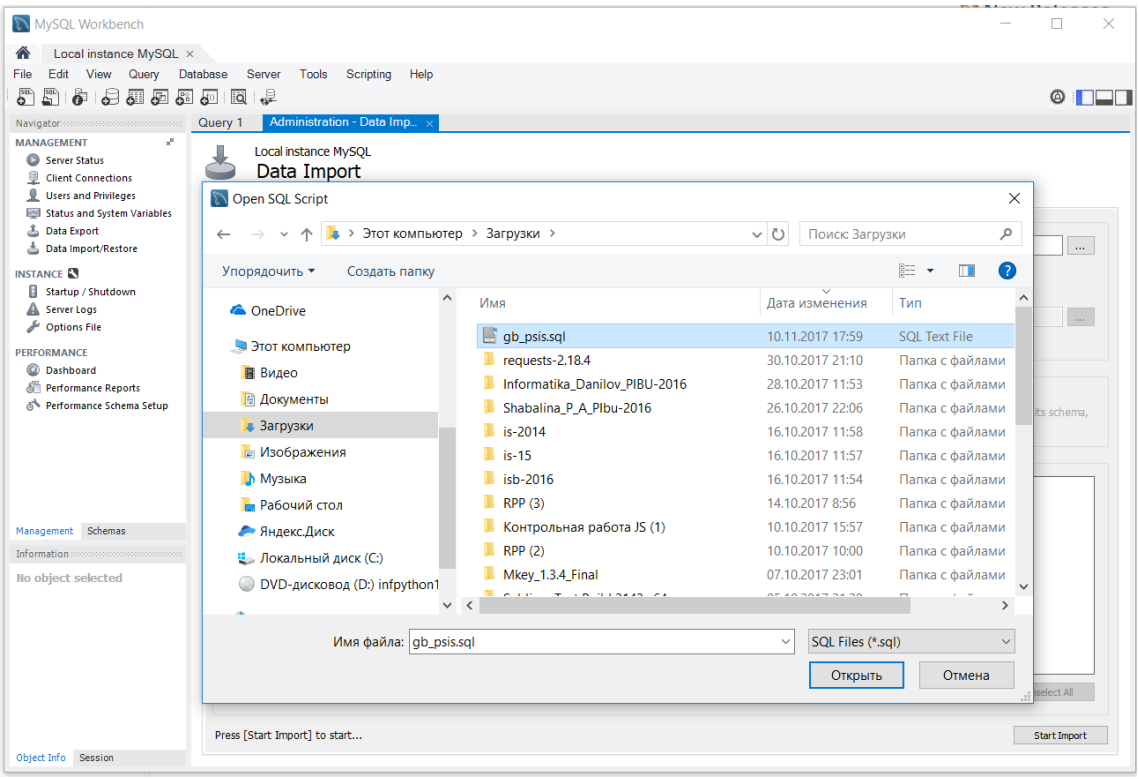
Прежде чем перейдем к практике приведу примеры названных программных продуктов и некоторые их возможности в общем виде.

Обращаю ваше внимание, что в этой части я не буду затрагивать и описывать подробности работы с Workbench, так как работа непосредственно в phpMyAdmin может сэкономить время на разработку при небольшой БД.

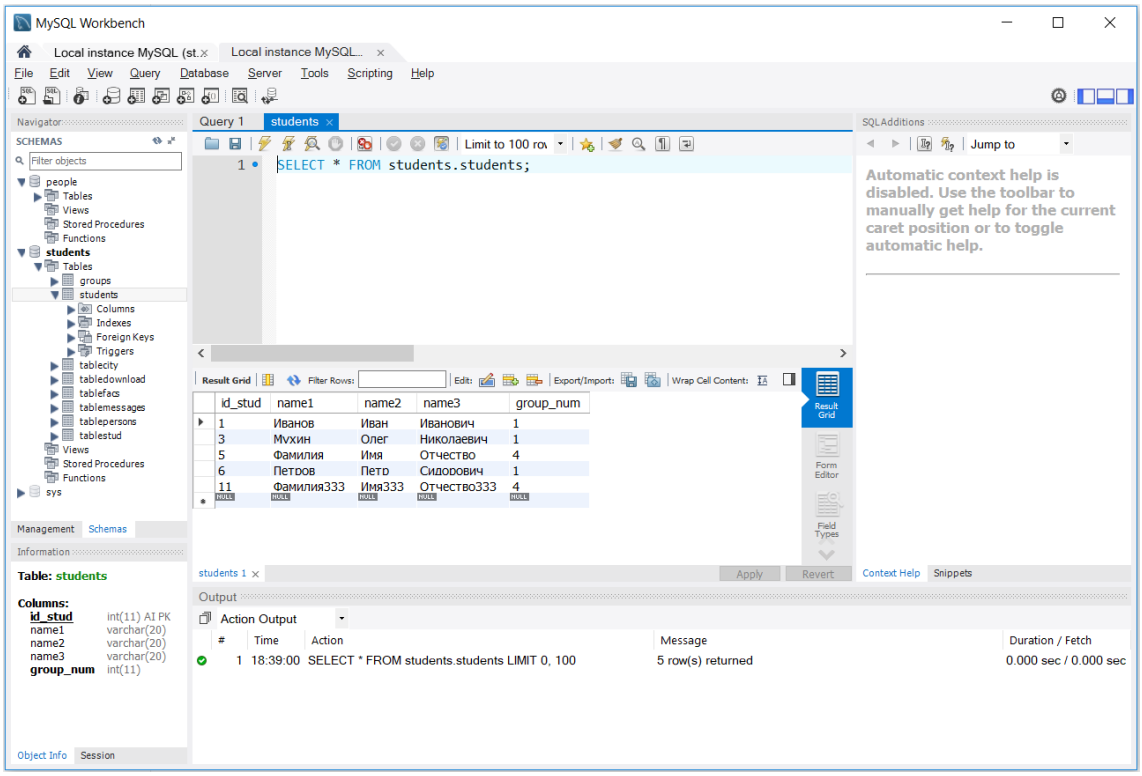
Итак, вот отсюда можно скачать бесплатную версию MySQL Workbench: **`https://dev.mysql.com/downloads/workbench/`**



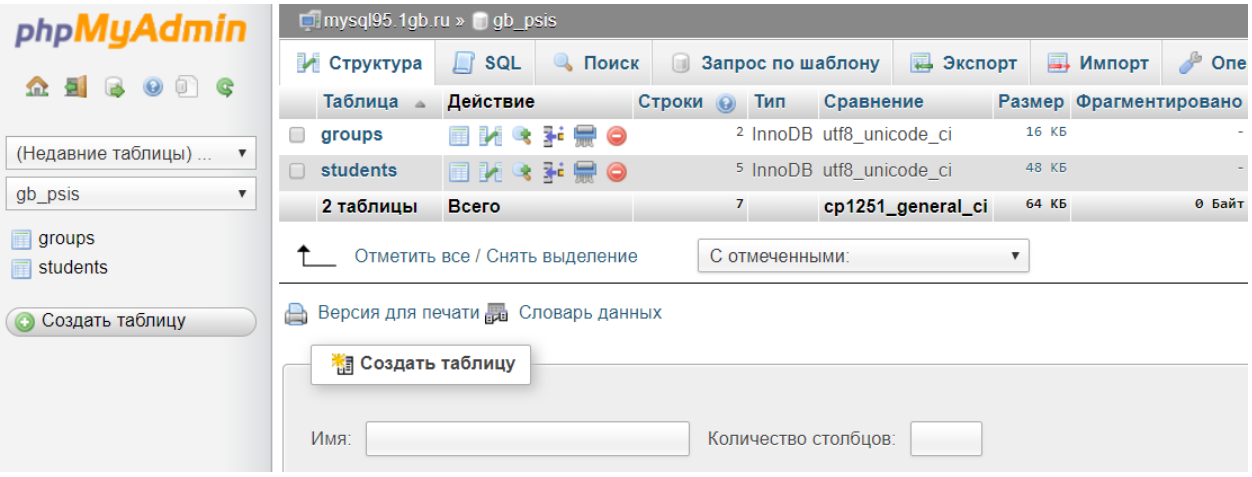
Вот так можно в эту программу импортировать чужую БД, например, созданную в сети через phpMyAdmin (файл gb_psis.sql был как раз сохранён с моего хостинга – эта и есть та база, что мы использовали в первой части):



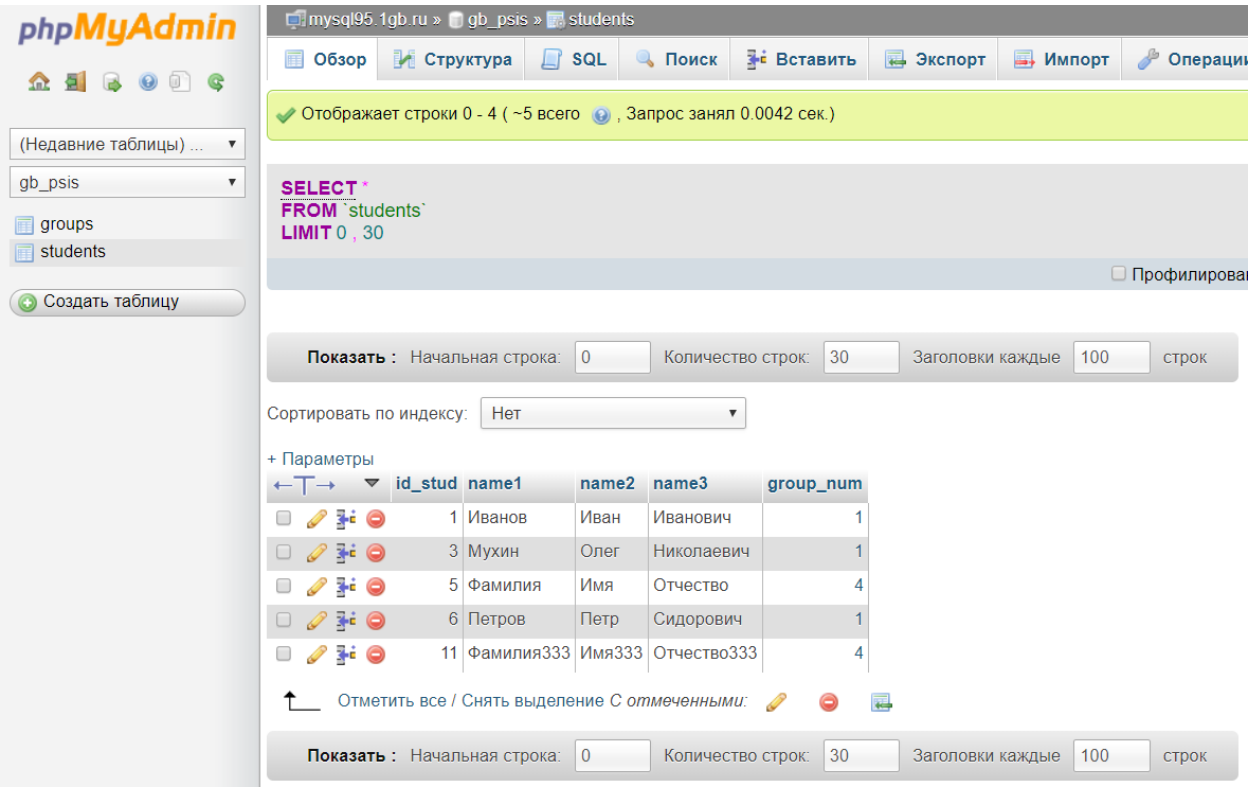
И после импорта с ней уже можно поработать (обратите внимание – в левой части есть проводник по БД и там видны названия таблиц – groups и students):



А вот как эта же БД выглядела во время её разработки непосредственно на хостинге через панель phpMyAdmin (обратите внимание, что в верхней части как раз виден хостинг (и конкретный сервер **`mysql95.1gb.ru`** для хранения БД) и имя БД, что мы использовали в первой части нашей работы):



А так можно просматривать и редактировать содержимое таблиц:

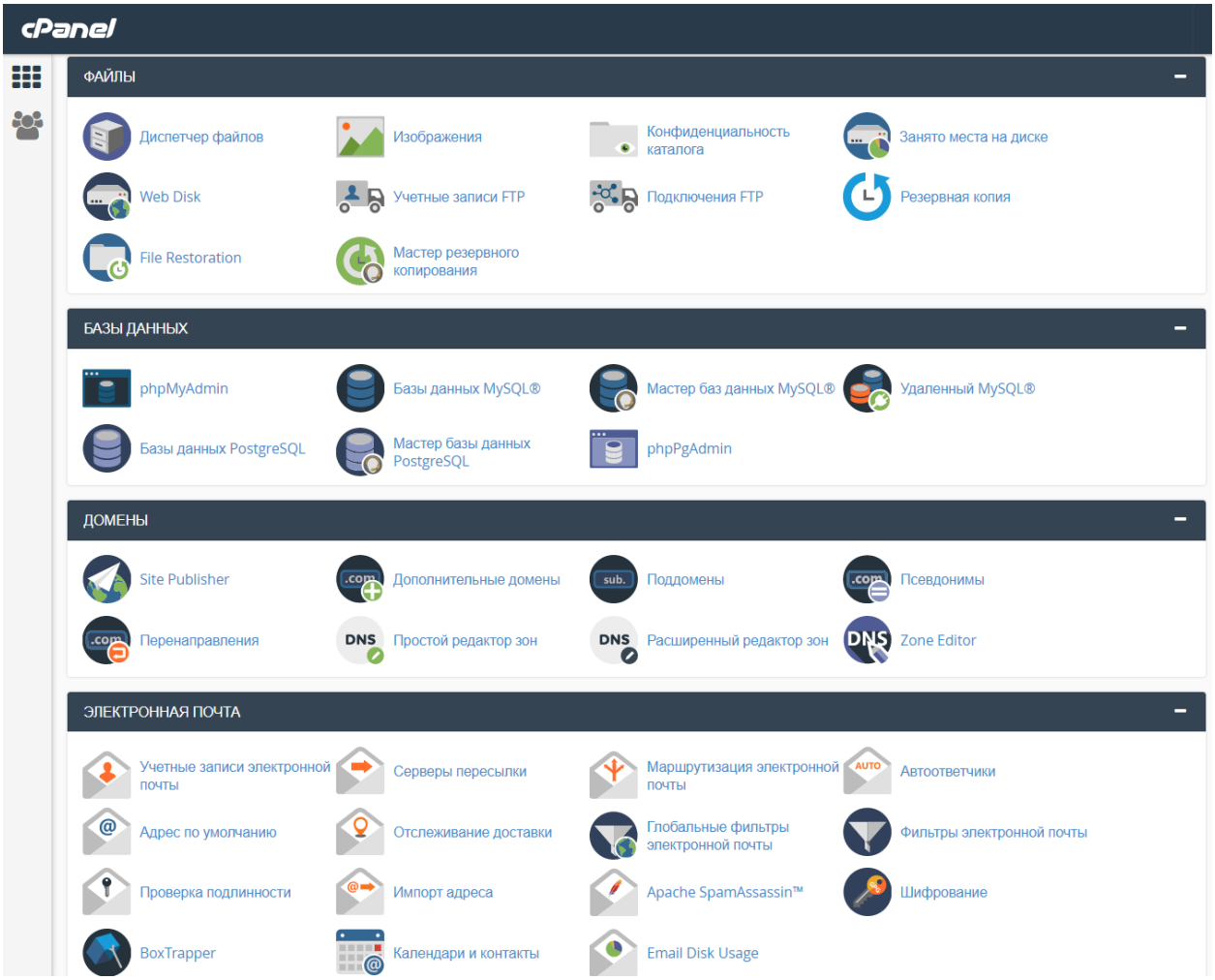


На этом первичный обзор возможностей закончен, теперь перейдём к практике. Создадим БД, две таблицы в ней и свяжем некоторые поля из разных таблиц. Ну, а как соединять БД с программой на C# и работать с таблицами и полями вы уже изучили в первой части…

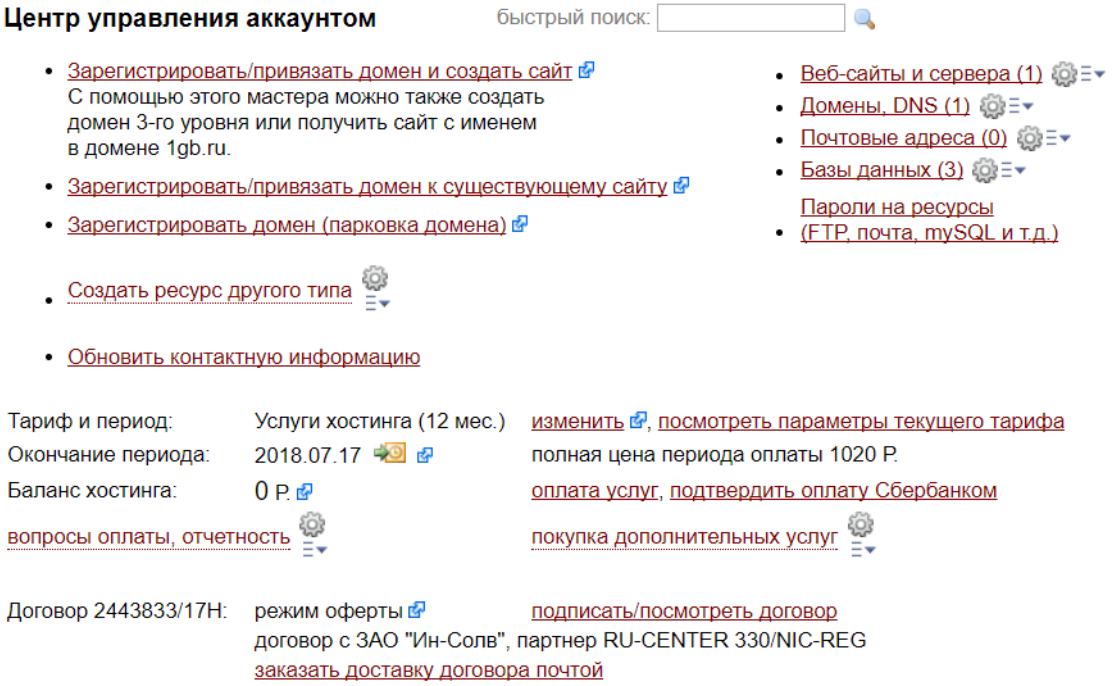
Выбор инструмента

Когда вы зарегистрировались (*про выбор хостинга читайте ниже в «лирическом отступлении»*) на каком-то хостинге с базами данных, то вам предоставляется возможность кроме размещения собственно сайта (html + css + js + php) добавлять базы данных, как правило: MySQL, PostgreeSQL или Microsoft SQL Server.

На разных хостингах интерфейс добавления новой БД устроен по-разному, поэтому описывать что-то конкретное тут не имеет смысла, просто найдите пункт меню или кнопку «Работать с базами данных» или «Добавить базу данных». Весь этот «интерфейс пользователя» обычно называют клиентской панелью – CPanel. В некоторых случаях такая панель может занимать 2-3 экрана браузера и предоставлять очень широкий функционал (тут поместилась только треть функционала):



Не редкость, когда клиентской панели, как таковой, нет вообще:



Лирическое отступление.

Чем меньше наворотов в интерфейсе, тем дешевле вам обойдется содержание вашего хостинга. На непродолжительный срок, например, для тренировок (от 10 дней до месяца) можно найти и бесплатный вариант. Простенькие (платные) варианты от 30 руб./мес. – зависит от срока оплаты (если оплачивать сразу за пару лет, то скидки могут быть существенными). Как бы там ни было, надо иметь ввиду, что то, что дороже, то не всегда будет лучше. Но при покупке недорогого хостинга можно встретиться с ограничением на количество создаваемых баз данных, на объем хранимой информации, на возможность доступа к БД с любого IP. Ограничение по IP означает, что ваша программа на C# будет иметь доступ к БД только с компьютера с таким IP, который вы сами указали в клиентской панели. Иногда это удобно – например, вы хотите, чтобы студенты могли проходить тестирование только из учебной аудитории, а не у себя дома и, конечно, это обеспечивает безопасность БД (никто со стороны не проникнет). Зачастую это создаёт ненужную проблему, так как вы не можете просто так перенести программу в другое место, так как доступ к БД будет прекращён, пока вы лично его не разрешите (введя IP нового места работы). Эту проблему можно преодолеть, купив тариф подороже или просто у другого хостинг-провайдера. Например, на 1GB.ru даже на базовых тарифах нет ограничений ни на количество баз данных ни на IP-адреса, а на shneider-host.ru присутствуют оба эти ограничения на начальном тарифе. Однако интерфейс пользователя, наполненность клиентской панели и отзывчивость службы поддержки на высоте именно на shneider-host.ru.

***Внимание!** Существуют хостинги баз данных без поддержки сайта, то есть БД MySQL вы там можете создать, хранить и работать с ней, а html, css и js там не размещают – попробуйте поискать стоимость услуг и сравнить дополнительные опции таких вариантов. Например, **для тренировок очень удобны сервисы с размещением базы без оплаты** (хоть и на ограниченный период)*

https://www.freesqldatabase.com/ или https://www.freemysqlhosting.net/,

*так как там довольно легко пройти регистрацию, сразу же получить доступ к **phpMyAdmin** и приступить к созданию БД, причем первые 10 дней бесплатно с поддержкой всего функционала, включая и динамический доступ по IP. Впоследствии, вы можете просто оплатить этот сервис для дальнейшей работы на нём или поискать, что-нибудь более функциональное или приемлемое по цене.*

Теперь собственно создание.

Нажмите найденное (там, где что-то про Базы данных) и следуйте, как говорится, инструкциям. На некоторых сервисах всё достаточно удобно, понятно и логично устроено, но так не везде – просто нужно быть внимательнее. Не с первой так со второй попытки всё получится, например, тут всё уж очень аскетично:

Базы данных mySQL / PostgreSQL

имя	пользователь	тип	активна	действия
gb_mytest Москва, РТКомм/СТЕК	gb_mytest	mySQL 5.7.* x64 / I34 / aux1 / NEW_PASS	да	удалить новый пароль управлять базой
gb_psis Москва, РТКомм/СТЕК	gb_psis	mySQL 5.7.* x64 / I34 / aux2 / NEW_PASS	да	удалить новый пароль управлять базой
gb_x_pcoding Москва, РТКомм/СТЕК	gb_x_pcoding	mySQL 5.7.* x64 / I34 / aux1 / NEW_PASS	да	удалить новый пароль управлять базой

изменить все mySQL пароли

создать базу для сайта: pcoding-ru

имя базы данных: newdb 4-10 символов, английский алфавит и цифры

тип mySQL / pgSQL: mySQL 5.7 (InnoDB)

mySQL 3.23
mySQL 4.1
mySQL 5.0 (InnoDB)
mySQL 5.5 (InnoDB)
mySQL 5.7 (InnoDB)
Postgres 8.3 latin1
Postgres 8.3 ru_RU.cp1251
Postgres 8.3 ru_RU.utf8
Postgres 8.4 latin1

- CP1251
те опцию; ваши скрипты выберут кодировку сами
ивают old_passwords
совместимость базы с различным ПО, включайте только если вы не можете
ясно видите ошибку про old_passwords

Когда новая БД будет создана, то вы должны себе скопировать адрес сервера для её хранения, имя БД, логин пользователя и пароль. Напомню, как мы настраивали свою программу на подключение к БД в первой части:

```
db.Server = "mysql95.1gb.ru";   // хостинг БД
db.Database = "gb_psis";       // Имя БД
db.UserID = "gb_psis";         // Имя пользователя БД
db.Password = "ca8484adc89a";   // Пароль пользователя БД
```

Все параметры, кроме адреса сервера хостинга, вы можете настраивать сами во время создания новой БД. Отмечу, что, для корректной работы с кириллицей, важно правильно настроить кодировку – обратите внимание, что для корректной работы с C# в phpMyAdmin следует выбирать кодировку utf8_unicode_ci:

mysql95.1gb.ru » gb_psis » students

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Операции | Три

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
1	id_stud	int(11)			Нет	Нет	AUTO_INCREMENT	
2	name1	varchar(20)	utf8_unicode_ci		Нет	Нет		
3	name2	varchar(20)	utf8_unicode_ci		Нет	Нет		
4	name3	varchar(20)	utf8_unicode_ci		Нет	Нет		
5	group_num	int(11)			Нет	Нет		

Отметить все / Снять выделение *С отмеченными:*

Версия для печати | Связи | Анализ структуры таблицы

Добавить 1 поле(я) ◉ В конец таблицы ○ В начало таблицы ○ После id_stud OK

+ Индексы

Информация

Используемое пространство	
Данные	16 КБ
Индекс	32 КБ
Всего	48 КБ

Статистика строк	
Формат	динамический
Сравнение	utf8_unicode_ci
Следующий автоматический индекс	14
Создание	Окт 28 2017 г., 07:38
Последнее обновление	Ноя 10 2017 г., 17:48

Причём выбор кодировки возможен как для всей БД, так и для отдельный полей индивидуально (в том числе и после создания):

mysql95.1gb.ru » gb_psis » students

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Операции | Триггеры

Изменить

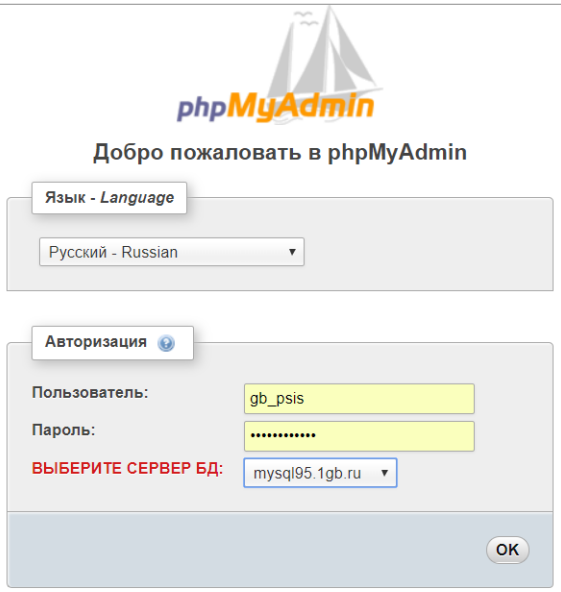
Структура

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты
name1	VARCHAR	20	Нет	utf8_unicode_c	

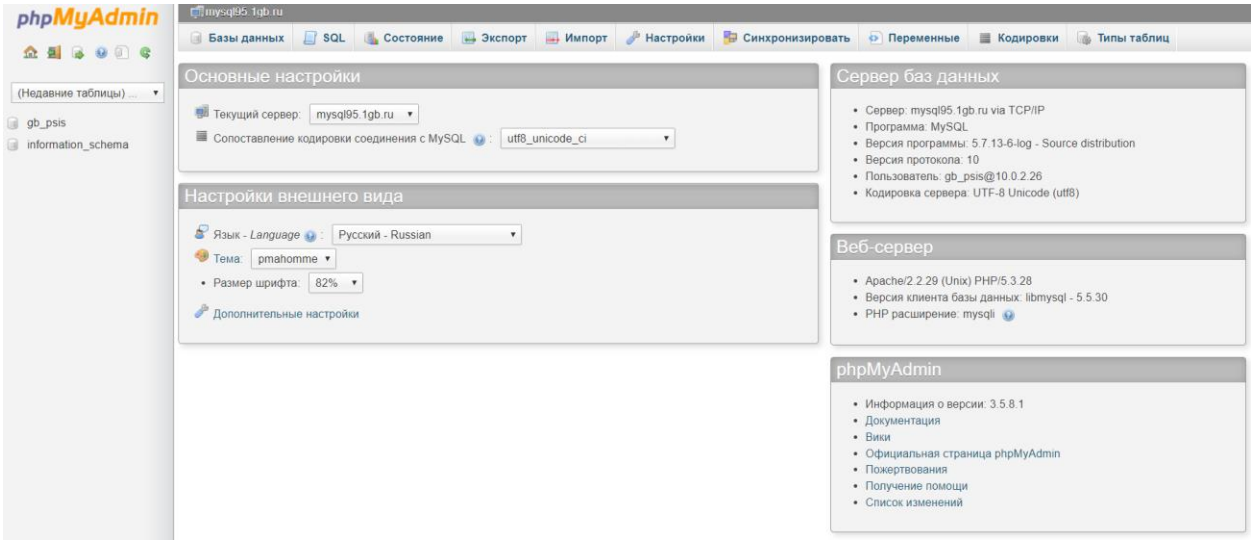
utf8_general_ci
utf8_general_mysql500_ci
utf8_german2_ci
utf8_hungarian_ci
utf8_icelandic_ci
utf8_latvian_ci
utf8_lithuanian_ci
utf8_persian_ci
utf8_polish_ci
utf8_roman_ci
utf8_romanian_ci
utf8_sinhala_ci
utf8_slovak_ci
utf8_slovenian_ci
utf8_spanish2_ci
utf8_spanish_ci
utf8_swedish_ci
utf8_turkish_ci
utf8_unicode_520_ci
utf8_unicode_ci

В принципе, сам инструмент phpMyAdmin можно скачать (`https://www.phpmyadmin.net/`) и установить на своём компьютере и работать локально (без подключения к сети).

Итак, вы зарегистрировались на хостинге, создали там базу данных MySQL, и собираетесь работать с данными. Вам будет предоставлена ссылка для входа в phpMyAdmin, перейдя по ней, вы встретитесь с окном авторизации:



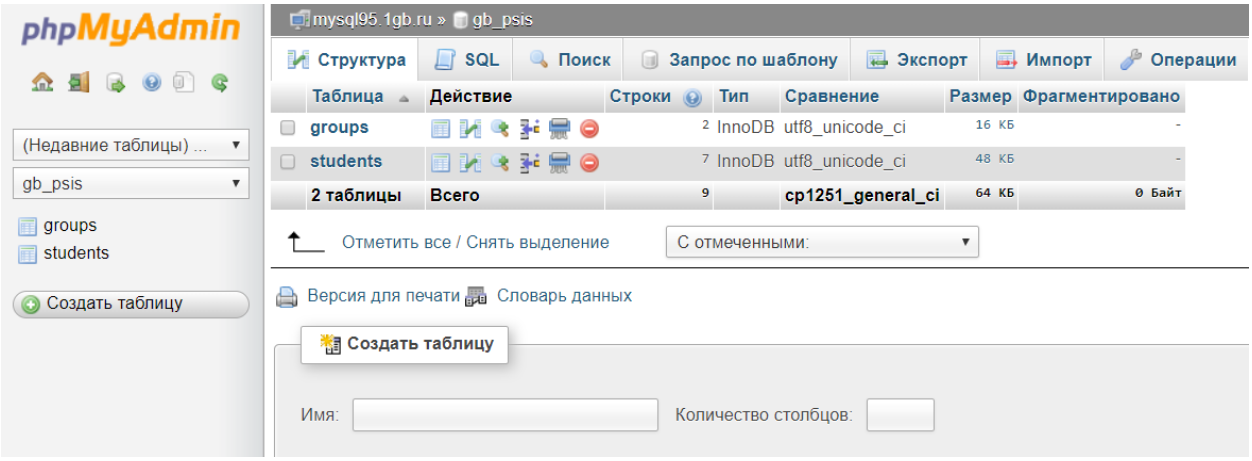
Пройдите авторизацию с данными установленными ранее, и вы перейдёте к окну редактирования вашей базы, где уже можно будет добавлять таблицы, редактировать их содержимое, связывать поля и строить запросы:



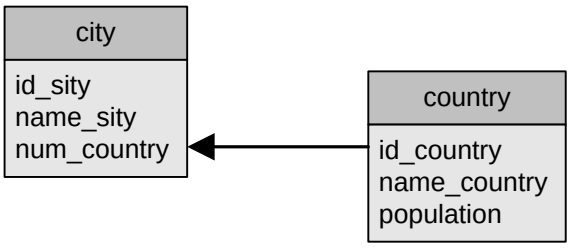
Не забудьте сразу установить кодировку, соответствующую кодировке C#.

При первом запуске в левой колонке вы увидите доступ к виртуальной базе information_shema (которая содержит метаданные – информацию о других базах

данных, которые поддерживает сервер MySQL – это несколько таблиц только для чтения, более подромно можно почитать тут: `http://www.rldp.ru/mysql/mysqlpro/schema.htm`) и к созданной вами, кликните по ней и приступите к созданию таблиц (кнопка слева «Создать таблицу»):



В моём случае уже видны две таблицы, так как я структуру базы данных создавал ранее. Давайте создадим две новые таблицы «Страны» (с полями идентификатор, название страны, население страны) и «Города» (с полями идентификатор, название города, ссылка на название страны) и свяжем их по полям:



Создадим новую таблицу:

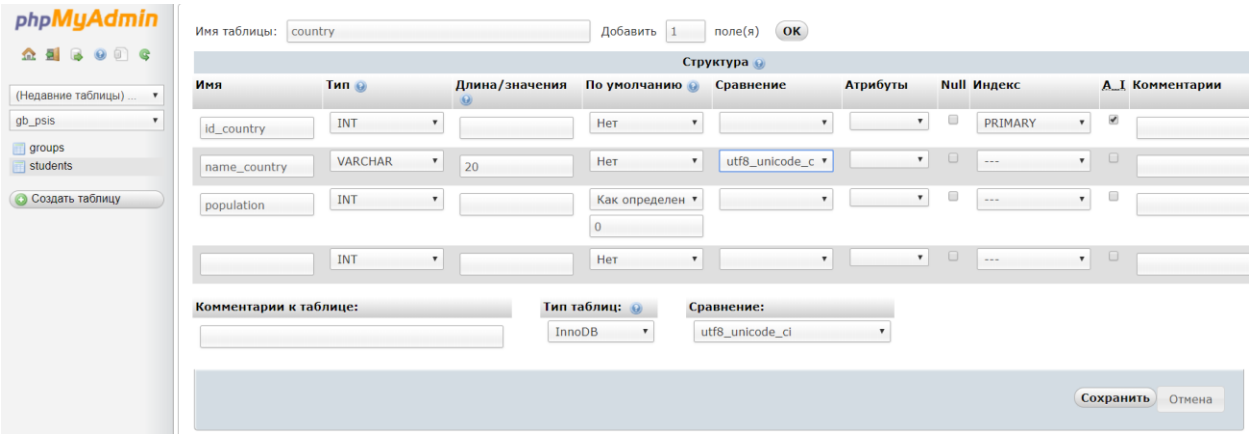


Таблица должна быть типа InnoDB, чтобы была возможность поддерживать связи между таблицами базы данных. Обратите также внимание, что поле id_country следует установить как Primary и добавить параметр AUTO_INCREMENT (чтобы индекс сам увеличивался).

Возможные настройки параметра Индекс:

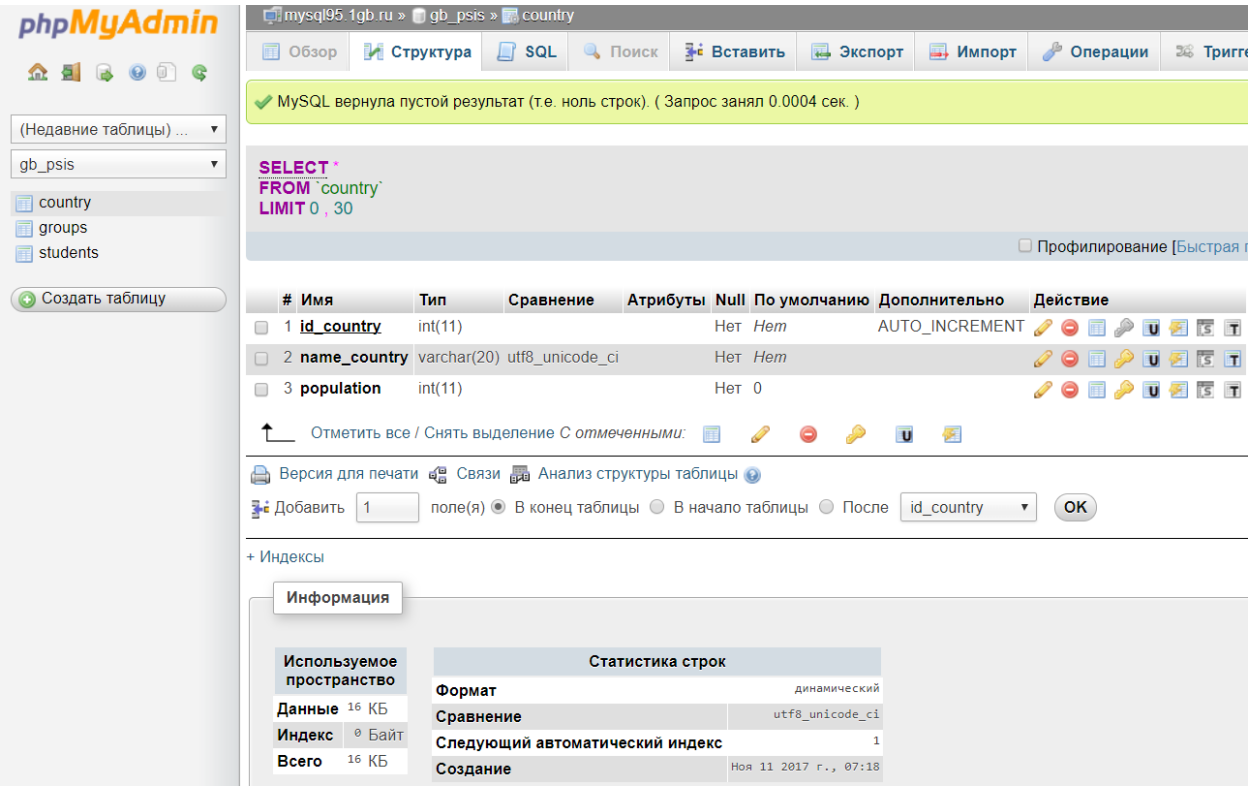
primary – первичный ключ;

unique – уникальный идентификатор (допускает NULL-значения);

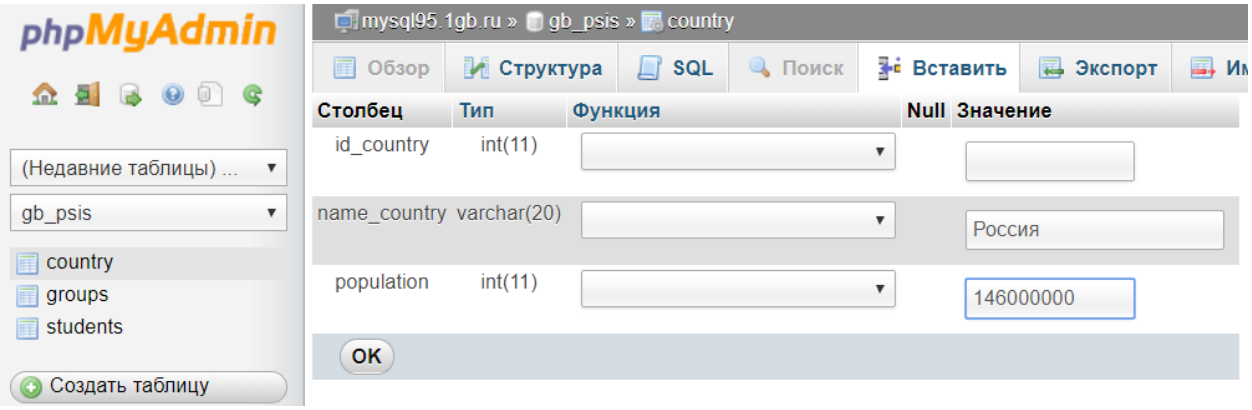
fulltext – полнотекстовой индекс (поиск по всему тексту);

index – простой индекс.

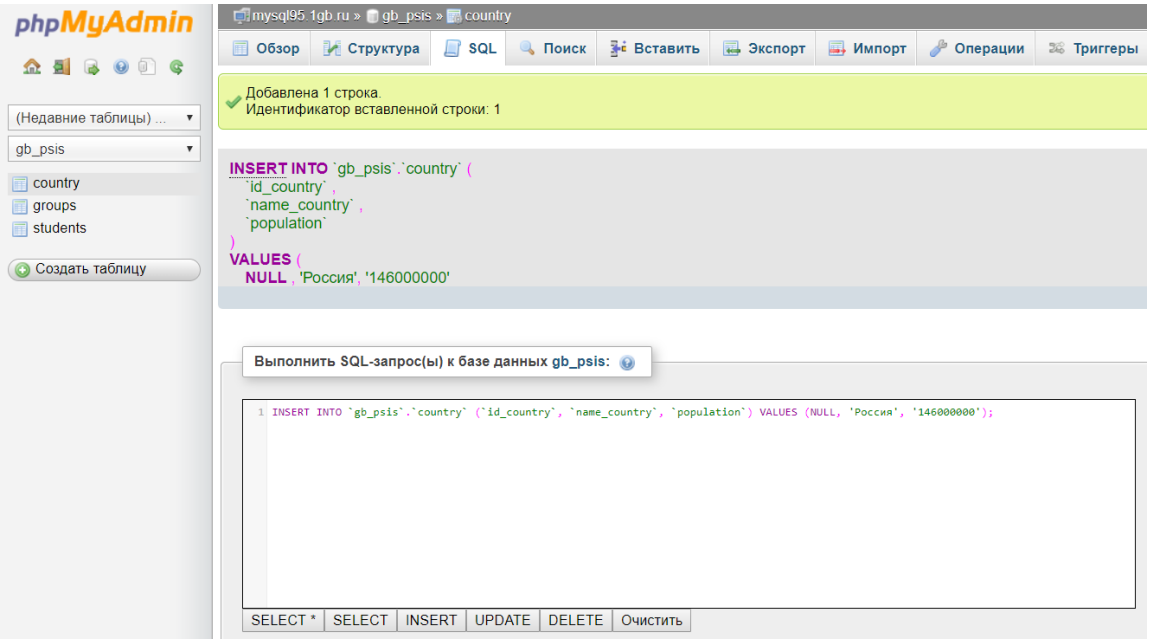
Сохраните таблицу (клавиша «Сохранить») и снова кликните в левом столбце по ней – откроется режим просмотра структуры таблицы:



Поля в таблице можно удалять, добавлять и изменять их настройки. Если вас что-то не устраивает в настройках полей, воспользуйтесь соответствующими клавишами в правой части таблицы. Если вы хотите добавить поля к таблице, то выберите соответствующее действие непосредственно под таблицей. Если вы хотите добавить данные вручную, то нажмите на клавишу «Вставить» в верхнем меню:

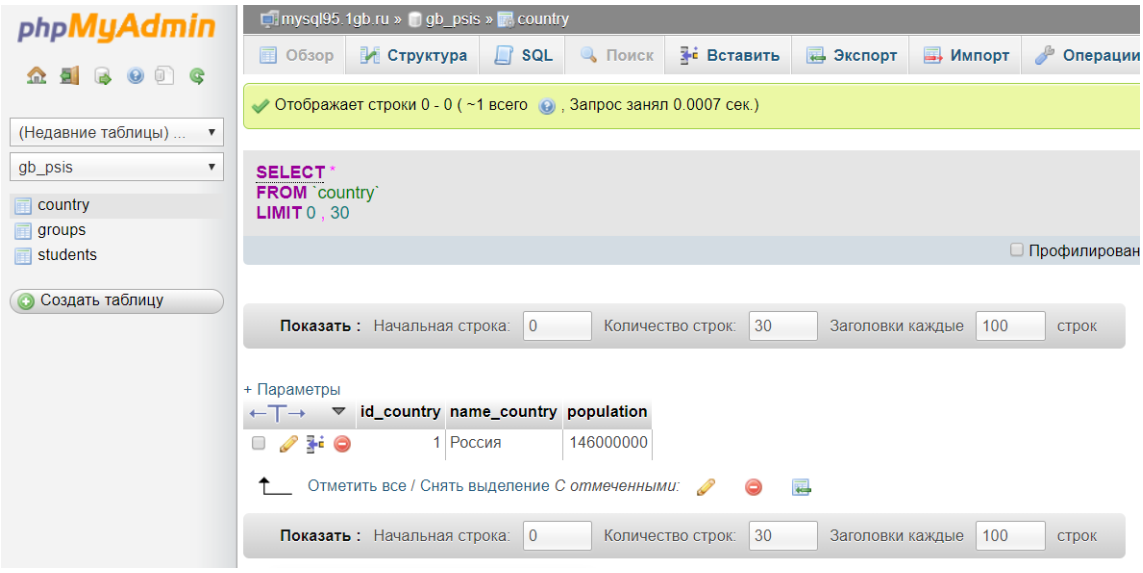


Наберите необходимые данные (обратите внимание, что номер id_country мы не пишем сами, так как поле настроено так, что будет самостоятельно увеличиваться) и нажмите OK для добавления записи. В таблицу будет добавлена одна строка и вам будет показан выполненный запрос:

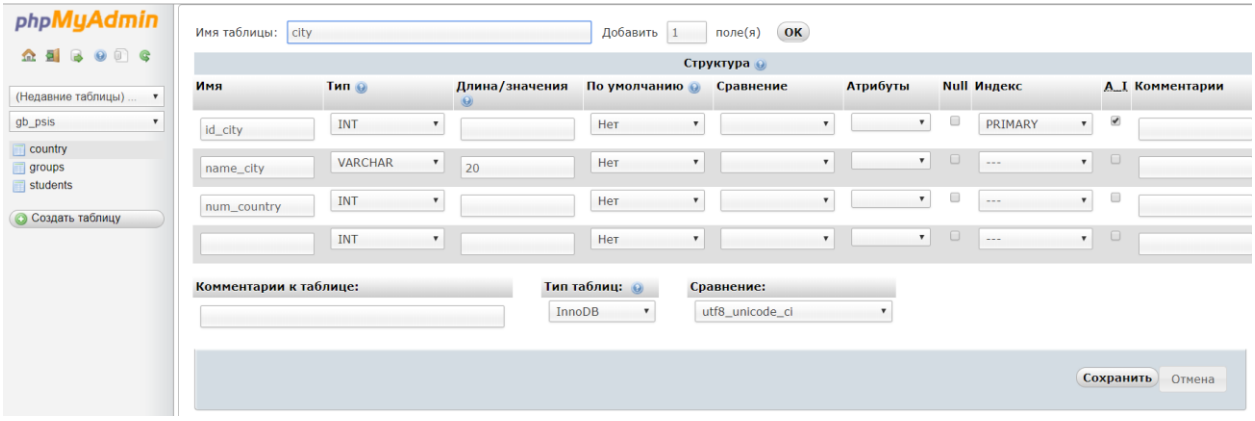


На данной вкладке (SQL) вы можете самостоятельно создавать и запускать на исполнение запросы к вашей БД. Это может пригодиться, когда вы будете писать программу на C# – прежде чем добавлять запрос в код и испытывать его в рамках программы, можно его апробировать непосредственно в панели phpMyAdmin.

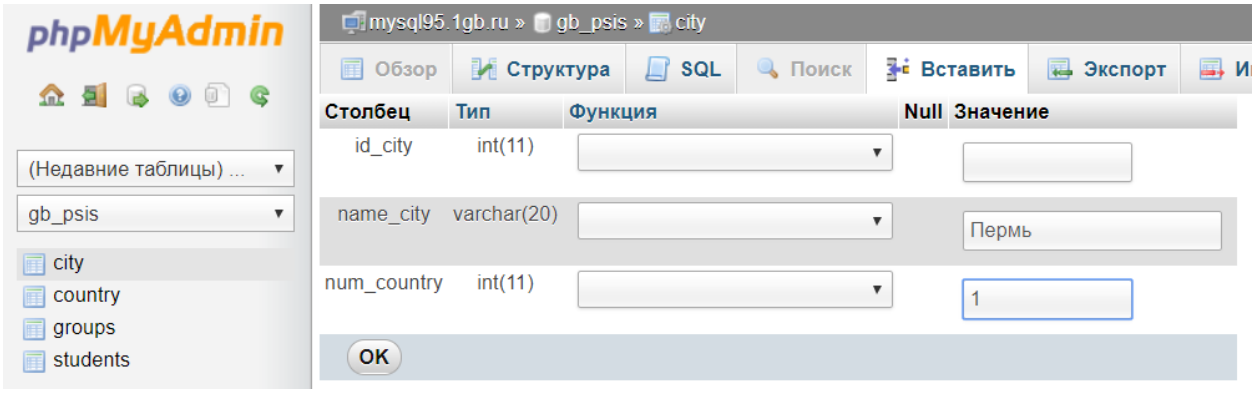
После добавления данных становится активной клавиша «Обзор», перейдите по ней и вы сможете просмотреть содержимое таблицы и, при необходимости, изменить данные:



Аналогичным образом создадим и вторую таблицу:

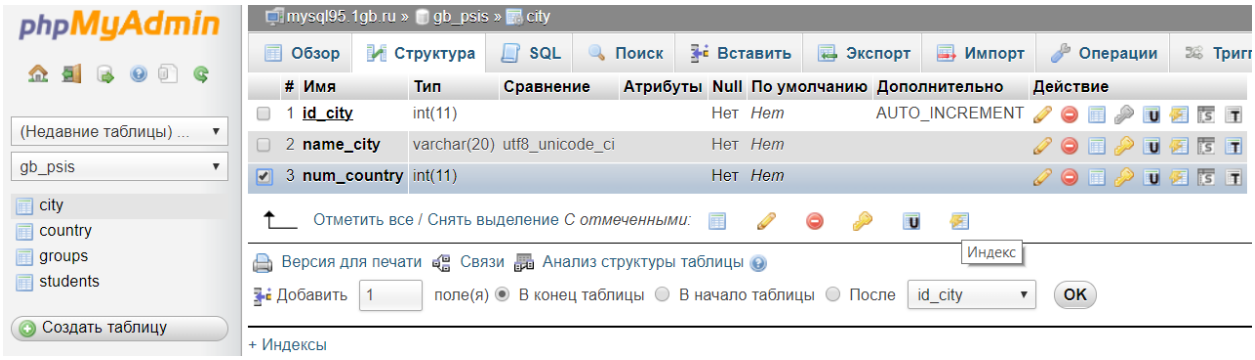


И добавим в неё одну запись:

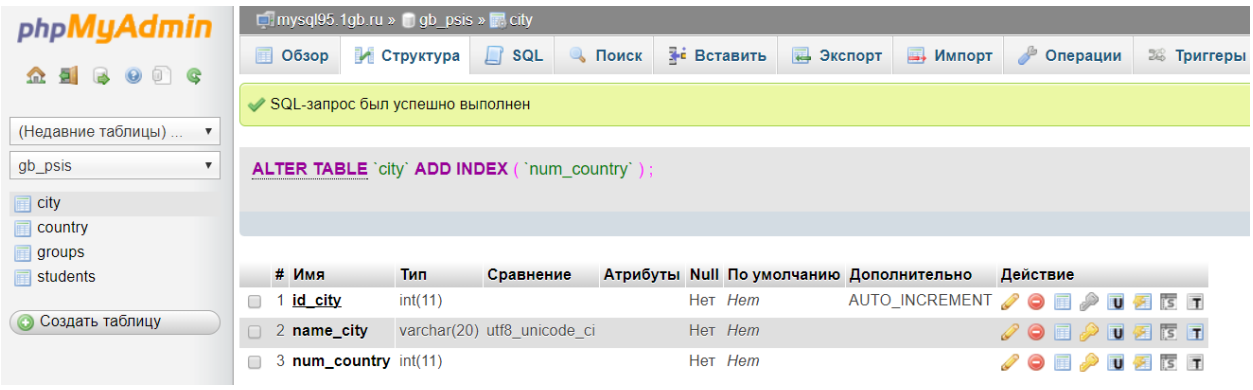


Обратите внимание, что пока таблицы не связаны, можно вручную проставить номер num_country (это ссылка на запись про страну).

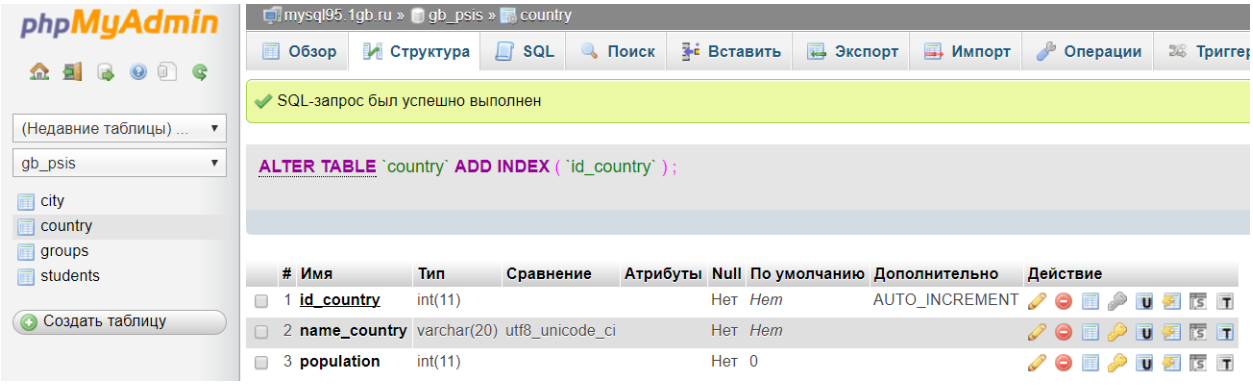
Так как уже есть две таблицы, то между ними можно создать связь, для чего необходимо сначала добавить в индекс связываемые поля таблиц. Выберите пункт «Структура», выделите необходимое для индексирования поле (num_country) и нажмите на клавишу «Индекс» в нижней части справа:



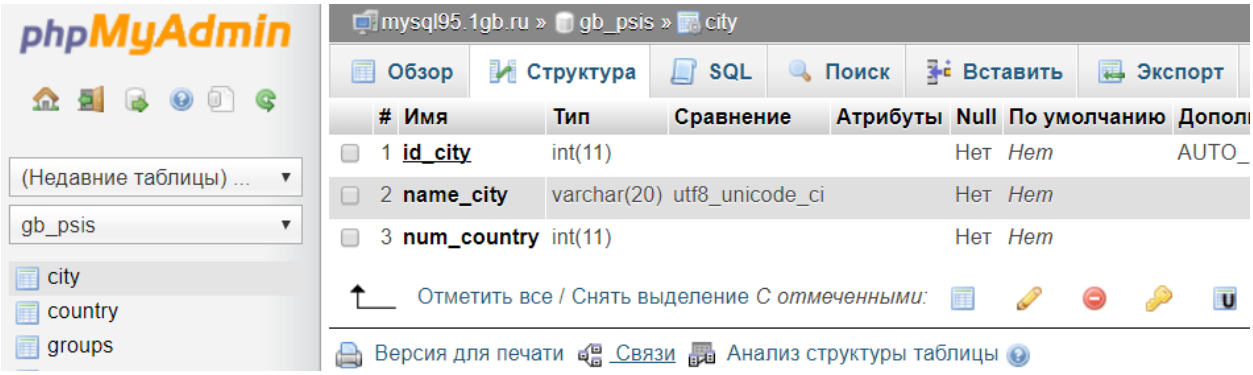
В ответ выполнится запрос, который, кстати, можно было бы и вручную набрать во вкладке SQL и самостоятельно запустить на исполнение:



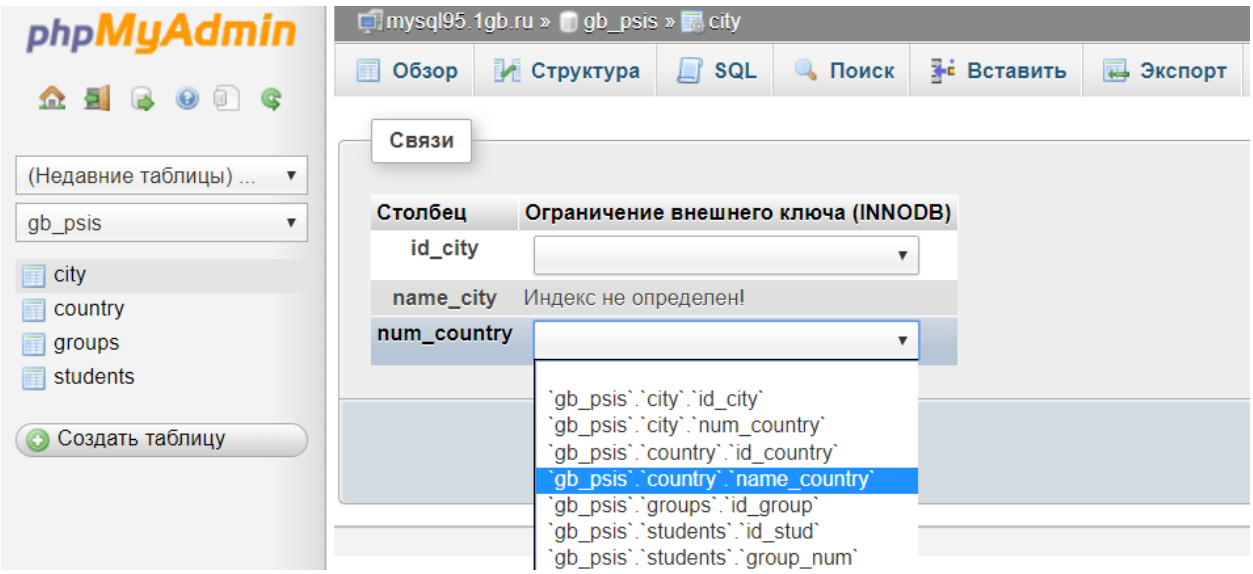
Аналогичные действия проделайте для поля id_country:



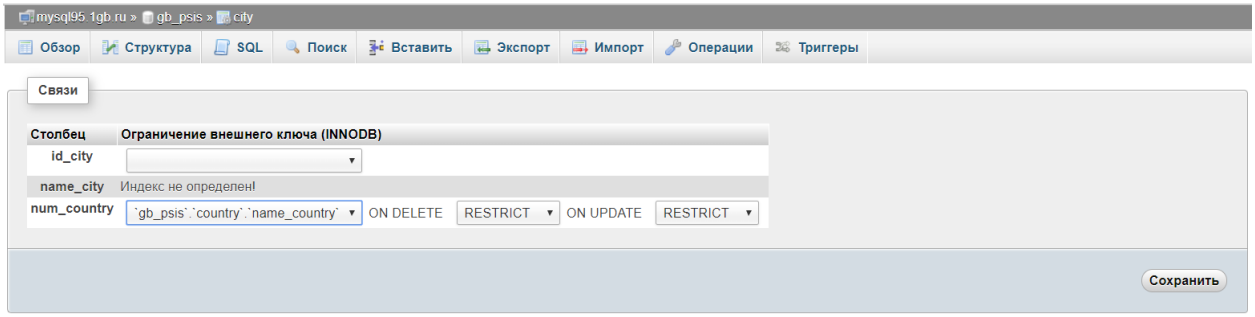
После назначения индексируемых полей можно установить между ними связь, для чего выбираем таблицу city, выбираем режим структуры и кликаем на клавишу «Связи» в нижней части:



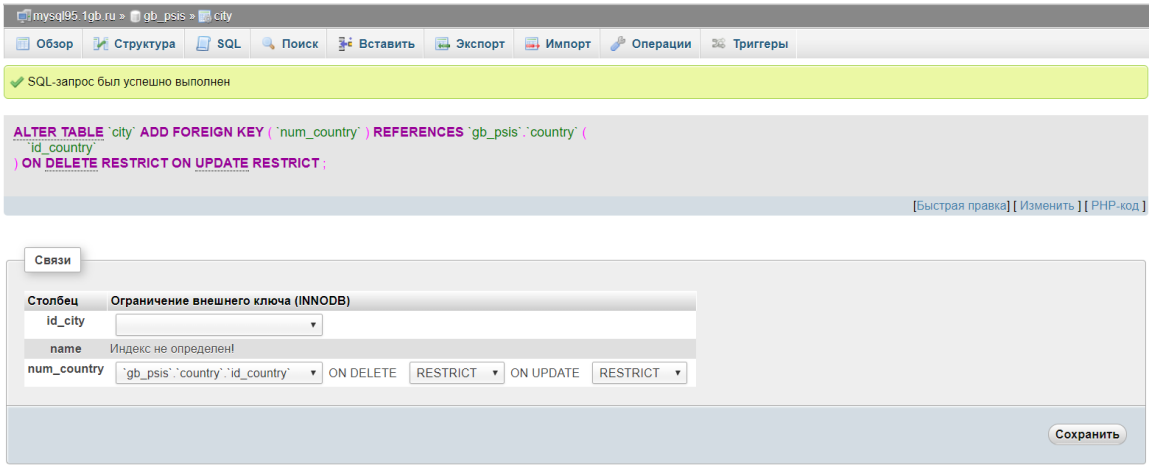
В следующем окне выбираем соединение поля num_country с полем name_country:



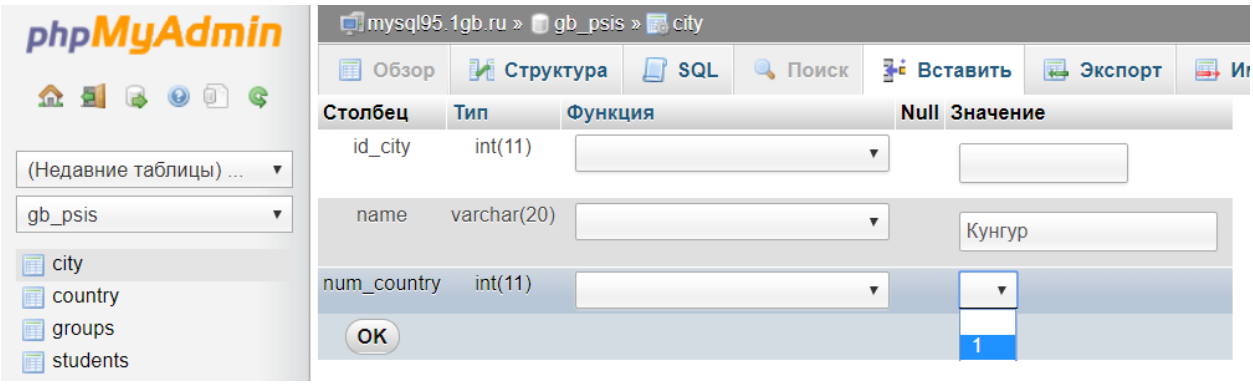
В следующем окне определите дополнительные действия для таблицы:



Выражения ON DELETE и ON UPDATE внешних ключей используются для указания действий, которые будут выполняться при удалении строк родительской таблицы (ON DELETE) или изменении родительского ключа (ON UPDATE). В данном случае приложению будет запрещено удалять или изменять родительский ключ, если существуют ссылающиеся на него дочерние ключи. Нажмите сохранить и получите следующий ответ:



После установки связей в поле num_country уже нельзя будет вручную заносить данные, их нужно будет выбирать из выпадающего списка, формирующегося из поля id_country, на которое мы и установи ссылку:



Итак, таблицы созданы, связи между соответствующими полями таблиц налажены, теперь уже можно приступать к разработке приложения по работе с базой данных. Основываясь на опыте, полученном в первой части, попробуйте самостоятельно сделать запросы по добавлению и удалению записей, по работе со связными таблицами, по изменению записей.

Прежде чем писать запросы в коде программы можно провести тесты на самой БД в панели phpMyAdmin. Предположим у меня вот такое содержимое таблиц:

id_country	name_country	population
1	Россия	146000000

id_city	name_sity	num_country
1	Пермь	1
2	Кунгур	1

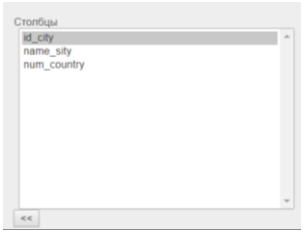
Выделите таблицу city и перейдите во вкладку SQL, там будет стоять запрос по умолчанию на выборку всего содержимого из таблицы с ограничением «первые 30»:

```
SELECT *
FROM `city`
WHERE 1
LIMIT 0 , 30
```

Давайте доработаем его и выведем только отсортированные по алфавиту названия городов:

```
SELECT `name_sity` FROM `city` ORDER BY `name_sity`
```

Для того чтобы не писать названия полей вручную, пользуйтесь таблицей «Столбцы» в правой части вкладки SQL (выделить столбец и нажать клавишу переноса «<<»):



Ряд клавиш под полем редактирования самого запроса предоставляют нам шаблоны написания запросов, для пробы нажмите на клавишу INSERT:

Вам будет предоставлен шаблон на добавление записи:

```
INSERT INTO `city`(`id_city`, `name_sity`, `num_country`)
VALUES ([value-1], [value-2], [value-3])
```

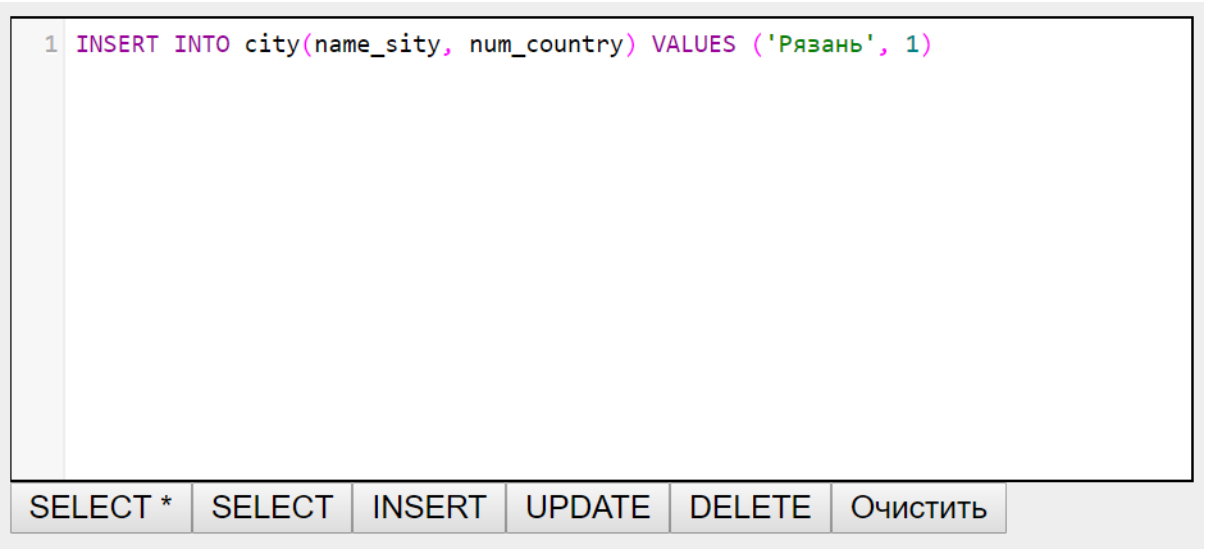
где поля value вы можете заполнить сами и запустить уже готовый запрос на исполнение. Напомню, что поле id_city у нас с автоинкрементом, поэтому туда номер не следует писать вручную (он будет добавляться атвоматически), а шаблон можно исправить так:

```
INSERT INTO `city`(`id_city`, `name_sity`, `num_country`)
VALUES (null, 'Москва', 1)
```

или так:

```
INSERT INTO `city`(`name_sity`, `num_country`) VALUES ('Ижевск', 1)
```

Попробуйте работу запросов и убедитесь в изменении записей в таблицах. Следует отметить, что базы данных чувствительны к использованию правильных символов в качестве апострофов. Для примера приведу пару скриншотов, которые демонстрируют использование корректных символов:



```
1 DELETE FROM `city` WHERE `name_city` = 'Рязань'
```

SELECT * | SELECT | INSERT | UPDATE | DELETE | Очистить

Ну, и, наконец, можно испытать запрос, адресованный на извлечение данных из связанных таблиц. Пусть нужно вывести данные о названиях городов и их странах. Если мы оформим простой запрос:

```
SELECT name_sity, num_country FROM city
```

то получим неудовлетворительный результат:

name_sity	num_country
Пермь	1
Кунгур	1
Москва	1
Ижевск	1
Рязань	1
Лондон	3

Очевидно, что вместо порядковых номеров в последнем столбце хотелось бы видеть названия самих стран. Исправьте SQL-запрос так:

```
SELECT city.name_sity, country.name_country FROM country
INNER JOIN city ON country.id_country = city.num_country
```

и получите приемлемый вариант:

name_sity	name_country
Пермь	Россия
Кунгур	Россия
Москва	Россия
Ижевск	Россия
Рязань	Россия
Лондон	Великобритания

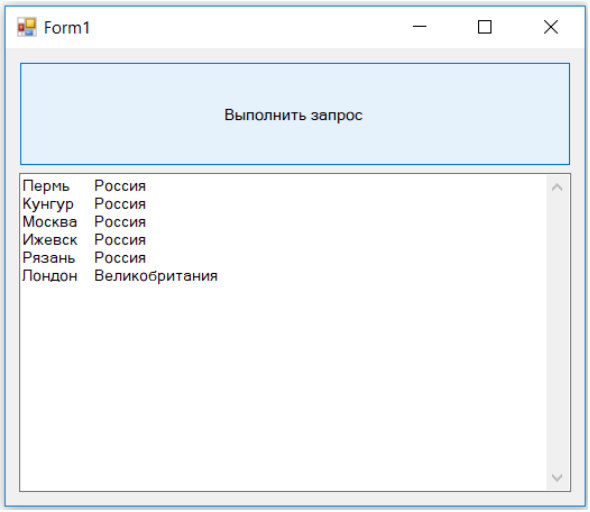
Если вы зарегистрировали для себя хостинг баз данных MySQL и смогли подготовить базу данных с указанными таблицами, то, после изучения способов работы средствами C# с базой данных, вы смогли бы подготовить следующее небольшое приложение по работе с этой базой данных и с запросом к связанным таблицам:

```
using System;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace MySQLQuery
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        MySqlConnection conn;

        private void Form1_Load(object sender, EventArgs e)
        {
            MySqlConnectionStringBuilder db;
            db = new MySqlConnectionStringBuilder();
            db.Server = "mysql95.1gb.ru";   // хостинг БД
            db.Database = "gb_psis";      // Имя БД
            db.UserID = "gb_psis";        // Имя пользователя БД
            db.Password = "ca8484adc89a";   // Пароль пользователя БД
            db.CharacterSet = "utf8";       // Кодировка Базы Данных

            conn = new MySqlConnection(db.ConnectionString);
            try
            {
                conn.Open();
                MessageBox.Show("Подключение к БД установлено");
                btnSQL.Enabled = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Проблемы с подключением к БД \n\r" + ex.ToString());
            }
        }

        private void btnSQL_Click(object sender, EventArgs e)
        {
            string sql;
            sql = "SELECT city.name_sity, country.name_country FROM country ";
            sql += "INNER JOIN city ON country.id_country = city.num_country";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            textBox1.Clear();
            while (reader.Read())
            {
                textBox1.Text += reader[0].ToString() + '\t' +
                                 reader[1].ToString() + Environment.NewLine;
            }
            reader.Close();
        }
    }
}
```

Данные, приведённые в строке соединения действующие, специально для проведения испытаний обучающимися. Не злоупотребляйте этим, не пишите вредоносных программ и не удаляйте таблицы и записи в них – это нужно для последующих поколений студентов…

И не забывайте про подключение библиотеки для работы с БД MySQL (`MySql.Data.MySqlClient`).

Дизайн программы и результаты её работы таковы:



В программе всего две процедуры:

– `Form1_Load` – обеспечивает автоматическое подключение к БД с выдачей информационного сообщения о результатах подключения;

– `btnSQL_Click` – собственно реализует SQL-запрос с выдачей результатов в многострочное текстовое поле.

Этого материала вполне достаточно, чтобы понимать порядок разработки приложения по работе с сетевой базой данных, осознавать встречающиеся затруднения и искать способы их преодоления самостоятельно в сети интернет.

Часть 3. Некоторые примеры структуры базы данных и запросов к ней

Предположим в парикмахерской для учёта работ, производимых мастерами, ведётся БД, в которой есть три таблицы («Клиенты» – Clients, «Мастера» – Masters, «Работы» – Services)– базовая таблица Clients с индексами:

+ Параметры

id_client	name_client	num_service	num_master
1	Иванов	3	3
2	Петров	3	4
3	Сидоров	4	1

и две таблицы, где хранится информация о содержимом этих индексов:

services
students
Создать таблицу

Показать : Начальная строка: 0
Сортировать по индексу: Нет
+ Параметры

id_service	name_service
3	стрижка
4	покраска

masters
services
students
Создать таблицу

Показать : Начальная строка: 0
Сортировать по индексу: Нет
+ Параметры

id_master	name_master
1	МастерА
2	МастерБ
3	МастерВ
4	МастерГ

Обсудим, как правильно связать эти таблицы и, какой нужно сделать запрос, чтобы результаты выводились в приемлемом для человека виде, например, так:

```
Иванов МастерВ стрижка
Петров МастерГ стрижка
Сидоров     МастерА покраска
```

Начнём с назначения индексных полей. В базовой таблице это сразу два поля – num_service и num_master, так как именно там будут храниться ссылки, поэтому в phpMyAdmin выелите таблицу Clients, перейдите на вкладку Структура, выделите «галочку» для данных столбцов и нажмите клавишу «Индекс» в правом нижнем углу:

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
1	id_client	int(11)			Нет	*Нет*	AUTO_INCREMENT	
2	name_client	varchar(20)	utf32_unicode_ci		Нет	*Нет*		
3	num_service	int(11)			Нет	*Нет*		
4	num_master	int(11)			Нет	*Нет*		

Отметить все / Снять выделение *С отмеченными:* Индекс

Версия для печати Связи Анализ структуры таблицы

Аналогичные действия осуществите и с оставшимися двумя таблицами – Services для поля id_services:

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
1	id_service	int(11)			Нет	*Нет*	AUTO_INCREMENT	
2	name_service	int(11)			Нет	*Нет*		

Отметить все / Снять выделение *С отмеченными:*

Версия для печати Связи Анализ структуры таблицы Индекс

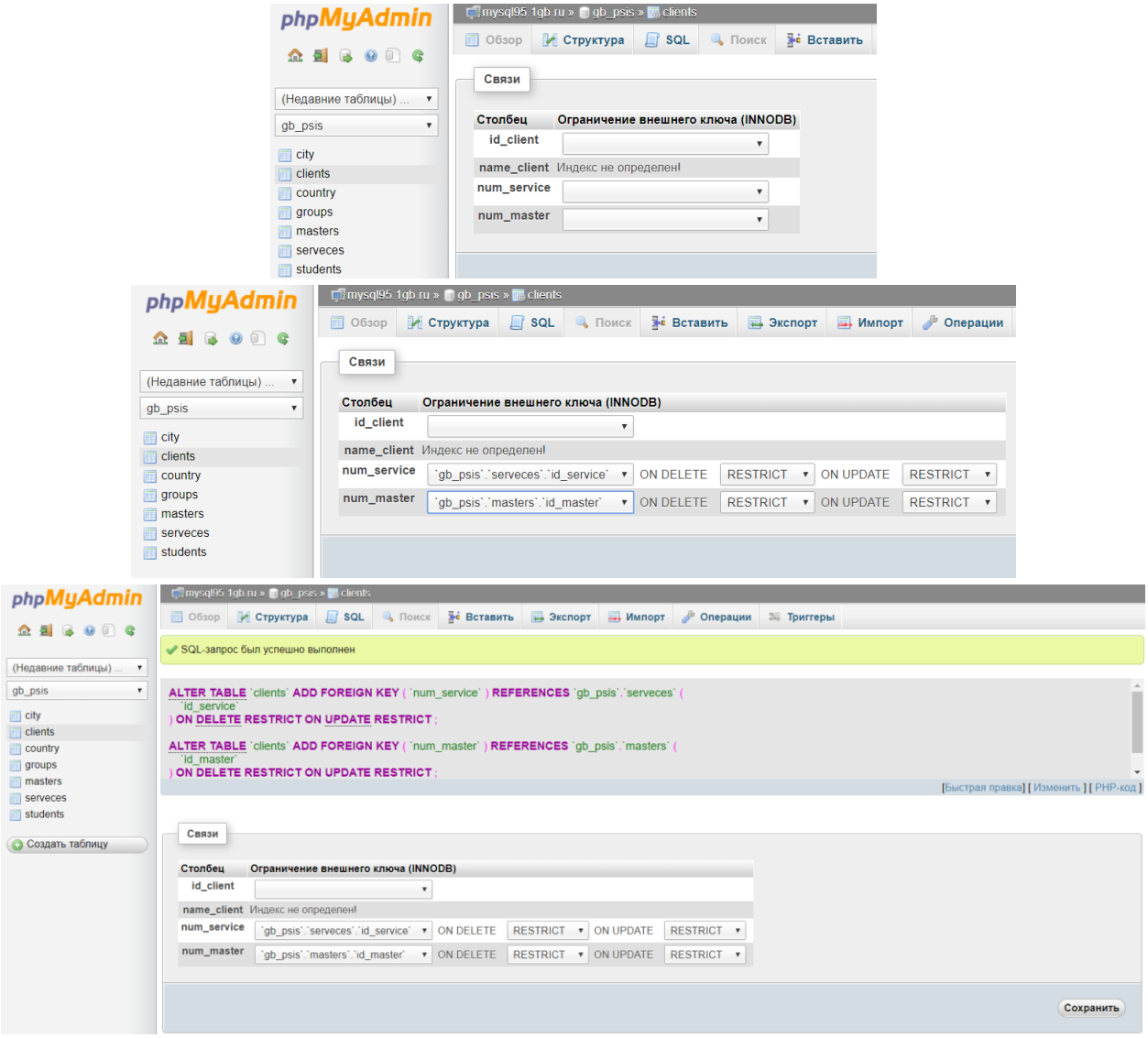
и Masters для поля id_masters:

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
1	id_master	int(11)			Нет	*Нет*	AUTO_INCREMENT	
2	name_master	varchar(20)	cp1251_general_ci		Нет	*Нет*		

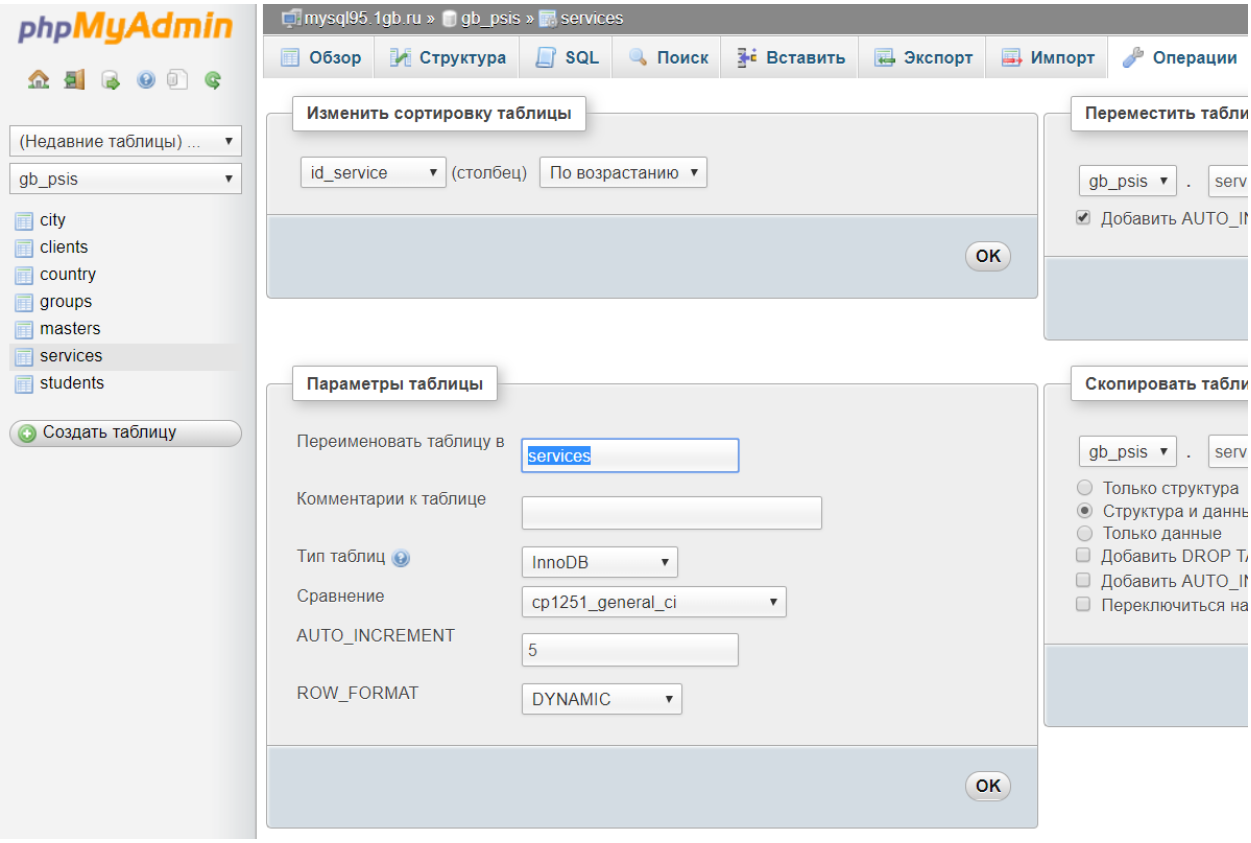
Отметить все / Снять выделение *С отмеченными:*

Версия для печати Связи Анализ структуры таблицы Индекс

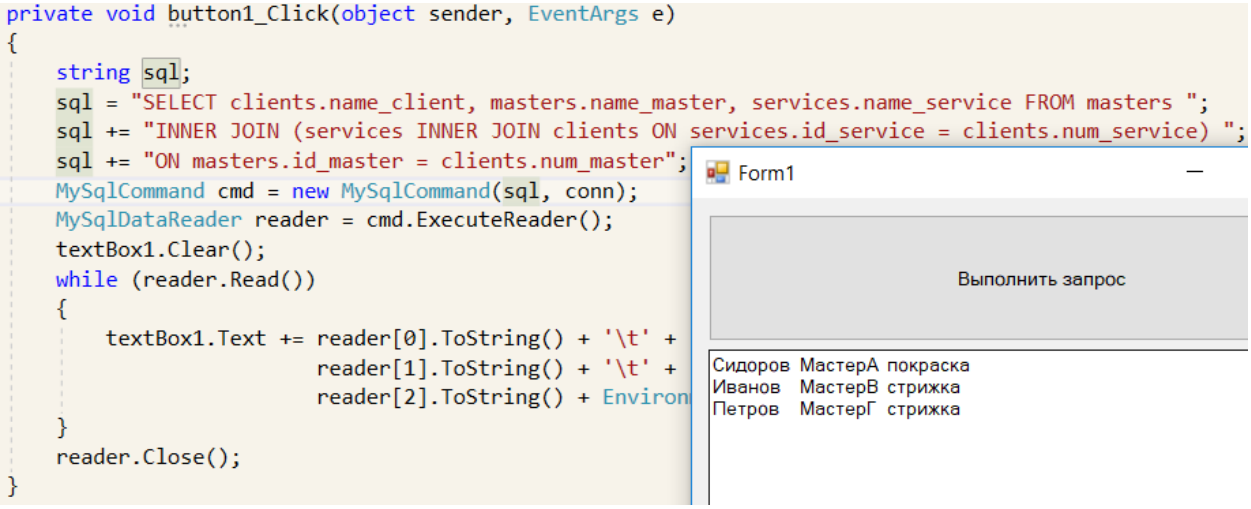
Далее выделяем таблицу Clients, нажимаем в нижней части «Связи», указываем индексные столбцы, назначаем связи и нажимаем «Сохранить»:



Обратите внимание, что при наборе названия таблицы Services я допустил ошибку – SerEces. В этом нет никакой опасности для дальнейшей работы, при переименовании таблиц связи остаются. Для переименования таблицы – выделите таблицу, перейдите на вкладку «Операции», в разделе «Параметры таблицы» внесите новой имя и нажмите «OK»:



Ну и наконец, рассмотрим сложный запрос с заменой индексов настоящими названиями:



Обратите внимание, что в таком запросе приходится дважды применять оператор INNER JOIN, который возвращает пересечение двух множеств по указанным критериям.

Приложение 1

Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пермский государственный аграрно-технологический университет
имени академика Д.Н. Прянишникова»

Кафедра Информационных технологий
и программной инженерии

КУРСОВОЙ ПРОЕКТ

по дисциплине: Программная инженерия
объектно-ориентированного программирования

на тему: Разработка клиент-серверного приложения.

Выполнил:
студент __-го курса __очного отделения
специальности 09.03.04 Программная инженерия
шифр П___-___-_____

_____

Проверил:
доцент кафедры ИТиПИ, к.т.н., доцент
Беляков Андрей Юрьевич

Пермь – 20__

Приложение 2

РЕЦЕНЗИЯ

на курсовой проект _____

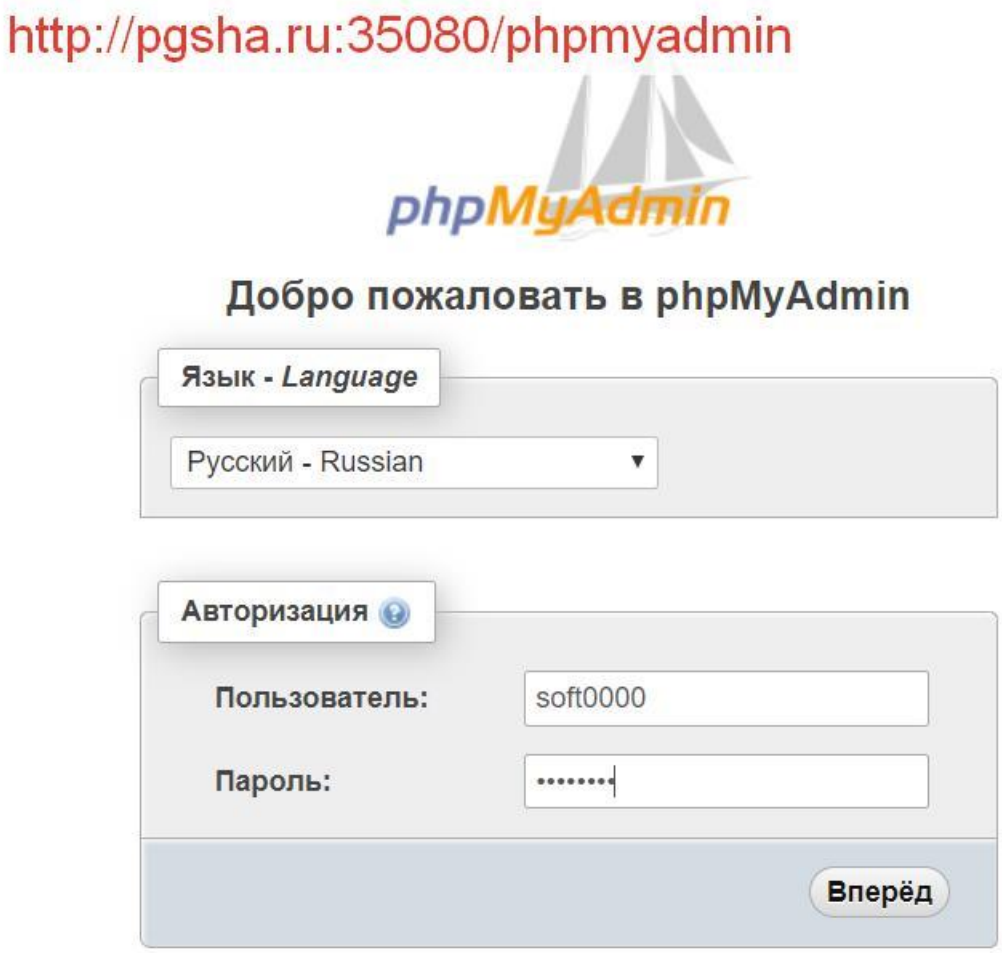
доцент кафедры ИТиПИ, к.т.н., доцент
Беляков Андрей Юрьевич

Порядок работы с хостингом университета

Часть 1. Как подключаться к БД через панель phpMyAdmin.
Часть 2. Как подключаться к БД через редактор MySQL WorkBench.
Часть 3. Как наладить FK для связи таблиц в phpMyAdmin.
Часть 4. Как экспортировать БД из phpMyAdmin и добавить в WorkBench.

1. Как подключаться к БД через панель phpMyAdmin.

Для работы с таблицами базы данных вы можете через браузер обратиться по адресу – http://pgsha.ru:35080/phpmyadmin



В поля пользователь и пароль введите значения, которые вам выдал преподаватель, нажмите клавишу “Вперёд” и, при наличии соединения, получите доступ к пустой базе данных.

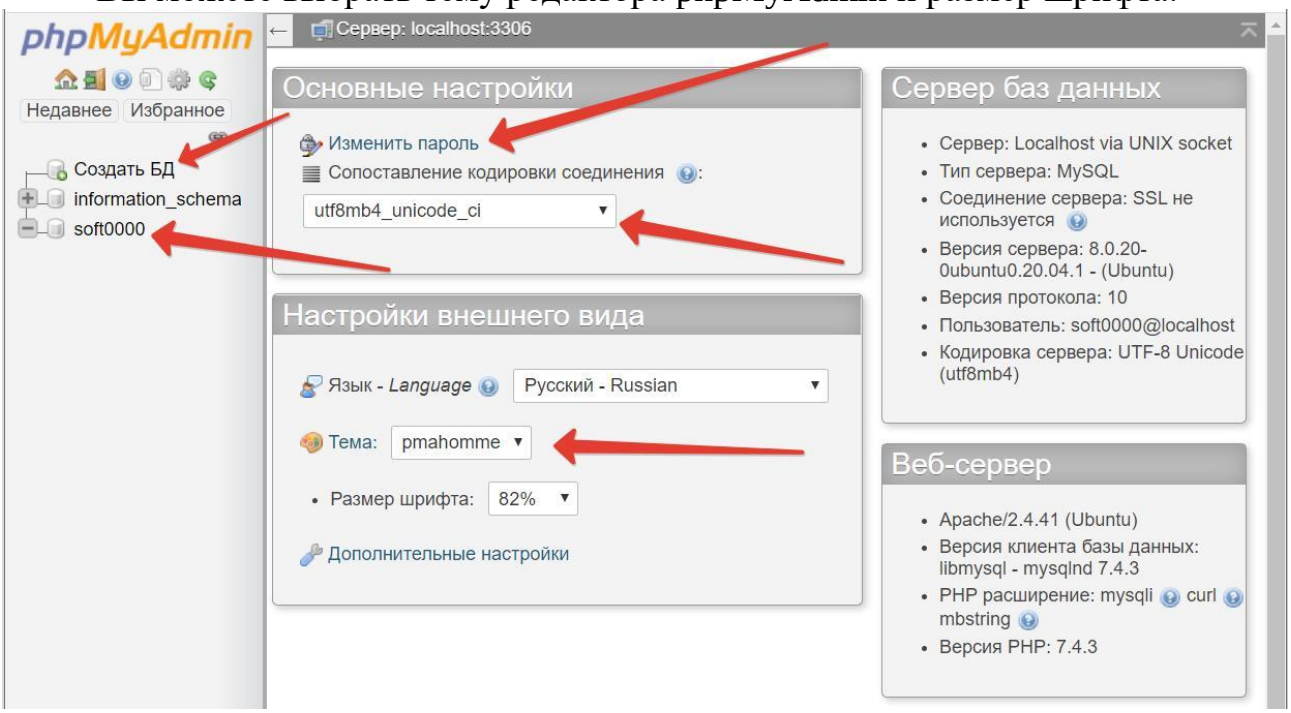
Слева в панели проводника по вашим базам данных вы увидите только одну, пустую пока, базу данных, по имени совпадающую с вашим аккаунтом. Можете начинать работать с ней или создать другую базу (их можно создавать несколько). На вашем уровне доступа установлены ограничения на выбор имени

базы данных – если ваш аккаунт имеет имя soft0000, то новые базы данных могут именоваться, например, так: soft0000_my_db, то есть начальное имя с линией подчёркивания «soft0000_» будет префиксом к новому имени.

Так же наложены ограничения на смену пароля.

Вы можете выбрать кодировку своей базы данных или отдельной таблицы. Выбирайте кодировку с юникодом utf8_unicode_ci или более полную по набору символов utf8mb4_unicode_ci.

Вы можете выбрать тему редактора phpMyAdmin и размер шрифта.



2. Как подключаться к БД через редактор MySQL WorkBench.

Зайдите в редактор WB и на вкладке соединения с базами данных нажмите на маленькую круглую кнопочку с символом «+». У вас появится диалоговое окно настройки подключения.

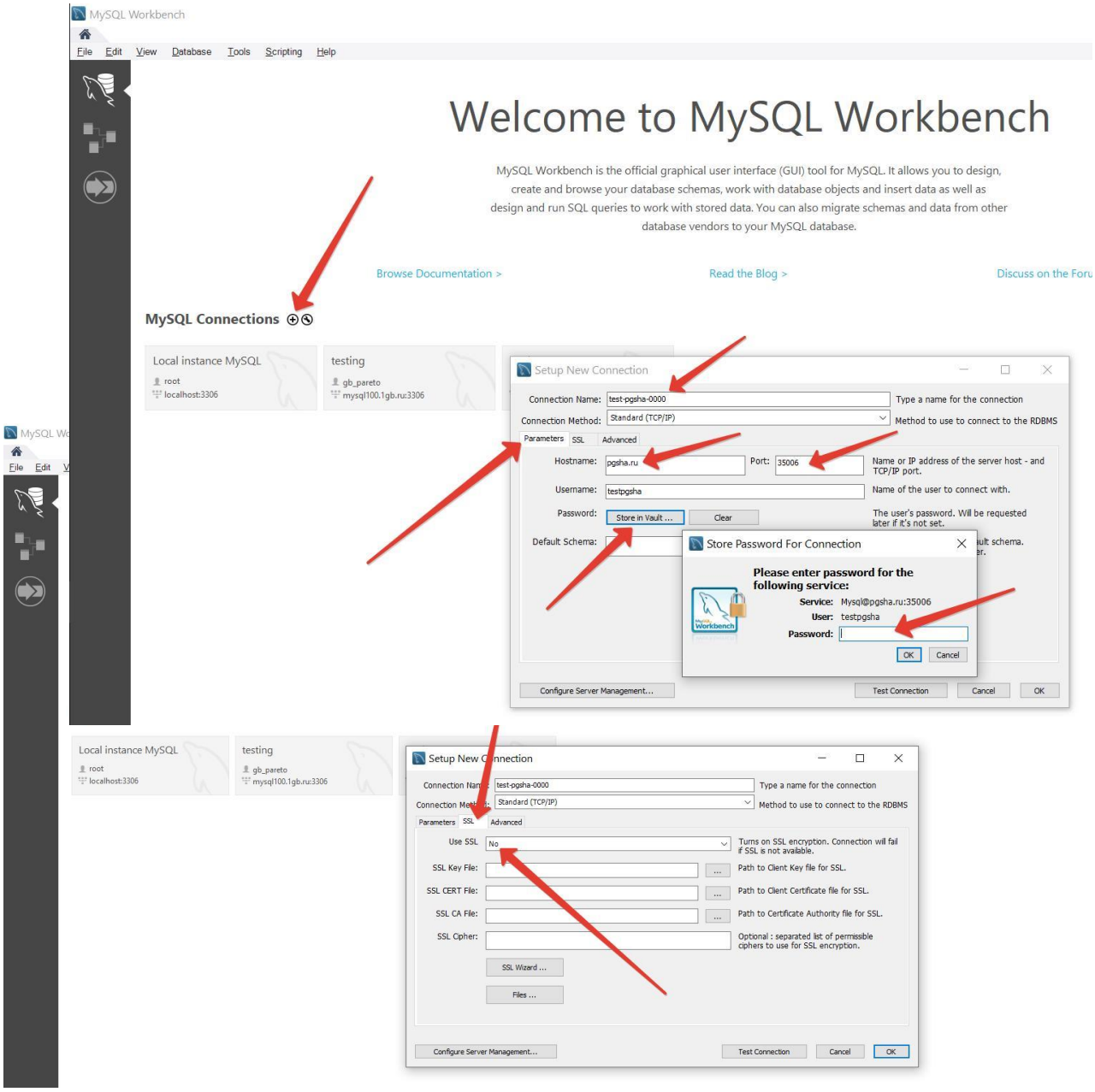
`Connection Name` вы придумываете сами для себя.

Обратите внимание, что для вас есть общие настройки для подключения:

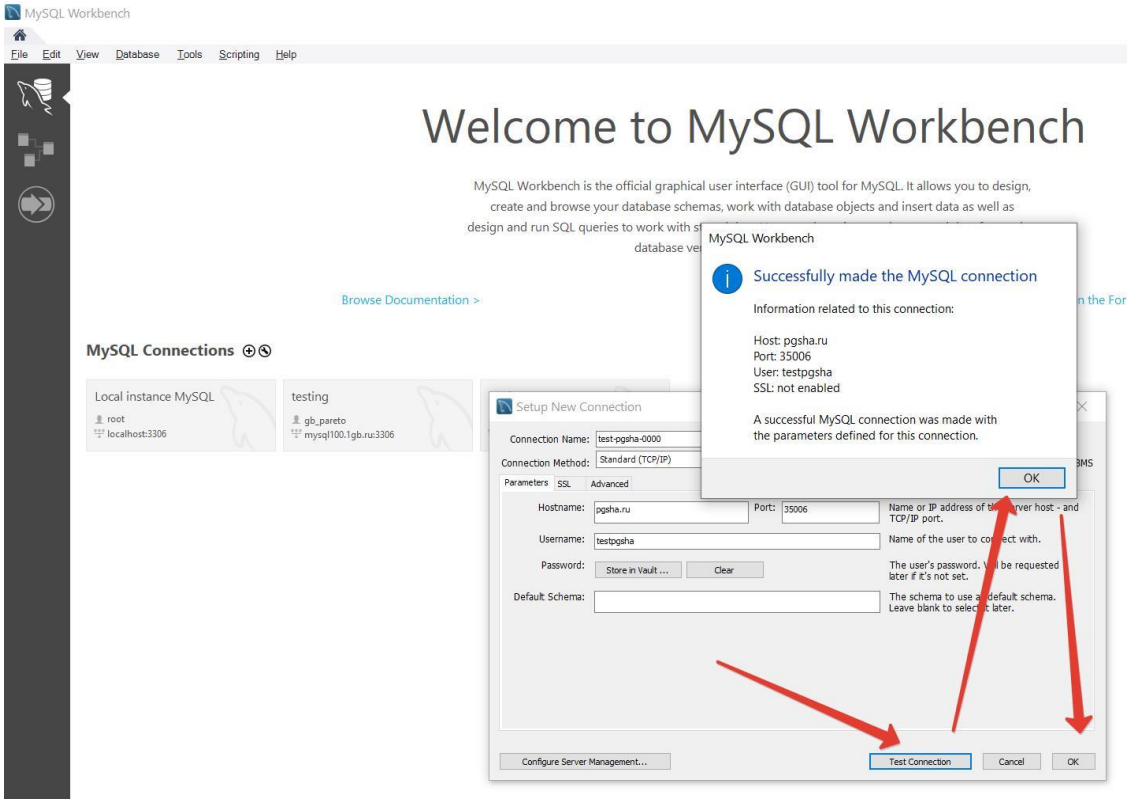
```
Хост: pgsha.ru
Порт: 35006
Use SSL: No
```

и индивидуальные (*их следует получить у преподавателя*):

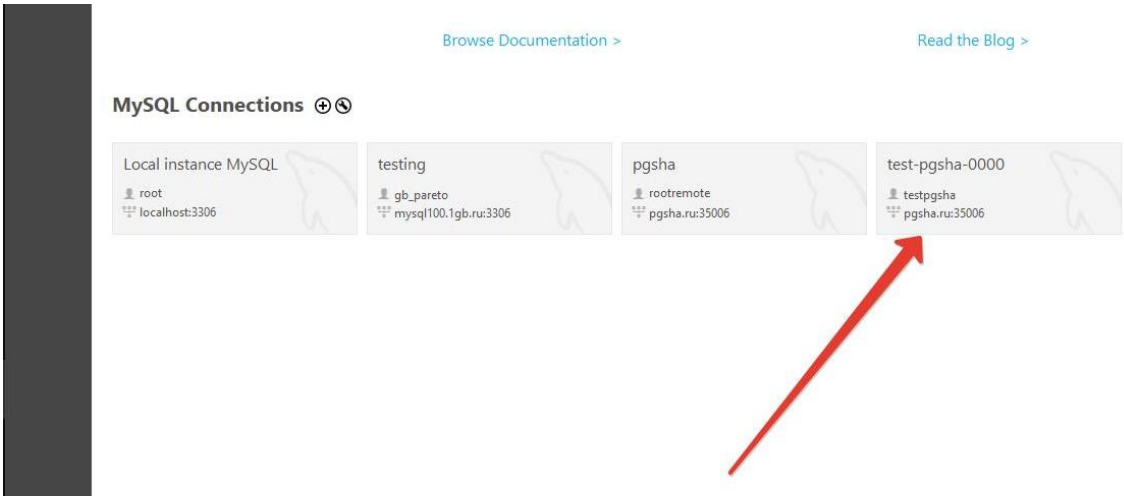
```
Username: ********
Password: ********
```



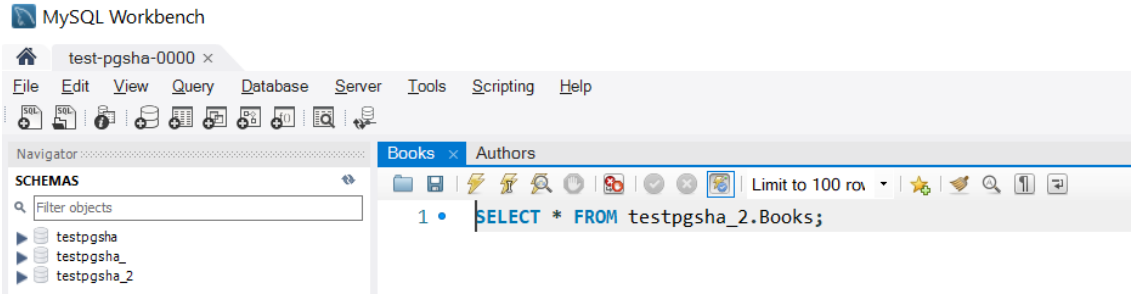
После установки всех параметров можете проверить подключение кнопкой Test Connection.



Если всё пройдёт успешно, то на экране редактора WB вы увидите новое соединение.

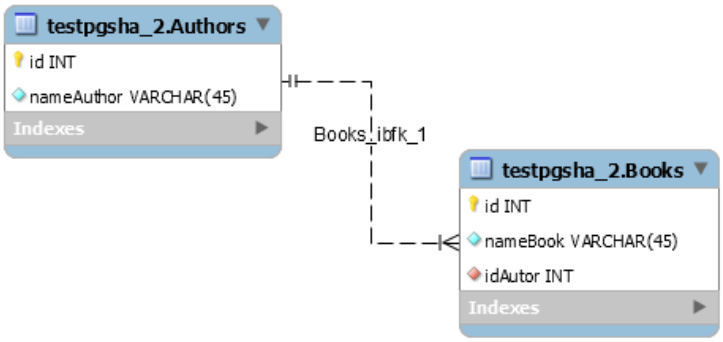


Когда вы на него нажмёте, то редактор WB произведёт соединение к вашему аккаунту на хостинге университета и отобразит базы данных, которые вы там раньше создавали и редактировали.

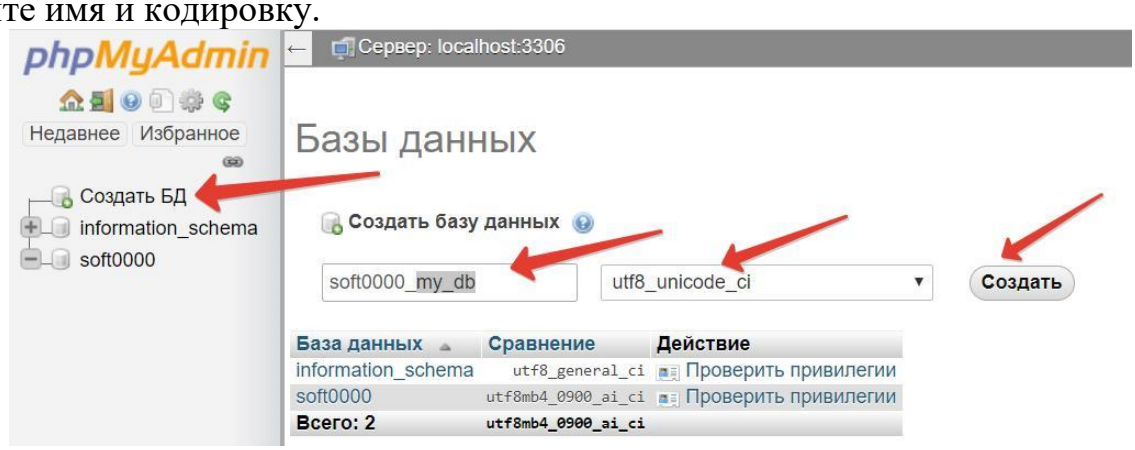


Часть 3. Как наладить FK для связи таблиц в phpMyAdmin.

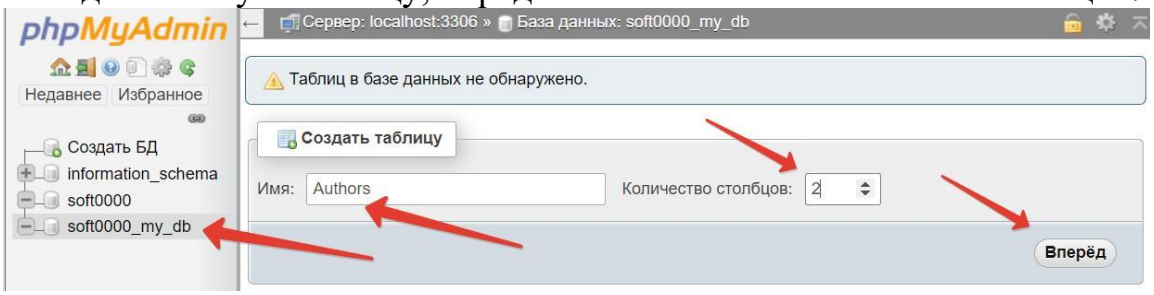
Я планирую создать вот такую схему данных, где есть родительская таблица для хранения авторов книг, и дочерняя таблица для хранения названий книг. Необходимо настроить связь один-ко-многим, поэтому для дочерней таблицы Books нужно будет создать Внешний ключ (FK) – ссылку на соответствующее поле в родительской таблице.



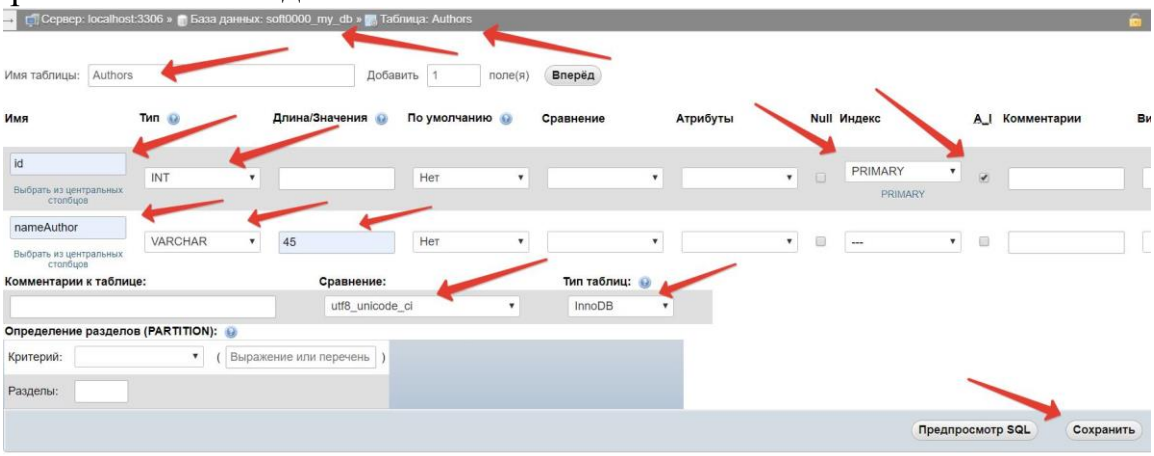
Итак, вы собираетесь создать новую базу данных – нажмите «Создать БД», выберите имя и кодировку.



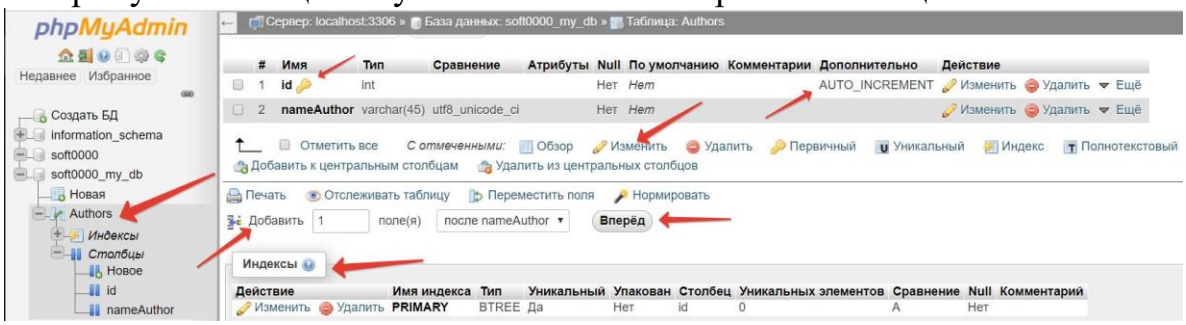
Потом выделите мышкой новую базу данных в проводнике баз данных слева и затем создайте новую таблицу, определив название и количество столбцов.



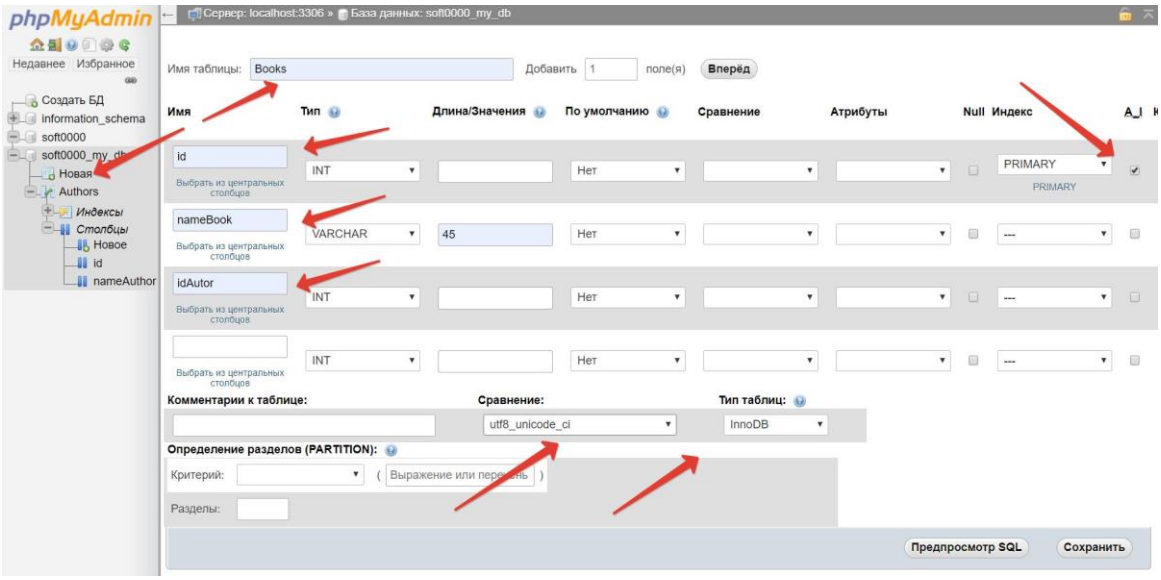
Настройте все необходимые поля



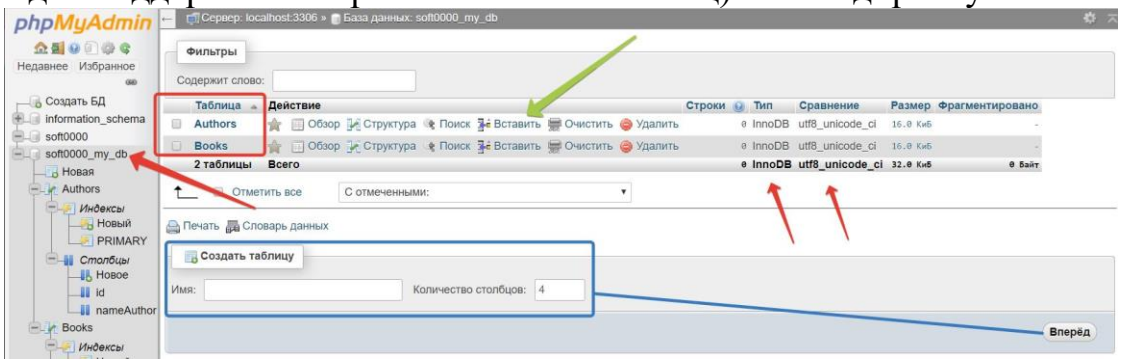
После нажатия на клавишу «Сохранить» вы получите, хоть пока и пустую, но уже готовую для работы таблицу, обратите внимание на выделенные стрелочками атрибуты таблицы и пункты меню по настройке таблицы.



Теперь создадим вторую таблицу, нажав на клавише «Новая» слева в текущей базе данных – настройте все обозначенные параметры и нажмите клавишу «Сохранить».



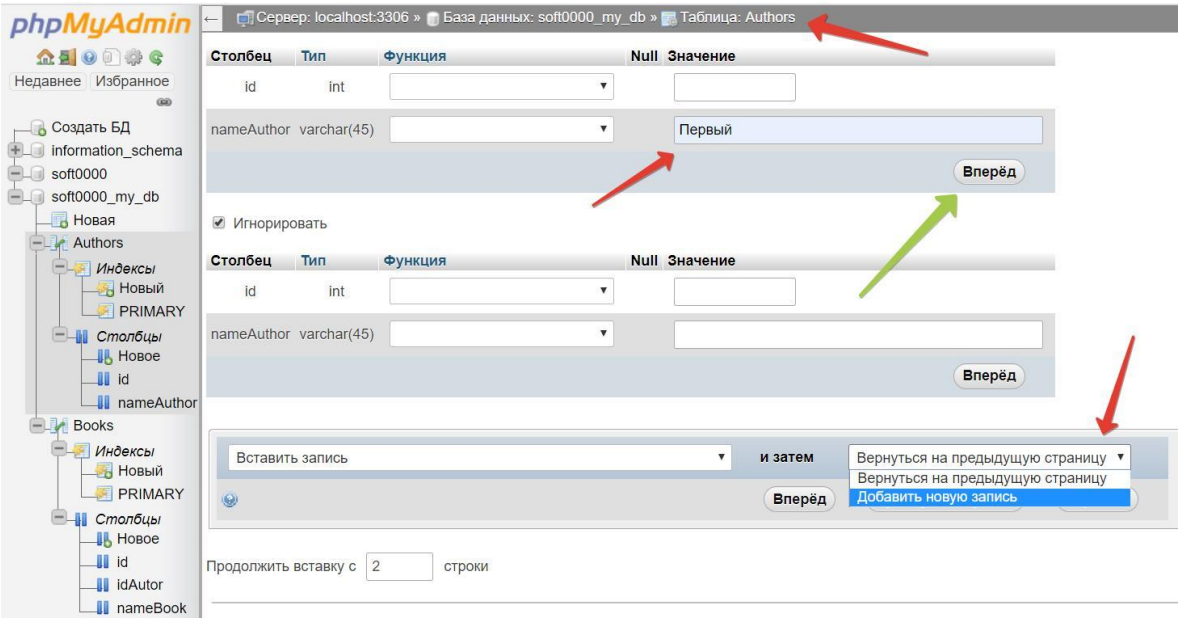
После чего выделите слева мышкой вашу новую Базу данных, и вы получите отображение списка всех таблиц (там же вы можете перейти к созданию новых таблиц). Обратите внимание на Тип созданных таблиц (InnoDB – этот движок нужен для поддержки контроля связанных таблиц) и на кодировку.



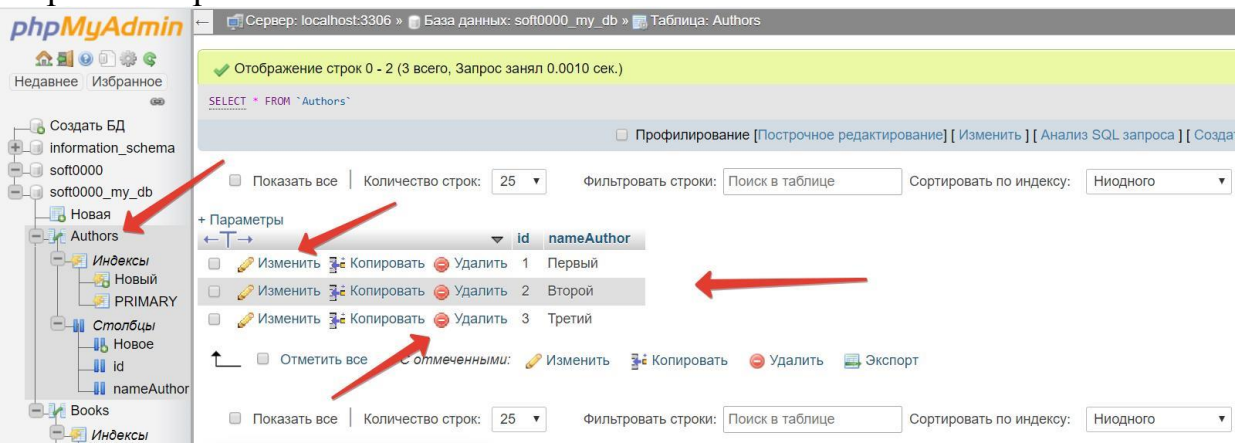
Давайте начнём заполнять таблицы, но, **<u>обязательно, начнём с родительской таблицы</u>**. Дело в том, что дочерняя таблица зависит от родительской, так как дочерняя содержит ссылки на родительскую и не может иметь неправильных ссылок (то есть номеров id, например, которых в реальности нет в родительской

таблице). Сейчас пока связи в виде внешних ключей не установлены, и мы можем заполнять в таблице Books поле idAuthors как угодно, но это, впоследствии, приведёт к существенным проблемам!..

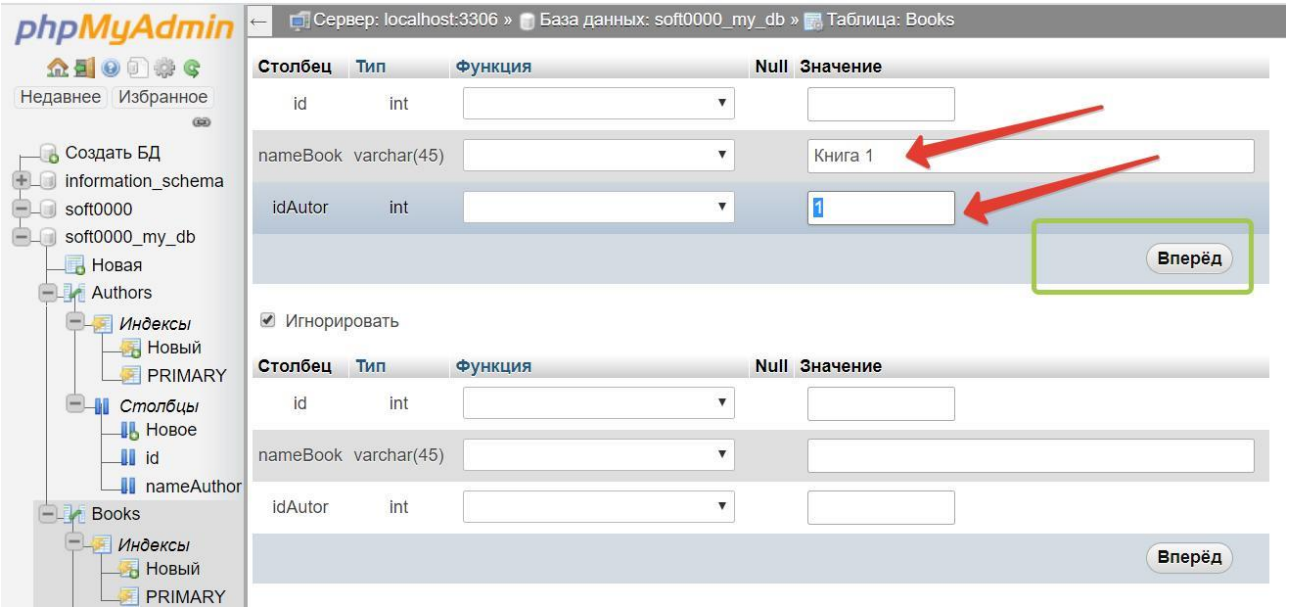
Итак, нажмите клавишу «Вставить» (она обозначена зелёной стрелочкой), чтобы вставить новую запись в таблицу Authors, и получите новое окно (см. далее) для редактирования записи.
Поле id не заполняйте, так как мы для него установили «Автоувеличение индекса» ещё при настройке таблицы. Заполните только поле nameAuthor и нажмите клавишу «Вперёд». Данные добавятся в таблицу в виде новой записи с соответствующими полями и, в зависимости от настроек, вы можете вернуться автоматически к добавлению новой записи или на предыдущую страницу редактора.



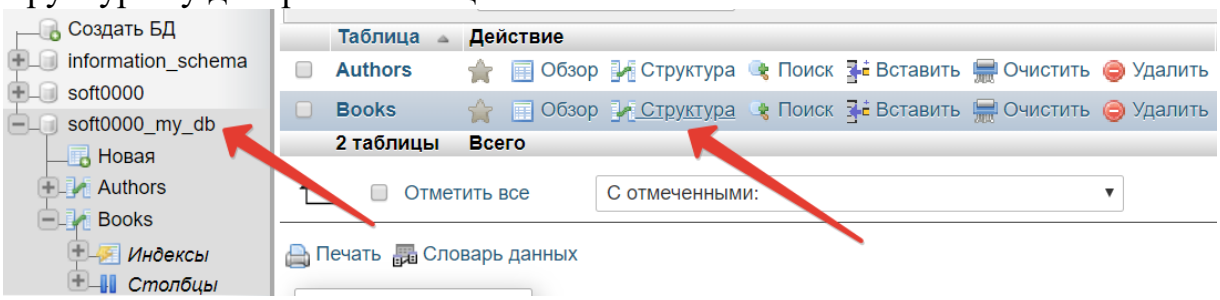
Добавьте аналогичным образом несколько записей, затем в левой части нажмите на таблицу Authors и вы увидите новое окно с данными и получите доступ к некоторым настройкам.



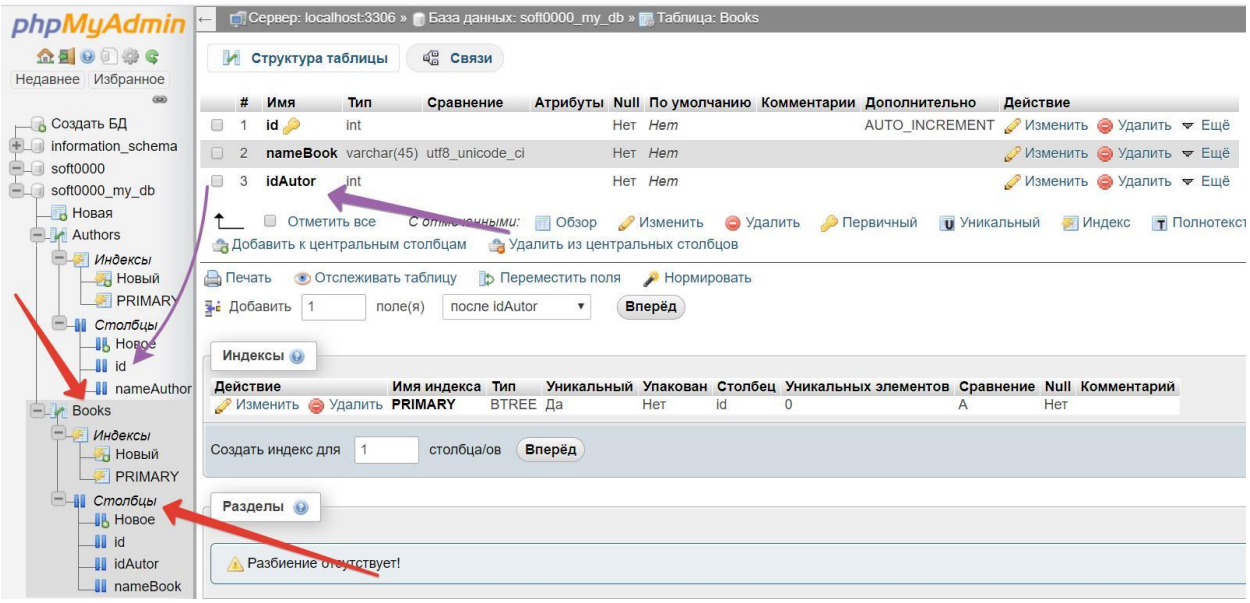
Теперь попробуйте добавлять данные в дочернюю таблицу Books (см. рис. далее) – обратите внимание, что пока не следует наживать клавишу «Вперёд» и, что поле idAuthor позволяет нам ввести любое число, даже такое, которого нет в таблице Authors – это можно исправить назначением внешнего ключа.



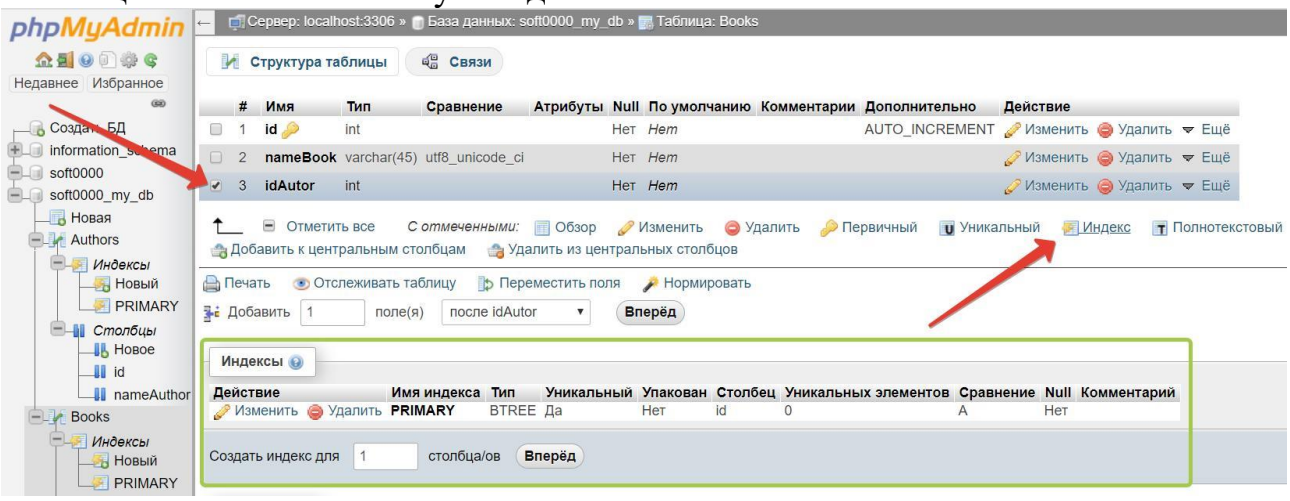
Снова слева выделите мышкой текущую БД и нажмите затем на клавишу «Структура» у дочерней таблицы.



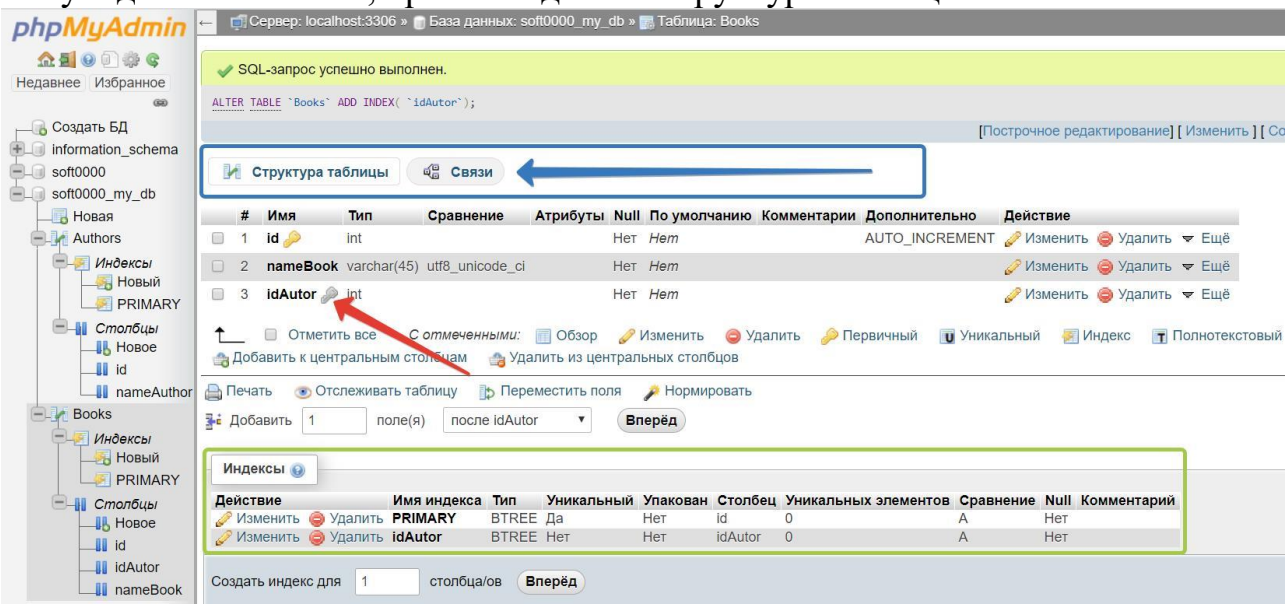
Получите отображение структуры таблицы – обратите внимание – на рисунке фиолетовыми стрелками показан столбец idAutor из таблицы Books и тот столбец, на который он должен ссылаться.



Для назначения foreign key (FK) нужно установить индекс для столбца, от которого будем делать ссылку (Books.idAuthor) и к которому будем ссылаться (Authors.id). Но Authors.id уже имеет индексацию – она автоматически установилась, когда мы этот столбец устанавливали, как PRIMARY KEY при настройке таблицы. Осталось настроить idAuthor – выделите на структуре таблицы этот столбец и нажмите клавишу «Индекс».

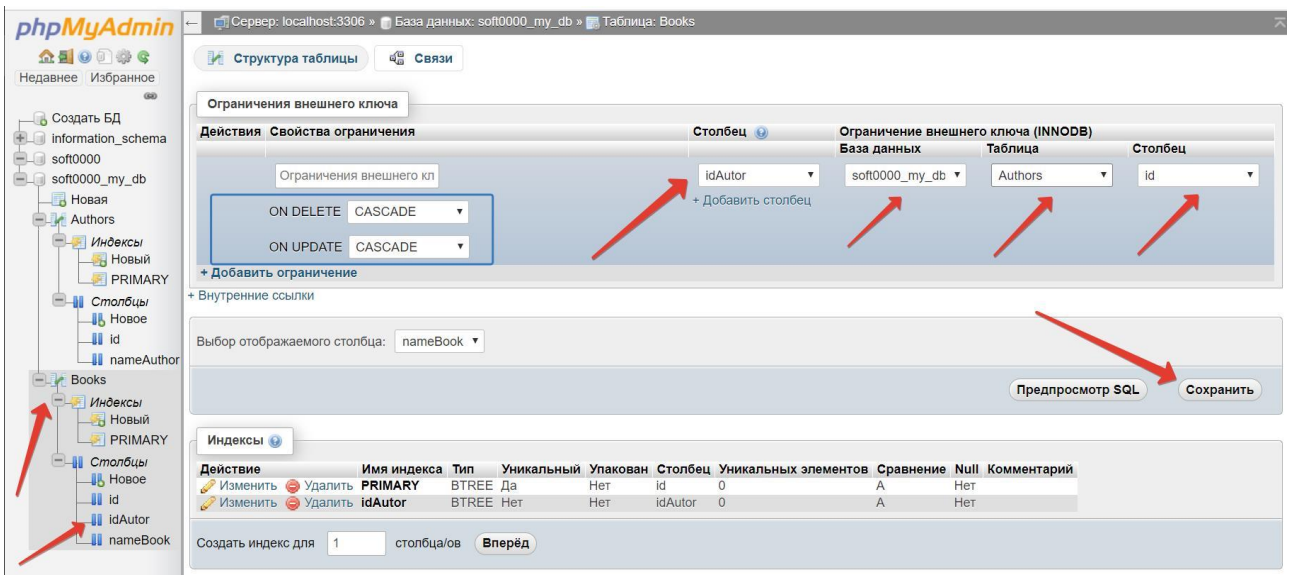


Вы увидите изменения, произошедшие в структуре таблицы.

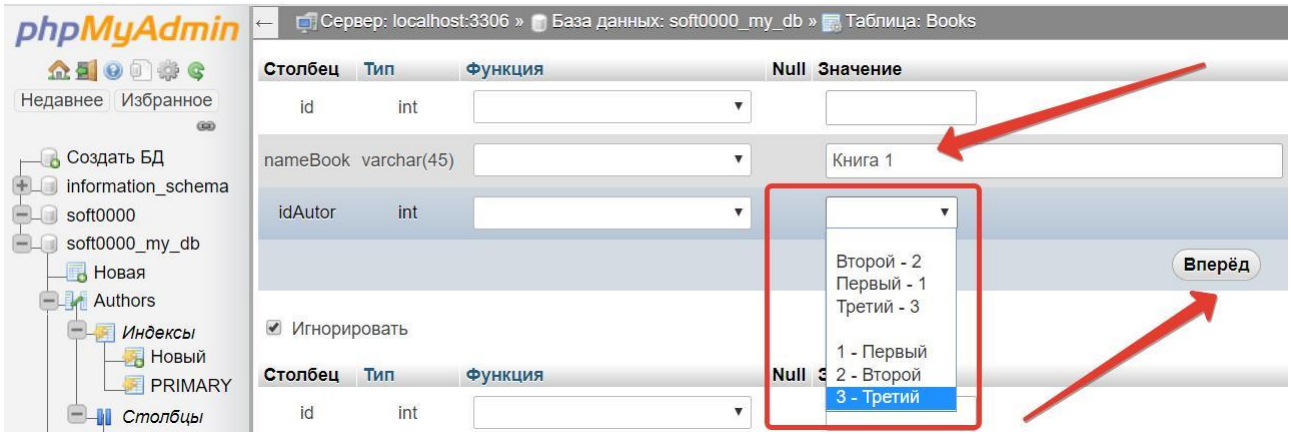


Пришла пора установить внешний ключ (FK), как связь между индексируемыми столбцами разных таблиц – нажмите на клавишу «Связи».

В появившемся окне установите столбец, от которого исходит ссылка и столбец, к которому она ведёт как показано на рисунке далее и нажмите «Сохранить» для сохранения внешнего ключа.



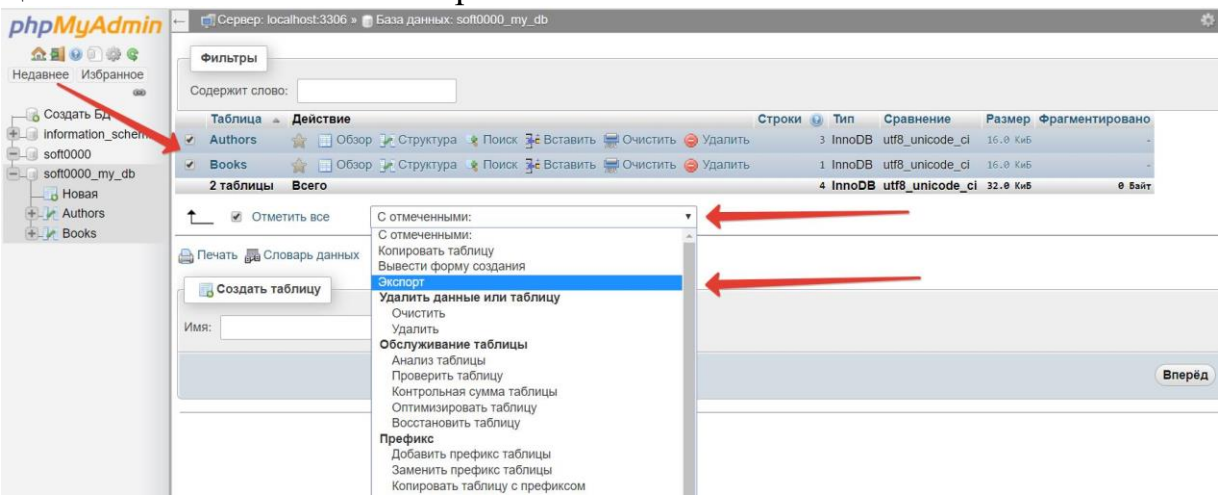
Теперь вернёмся к добавлению записей в таблицу Books – обратите внимание как изменилось поле для ввода idAuthors – теперь производится контроль ввода значений.



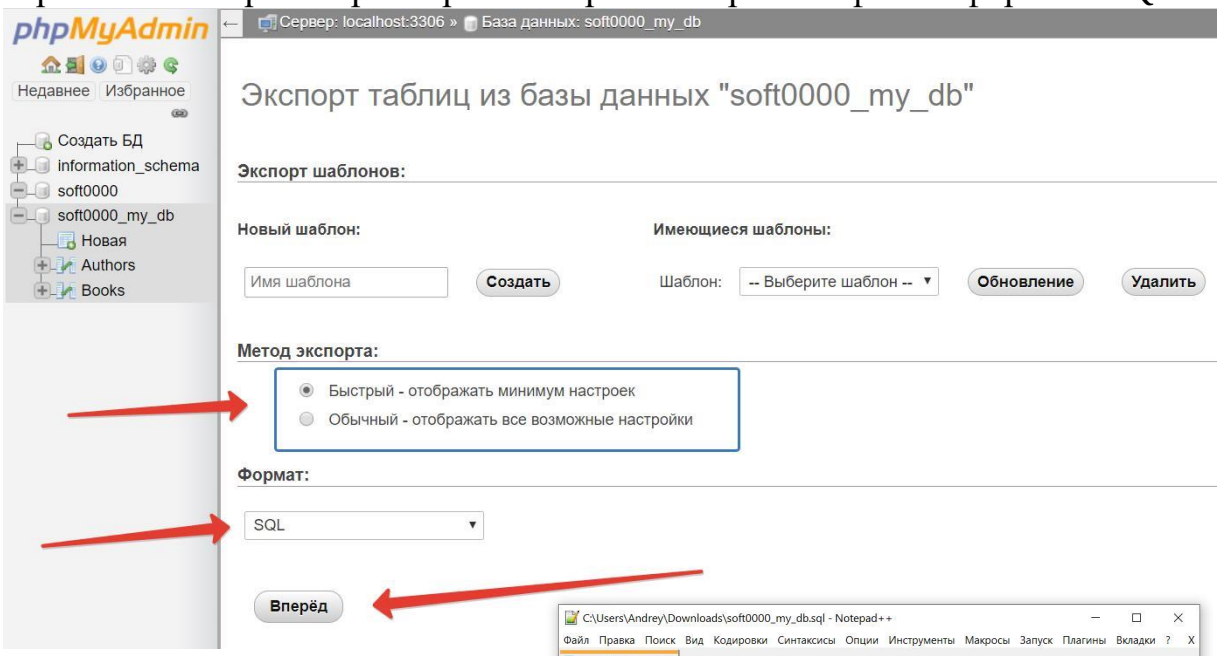
Для дочерней таблицы можно настроить более одного внешнего ключа для связи с разными родительскими таблицами. Это может использоваться для обеспечения связей «многие-ко-многим».

Часть 4. Как экспортировать БД из phpMyAdmin и добавить в WorkBench.

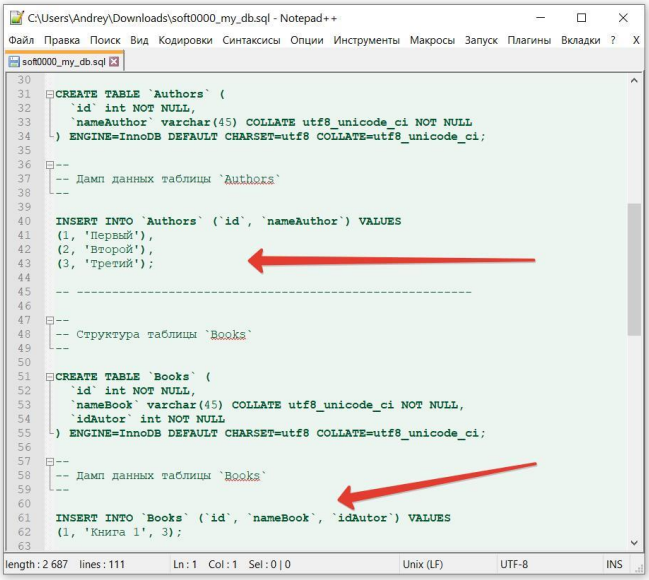
В режиме просмотра структуры базы данных выделите все таблицы и в выпадающем списке нажмите «Экспорт».



Вы попадёте в следующее меню, где можете сделать быстрый экспорт или настроить некоторые параметры экспорта – сохраните файл в формате SQL.

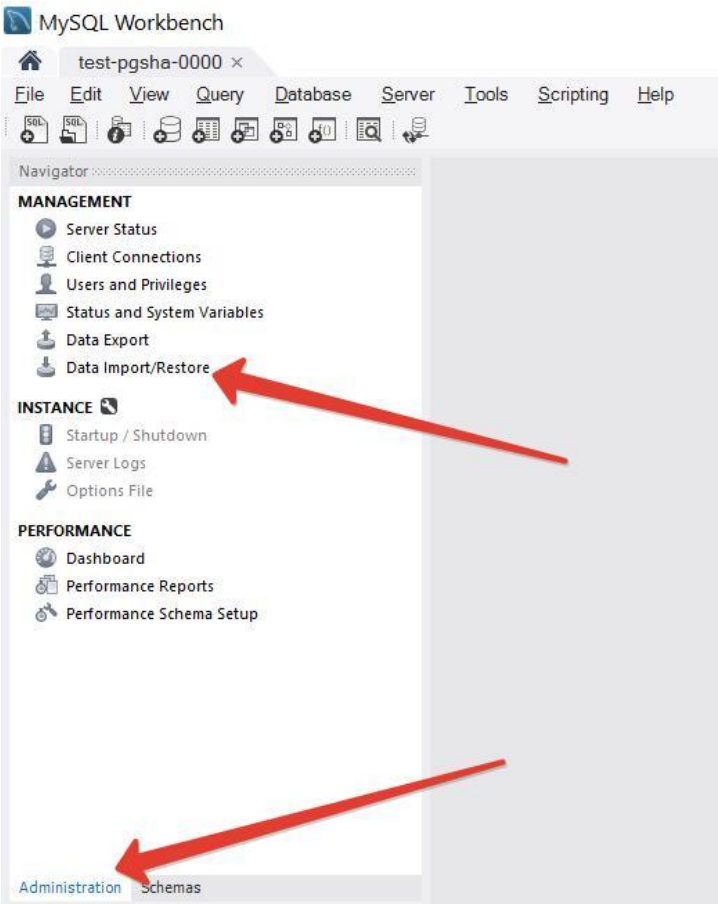


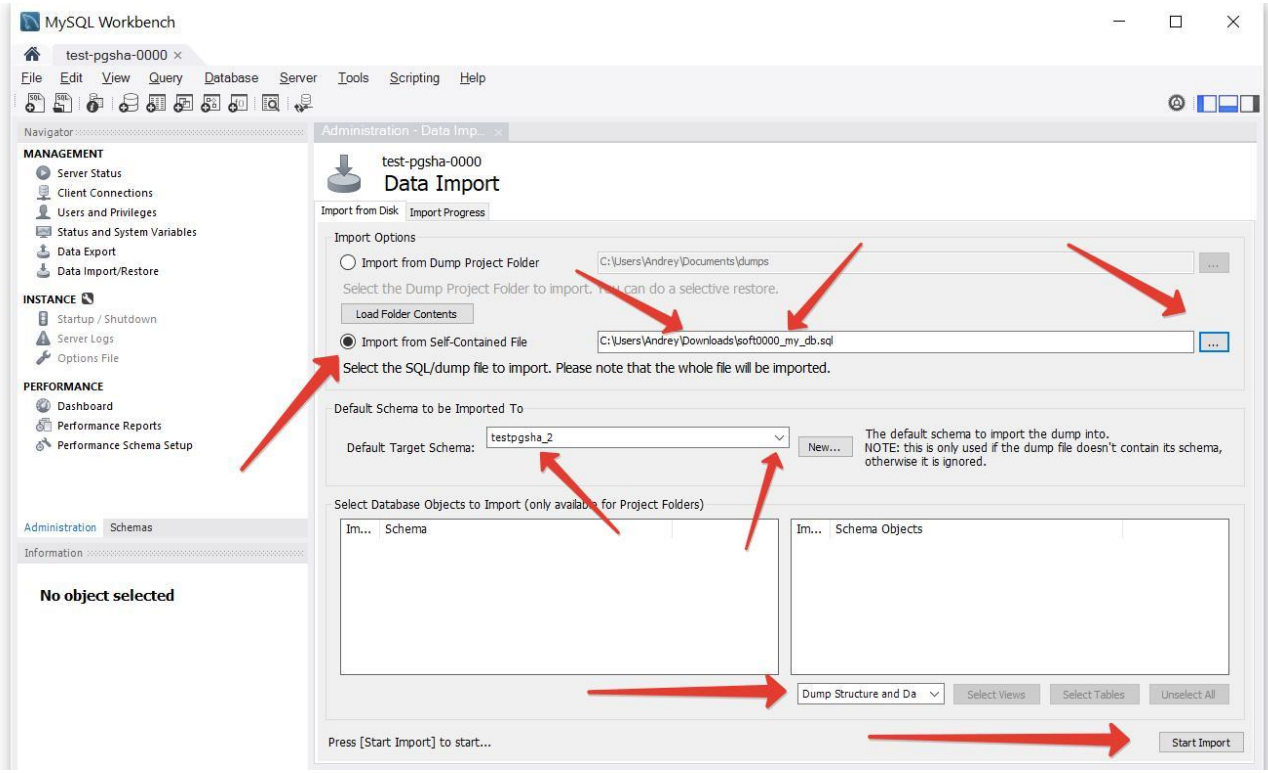
Его можно будет просмотреть обычным текстовым редактором.



Войдите в WB и выберите вкладку Администрирования, а там опцию для импорта данных.

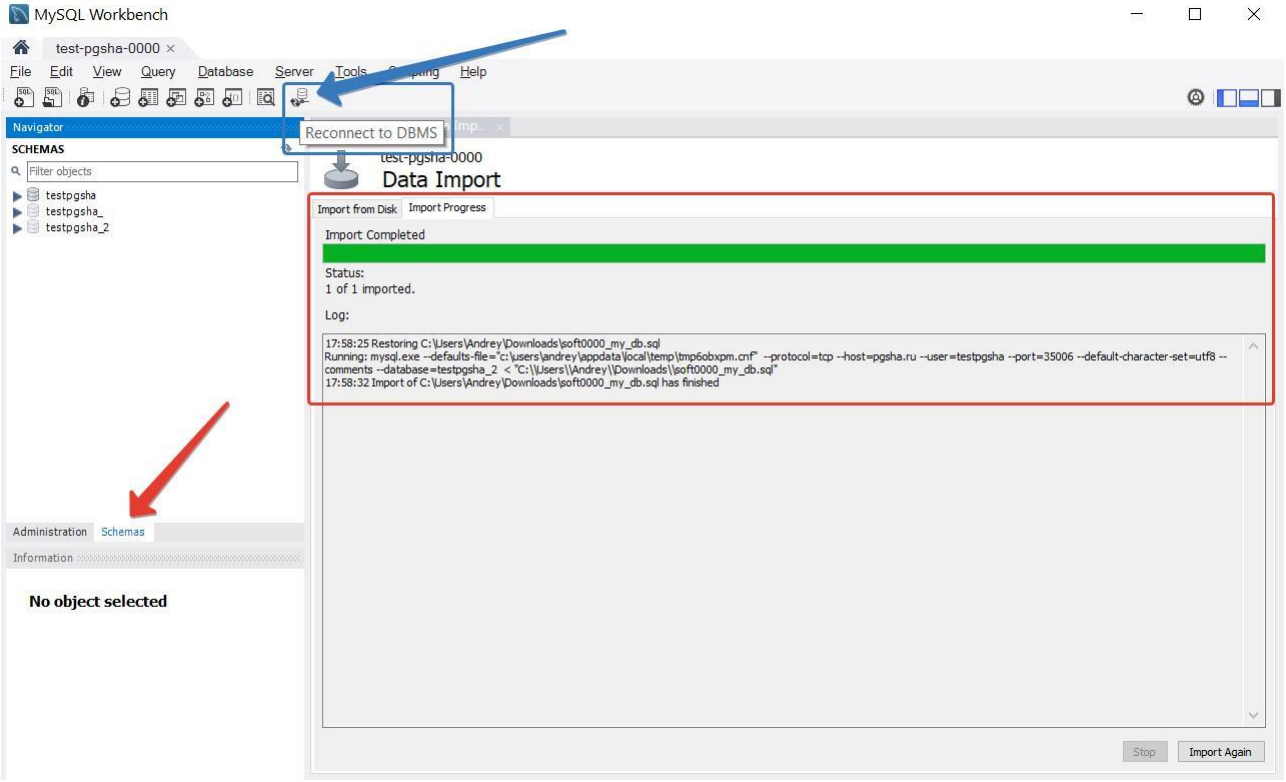
Далее сделайте выбор – импортировать из файла, выберите тот файл, что экспортировали из phpMyAdmin, затем укажите БД, в которую хотите произвести экспорт (или создайте новую) и нажмите Start Import.



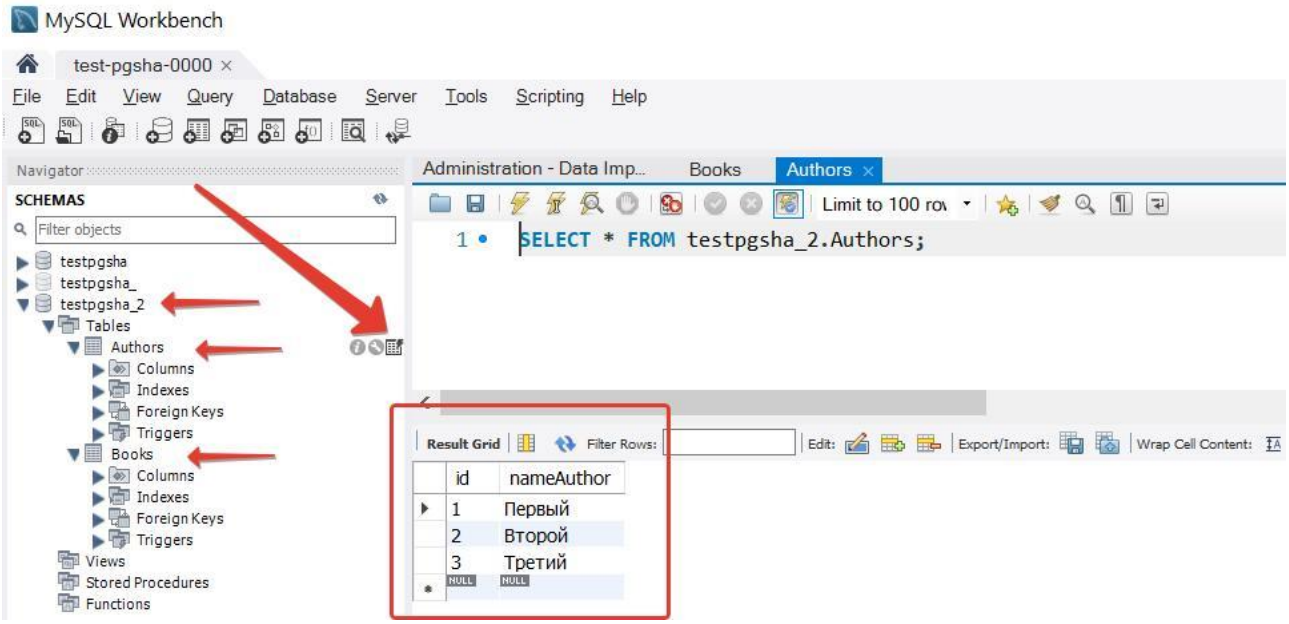


Если всё нормально, то вы увидите отображение процесса импорта и надпись о том, что этот процесс завершился успешно. После чего нажмите клавишу для

обновления соединения и переключитесь на вкладку «Схемы» (это и есть ваши базы данных).



Вы можете раскрыть список таблиц и посмотреть их содержимое.



Так можно поступать, когда вы удалённую базу данных хотите сбросить к себе на компьютер и работать с ней локально.

Даже, если вы не планируете работать с базой данных локально через редактор WorkBench не забывайте периодически описанным выше способом делать дампы вашей базы данных на случай отказа хостинга…