А. И. Новиков

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ В АСУ

Учебно-методическое пособие

Санкт-Петербург
2023

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»
Высшая школа технологии и энергетики

А. И. Новиков

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ В АСУ

Учебно-методическое пособие

Утверждено Редакционно-издательским советом ВШТЭ СПбГУПТД

Санкт-Петербург
2023

УДК 004.655.3
ББК 32.972.134
Н731

Рецензенты:
кандидат технических наук, доцент, заведующий кафедрой прикладной математики и информатики Высшей школы технологии и энергетики Санкт-Петербургского государственного университета промышленных технологий и дизайна
И. В. Ремизова;
доктор технических наук, профессор, заведующий кафедрой автоматизации процессов химической промышленности Санкт-Петербургского государственного технологического института (Технического университета)
Л. А. Русинов

Новиков, А. И.
Н731 Системы управления базами данных в АСУ: учебно-методическое пособие / А. И. Новиков. — СПб.: ВШТЭ СПбГУПТД, 2023. — 139 с.

Учебно-методическое пособие соответствует программам и учебным планам дисциплин «Системы управления базами данных в АСУ» и «Облачные технологии в СУБД» для студентов всех форм, обучающихся по направлениям подготовки 27.03.04 «Управление в технических системах» и 09.03.03 «Прикладная информатика», а также по направлениям 01.03.02 «Прикладная математика и информатика» и 15.03.04 «Автоматизация технологических процессов и производств». Изложены основы работы с различными СУБД и синтаксис языка SQL. Приведены примеры выполнения SQL-запросов, в том числе с использованием языков PHP и HTML. Содержит разделы для самостоятельного углубленного изучения.

Пособие предназначено для подготовки бакалавров очной и заочной форм обучения. Отдельные разделы пособия могут быть полезны магистрантам и аспирантам.

УДК 004.655.3
ББК 32.972.134

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Система управления базами данных (СУБД, DBMS) – это комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, выбирать и удалять). В большинстве случаев речь идет о Реляционных базах данных – т. е. о таких БД, которые представляются в виде двумерных таблиц. В качестве примера другой часто встречающейся модели данных стоит упомянуть Иерархическую, т. е. такую, в которой данные представлены в виде «дерева» (как, например, структура каталогов на диске ПК).

Для «общения» с БД используется язык структурированных запросов SQL. Язык SQL описан как международный стандарт ISO/IEC 9075. SQL не является конкретным программным обеспечением от одного производителя, это стандарт, имеющий различные реализации от различных производителей. Поэтому в данном пособии рассматривается работа с двумя СУБД: Microsoft Access и MySQL.

На самом деле, SQL не предназначен для постоянного ручного ввода запросов к базе данных. Он предназначен для автоматизации процесса проведения запросов и используется в составе какого-либо из языков программирования. В данном случае для этого применяется язык PHP, который, в свою очередь, работает как препроцессор для языка HTML.

* Пособие содержит разделы для самостоятельного углубленного изучения, названия которых отмечены звездочкой.

1. ЯЗЫК SQL

СУБД – Система управления базами данных (**DBMS**, Database Management System)[1] – комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, выбирать и удалять). Система обеспечивает безопасность, надежность хранения и целостность данных, а также предоставляет средства для администрирования БД.

В большинстве случаев речь идет о **Реляционных базах данных** (и, соответственно, о реляционных СУБД, РСУБД[2]) – т. е. о таких БД, которые представляются в виде двумерных таблиц. Название происходит от англ. **relation** (отношение, зависимость, связь). С реляционными БД тесно связано понятие **нормализация** – т. е. процесс устранения недостатков структуры БД, приводящих к ее **избыточности**, а также теория (нормализации) для осуществления данного процесса.

В качестве примера другой часто встречающейся модели данных стоит упомянуть **Иерархическую**, т. е. такую, в которой данные представлены в виде «**дерева**» (как например структура каталогов на диске ПК).

SQL (Structured Query Language, язык структурированных запросов) – декларативный язык программирования (т. е. описывающий требуемый результат, а не способ его получения), применяемый для управления данными в реляционной БД (управляемой в свою очередь соответствующей СУБД). Читается как «Эс-Ку-Эль» или «Эс-Кью-Эль». Язык SQL описан как международный стандарт **ISO/IEC 9075** (а изначально был принят в качестве американского стандарта **ANSI**). Технически, как язык программирования, SQL является так называемым языком программирования «не полным по Тьюрингу».

Операторы SQL делятся на:

- операторы определения данных, например, **CREATE** и **DROP** – для создания и удаления таблицы (или другого объекта);
- операторы манипуляции данными, например, **SELECT** – для выбора данных, удовлетворяющих заданным условиям, или **INSERT**, **UPDATE** и **DELETE** – для добавления, изменения и удаления данных;
- и др.

Операторы языка SQL (и другие ключевые слова) принято писать прописными (т. е. БОЛЬШИМИ) буквами.

!!! При выполнении студенческих работ в рамках данного курса эта рекомендация является обязательной к применению**!**

В конце SQL-запроса ставится точка с запятой.

SQL не является конкретным программным обеспечением от одного производителя. Это стандарт, имеющий различные реализации от различных производителей, например, Microsoft **SQL Server**, или **MySQL**. В **Microsoft Access** также имеется ограниченная поддержка SQL.

1.1. Синтаксис языка SQL

!!! *При выполнении студенческих работ запросы должны оформляться в несколько строк (как показано в примерах ниже), так, чтобы строка начиналась с нового ключевого слова***!**

1. Комментарий

```
/* Текст комментария (может занимать несколько строк) */
```

2. Создание таблицы

Например, список в котором отмечено, сколько лабораторных работ выполнил каждый из студентов, может быть создан следующим образом:

CREATE TABLE Students (No INTEGER **NOT NULL PRIMARY KEY**,
FIO VARCHAR(50),
Lab_Count INTEGER);

Фактически, эта команда создает «шапку» для таблицы (рис. 1) и предписывает, данные какого типа будут храниться в каждом столбце.

No	FIO	Lab_Count

Рис. 1. Шапка таблицы

В общем виде записывается как:

CREATE TABLE имя_табл (столбец1 тип_данных1 [**NOT NULL**] [**PRIMARY KEY**],
столбец2 тип_данных2 …);

Основные типы данных:

- **INTEGER** или **INT** – целые числа;
- **REAL** – числа с плавающей точкой (запятой);
- **DATETIME** – дата и/или время;
- **VARCHAR**(n) – строка (текст) заданной предельной длинны n символов.

Строковые значения (текст) вводятся в одинарных кавычках.

PRIMARY KEY – «Первичный ключ», однозначно идентифицирующий каждую запись в таблице. Также означает, что данная колонка должна содержать уникальные значения, которые не могут повторяться. Не может принимать значение **NULL**. Таблица может иметь только один первичный ключ *(хотя он и может состоять из нескольких столбцов)*.

NOT NULL – означает, что значение не может быть пустым (содержать **NULL**). По существу, **NOT NULL** делает соответствующее поле обязательным для заполнения командой **INSERT**.

3. Добавление строк в таблицу

INSERT INTO имя_таблицы (столбец_1, столбец_2, …)

VALUES (значение_1, значение_2, …);

*В скобках, после имени таблицы, могут быть указаны не все имена столбцов, которые использовались при создании таблицы (**CREATE TABLE**). Незадействованные поля будут заполнены значениями **NULL**. Но при этом столбцы, помеченные как **NOT NULL,** обязательны для заполнения. Также здесь можно изменить порядок следования столбцов.*

Например:

```
INSERT INTO Students (No, FIO, Lab_Count)
VALUES (1, 'Иванов Иван Иванович', 2);
/* Причем, если имена столбцов идут в том же порядке, в каком они созданы командой CREATE, то их можно вообще не указывать */
INSERT INTO Students
VALUES (2, 'Петров Петр Петрович', 0);
/* Если нужно добавить сразу несколько строк, то INSERT можно написать только один раз */
INSERT INTO Students
VALUES (3, 'Сидоров Сидор Сидорович', 1),
       (4, 'Николаев Николай Николаевич', 0),
       (5, 'Синицын Синиц Синицевич', 3),
       (6, 'Новиков Александр Игоревич', 4);
```

Результат добавления строк представлен на рис. 2.

No	FIO	Lab_Count
1	Иванов Иван Иванович	2
2	Петров Петр Петрович	0
3	Сидоров Сидор Сидорович	1
4	Николаев Николай Николаевич	0
5	Синицын Синиц Синицевич	3
6	Новиков Александр Игоревич	4

Рис. 2. Таблица с добавленными строками

4. Вывод всего содержимого таблицы

SELECT * FROM имя_таблицы;

Например (рис. 2):

SELECT * FROM Students;

!!! *В виде исключения, запрос на вывод всего содержимого таблицы можно (и даже нужно!) писать в одну строку.*

5. Вывод части столбцов таблицы

SELECT выводимый_столбец_1, выводимый_столбец_2, …

FROM имя_таблицы;

Например:

SELECT Lab_Count, FIO

FROM Students;

Данная функция позволяет вывести не все столбцы (как при использовании звездочки ***), а только требуемые из них. Причем можно поменять их порядок. Результат выполнения запроса из примера представлен на рис. 3.

Lab_Count	FIO
2	Иванов Иван Иванович
0	Петров Петр Петрович
1	Сидоров Сидор Сидорович
0	Николаев Николай Николаевич
3	Синицын Синиц Синицевич
4	Новиков Александр Игоревич

Рис. 3. Вывод части столбцов

6. Указание имени столбца

выводимый_столбец **AS** "Новое имя столбца"

Например:

```
SELECT No, FIO AS "Ф.И.О." , Lab_Count AS "Сдано работ"
FROM Students;
```

В новом имени столбца можно использовать пробелы и русские буквы (рис. 4).

No	Ф.И.О.	Сдано работ
1	Иванов Иван Иванович	2
2	Петров Петр Петрович	0
3	Сидоров Сидор Сидорович	1
4	Николаев Николай Николаевич	0
5	Синицын Синиц Синицевич	3
6	Новиков Александр Игоревич	4

Рис. 4. Новые имена столбцов

7. Вывод части строк таблицы

Например, выведем список студентов, у которых не сдано две или более лабораторные работы из четырех, т. е. сдано 0, 1 или 2 работы (рис. 5):

```
SELECT *
FROM Students
WHERE Lab_Count < 3;
```

No	FIO	Lab_Count
1	Иванов Иван Иванович	2
2	Петров Петр Петрович	0
3	Сидоров Сидор Сидорович	1
4	Николаев Николай Николаевич	0

Рис. 5. Вывод части строк

В общем виде записывается как:

SELECT выводимый_столбец_1, выводимый_столбец_2, …
FROM имя_таблицы
WHERE условие_отбора;

Где «**условие_отбора**» – запись, содержащая *оператор сравнения* (или несколько операторов сравнения и логические операторы).

Логические операторы, используемые для объединения условий, представлены в табл. 1.

Операторы сравнения представлены в табл. 2. Сравниваемые значения могут быть числами, текстом (строковыми значениями) или датами и временем.

Таблица 1 – Логические операторы

Оператор	Описание
NOT	«НЕ», логическое отрицание, выводит противоположные значения
AND	«И», логическое умножение
OR	«ИЛИ», логическое сложение

Таблица 2 – Операторы сравнения

Оператор	Описание
Простые	
=	Равно
>	Больше
<	Меньше
< > *или* !=	Не равно
>=	Больше или равно (не меньше)
<=	Меньше или равно (не больше)
Сложные	
BETWEEN	«Между», в заданном диапазоне
IN	«В» списке (позволяет перечислить несколько значений для сравнения)
LIKE	«Подобный» («похожий»), поиск подстроки по шаблону

8. Оператор BETWEEN

WHERE сравниваемый_столбец **BETWEEN** от **AND** до;

Например, выведем список всех студентов (рис. 6), которые уже выполнили хотя бы одну лабораторную работу (>= 1), но еще не закончили выполнение всех работ (< 4):

```
SELECT *
FROM Students
WHERE Lab_Count BETWEEN 1 AND 3;
```

No	FIO	Lab_Count
1	Иванов Иван Иванович	2
3	Сидоров Сидор Сидорович	1
5	Синицын Синиц Синицевич	3

Рис. 6. Диапазон значений

Данное условие эквивалентно следующей записи через простые операторы сравнения:

```
WHERE (Lab_Count >= 1) AND (Lab_Count <= 3);
```

Если необходимо вывести значения, находящиеся, наоборот, за пределами диапазона, то применяется **NOT BETWEEN**. Например (рис. 7):

```
SELECT *
FROM Students
WHERE Lab_Count NOT BETWEEN 1 AND 3;
```

No	FIO	Lab_Count
2	Петров Петр Петрович	0
4	Николаев Николай Николаевич	0
6	Новиков Александр Игоревич	4

Рис. 7. За пределами диапазона

Данное условие с **NOT BETWEEN** эквивалентно следующей записи через простые операторы сравнения:

```
WHERE (Lab_Count < 1) OR (Lab_Count > 3);
```

9. Оператор IN

WHERE сравниваемый_столбец **IN** (Значение_1, Значение_2, …);

Например, узнаем количество выполненных лабораторных работ для заданного списка студентов (рис. 8):

```
SELECT *
FROM Students
WHERE FIO IN ('Иванов Иван Иванович',
              'Петров Петр Петрович',
              'Николаев Николай Николаевич');
```

No	FIO	Lab_Count
1	Иванов Иван Иванович	2
2	Петров Петр Петрович	0
4	Николаев Николай Николаевич	0

Рис. 8. Поиск по списку

Данное условие эквивалентно следующей записи через простые операторы сравнения и логический оператор **OR**:

```
WHERE  FIO = 'Иванов Иван Иванович' OR
       FIO = 'Петров Петр Петрович' OR
       FIO = 'Николаев Николай Николаевич';
```

Если необходимо вывести значения, наоборот, не входящие в список, то применяется **NOT IN**. Например (рис. 9):

```
SELECT *
FROM Students
WHERE FIO NOT IN  ('Иванов Иван Иванович',
                   'Петров Петр Петрович',
                   'Николаев Николай Николаевич');
```

No	FIO	Lab_Count
3	Сидоров Сидор Сидорович	1
5	Синицын Синиц Синицевич	3
6	Новиков Александр Игоревич	4

Рис. 9. Поиск вне списка

Данное условие с **NOT IN** эквивалентно следующей записи через простые операторы сравнения:

```
WHERE  FIO <> 'Иванов Иван Иванович' AND
       FIO <> 'Петров Петр Петрович' AND
       FIO <> 'Николаев Николай Николаевич';
```

10. Условие с указанием части строки (шаблона)

```
WHERE имя_столбца Like '%часть_значения%';
```

Позволяет проводить поиск подстроки по шаблону. Для замены в тексте некоторых символов используются следующие знак:

«**_**» *(нижняя черта)* – для замены ровно 1 символа;
«**%**» – для замены любого количества символов (от 0 до ∞ символов).

Можно также комбинировать данные знаки, например:

«**_%**» – для замены от 1 до ∞ символов;
«**_ _%**» – для замены от 2 до ∞ символов;
и т. д…
«**_ _**» – для замены ровно 2 символов;
«**_ _ _**» – для замены ровно 3 символов;
и т. д…

В сравниваемом с шаблоном тексте на месте этих знаков может располагаться указанное количество *любых* символов.

!!! В СУБД **Microsoft Access** вместо знаков «_» и «**%**» применяются, соответственно, знаки «**?**» (для одного символа) и «*****» (для любого количества символов).

Например, выведем список всех студентов, чьи фамилии начинаются на букву «Н» (рис. 10):

```
SELECT *
FROM Students
WHERE FIO Like 'Н%';
```

No	FIO	Lab_Count
4	Николаев Николай Николаевич	0
6	Новиков Александр Игоревич	4

Рис. 10. Поиск строки по шаблону

11. Сортировка

SELECT выводимый_столбец_1, выводимый_столбец_2, …
FROM имя_таблицы
WHERE условие_отбора
ORDER BY столбец_для_сортировки [**ASC** | **DESC**];

Для сортировки по возрастанию, к **ORDER BY** добавляется ключевое слово **ASC**. Для сортировки по убыванию к **ORDER BY** добавляется ключевое слово **DESC**. *По умолчанию (если не указано ни **ASC**, ни **DESC**), сортировка производится по возрастанию.*

Например (рис. 11):

SELECT *
FROM Students
ORDER BY Lab_Count **DESC**

No	FIO	Lab_Count ▾ 1
6	Новиков Александр Игоревич	4
5	Синицын Синиц Синицевич	3
1	Иванов Иван Иванович	2
3	Сидоров Сидор Сидорович	1
2	Петров Петр Петрович	0
4	Николаев Николай Николаевич	0

Рис. 11. Сортировка по убыванию

*Допускается (но не рекомендуется**!**) использовать вместо имени столбца для сортировки его номер. Так равносильна примеру, рассмотренному выше, будет запись:*

ORDER BY 3 **DESC**

Кроме того, для сортировки можно применить сразу несколько столбцов, например:

ORDER BY Lab_Count **DESC**, FIO **ASC**

В этом случае, вначале таблица будет отсортирована по убыванию Lab_Count, после чего строки с одинаковым значением Lab_Count будут выстроены по возрастанию FIO.

12. Агрегатные функции

В **SQL** существует 5 базовых агрегатных функций:

- **SUM**(имя_столбца) – функция, возвращающая сумму значений указанного столбца. Может применяться только для *числовых* столбцов;
- **MIN**(имя_столбца) – функция, возвращающая минимальное значение в указанном столбце. Может применяться *не* только для числовых столбцов, но и для строк и дат;
- **MAX**(имя_столбца) – функция, возвращающая максимальное значение в указанном столбце. Может применяться *не* только для числовых столбцов, но и для строк и дат;
- **AVG**(имя_столбца) – функция, возвращающая среде значение указанного столбца. Может применяться только для *числовых* столбцов;
- **COUNT**(имя_столбца) – функция, возвращающая количество записей в указанном столбце, не содержащих **NULL**. Может применяться для *любых* столбцов (чисел, строк, дат и др.);
- **COUNT(*)** – вариант записи функции, возвращающий количество строк в таблице. При такой записи на эту функцию никак не влияют значения **NULL**, т. к. для нее не указывается столбец.

Например, определим среднее, минимальное и максимальное количество выполненных студентами лабораторных работ, а также общее количество сданных в группе отчетов (рис. 12):

```
SELECT AVG(Lab_Count), SUM(Lab_Count), MIN(Lab_Count),
       MAX(Lab_Count)
FROM Students;
```

AVG(Lab_Count)	SUM(Lab_Count)	MIN(Lab_Count)	MAX(Lab_Count)
1.6667	10	0	4

Рис. 12. Агрегатные функции

Стоит различать два варианта применения агрегатных функций:

1. Подсчет агрегатных функций ведется по всей таблице (как в примере выше), в запросе не применяется ключевое слово **GROUP BY**, результирующая таблица всегда состоит из одной строки. *При использовании данного варианта, все столбцы (перечисленные в **SELECT**) должны состоять только из агрегатных функций**!** Невозможно вывести столбцы без агрегатных функций (демонстрация чего будет представлена в раздел **5.2**).*
2. В запросе применена группировка через **GROUP BY**, подсчет агрегатных функций ведется для каждой из групп, количество строк в результирующей таблице равно количеству групп. *Можно вывести столбцы не только с агрегатными функциями (но также и те столбцы, по которым ведется группировка).*

Подробнее про использование агрегатных функций см. раздел **5.2** «Агрегатные функции и группировка» в Главе 5 «СЛОЖНЫЕ SQL-ЗАПРОСЫ».

13. Группировка

```
SELECT выводимый_столбец_1,
        АГРЕГАТ_ФУНКЦИЯ(выводимый_столбец_2), …
FROM имя_таблицы
WHERE условие_отбора
GROUP BY столбец_для_группировки
HAVING условие_группировки;
```

В качестве выводимых столбцов здесь можно использовать только столбцы, участвующие в группировке и агрегатные функции**!** Например, сгруппируем по количеству сданных работ, и найдем количество человек в каждой группе (рис. 13):

```
SELECT Lab_Count AS "Сдано работ", COUNT(*) AS "Кол-во человек"
FROM Students
GROUP BY Lab_Count;
```

Сдано работ	Кол-во человек
0	2
1	1
2	1
3	1
4	1

Рис. 13. Группировка

!!! В данном случае, нет разницы написать «COUNT(*)», или «COUNT(Lab_Count)» (или даже «COUNT(No)», или «COUNT(FIO)»), т.к. все поля исходной таблицы «Students» содержат значения (не содержат **NULL**). Но в случае наличия в таблице значения **NULL**, разные способы могут вернуть разные результаты**!**

Для группировки можно применить сразу несколько столбцов, например:

```
GROUP BY столбец_для_группировки_1, столбец_для_группировки_2, …
```

При группировке практически всегда используются агрегатные функции (**SUM**, **COUNT**, **MIN**, **MAX**, **AVG**).

Ключевое слово **HAVING** является необязательным. Подробнее про GROUP BY и HAVING см. раздел **5.2** «Агрегатные функции и группировка» в Главе 5 «СЛОЖНЫЕ SQL-ЗАПРОСЫ».

14. Объединение таблиц

До этого момента все манипуляции мы проводили только с одной таблицей. Для использования в запросе сразу нескольких таблиц необходимо обратить внимание на следующие отличия:

- во **FROM** нужно перечислить несколько таблиц:
  FROM имя_таблицы_1, имя_таблицы_2, …
- используются длинные имена столбцов. При использовании в запросе нескольких таблиц все имена столбцов должны записываться через точку в виде:
  имя_таблицы.имя_столбца_в_таблице
  В запросе, состоящем всего из одной таблицы, можно использовать только имя столбца, без указания имени самой таблицы.
  !!! *На самом деле, короткие имена можно использовать и в запросах, состоящих из нескольких таблиц, при условии, что эти имена уникальны (т. е. присутствуют только в одной таблице). Но в студенческих работах использование длинных имен является обязательным для всех запросов, состоящих из нескольких таблиц!*
- в **WHERE** прописываются связь таблиц. Для объединения таблиц одним из условий в **WHERE** должна выступать запись следующего вида:
  таблица_1.имя_столбца = таблица_2.имя_столбца
  Как правило, данное «имя_столбца» в обеих таблицах имеет одно и тоже название. Но это не является обязательным.
  Для объединения условий используются логические операторы, в данном случае логическое умножение **AND**.

В общем виде запрос к нескольким таблицам можно записать следующим образом:

SELECT имя_таблицы.столбец_1, имя_таблицы.столбец_2, …
FROM имя_таблицы_1, имя_таблицы_2, …
WHERE связи_таблиц **AND** условие_отбора;

Также связь таблиц может быть реализована не через **WHERE**, а через оператор **JOIN** (имеющий варианты **INNER JOIN**, **LEFT JOIN** и **RIGHT JOIN**). **JOIN** используется совместно с оператором **ON**, в котором и указывается связь столбцов таблиц (в том же виде, как это делается в **WHERE**):

ON таблица1.имя_столбца = таблица2.имя_столбца

Подробнее про INNER JOIN, LEFT JOIN и RIGHT JOIN см. раздел **5.3** «Объединение таблиц (JOIN)» в Главе 5 «СЛОЖНЫЕ SQL-ЗАПРОСЫ».

15. Полная структура запроса SELECT

Соберем воедино все ключевые слова языка **SQL** перечисленные выше, тогда получаем:

```
SELECT выводимый_столбец1 AS "Новое имя 1",      /* SUM, COUNT, MIN, MAX, */
       выводимый_столбец2 AS "Новое имя 2", …  /* AVG, COALESCE, … */
FROM имена_таблиц     /* INNER JOIN, LEFT JOIN, RIGHT JOIN */
ON связи_таблиц_через_join
WHERE связи_таблиц_без_join AND условие_отбора  /* =, >, <, <>, >=, <=, BETWEEN,
                                                   IN, LIKE, NOT, AND, OR */
GROUP BY столбец_группировки
HAVING условие_группировки  /* Тоже что для WHERE, плюс
                               SUM, COUNT, MIN, MAX, AVG */
ORDER BY столбец_сортировки [ASC|DESC]  /* SUM, COUNT, MIN, MAX, AVG */
LIMIT 20 OFFSET 0;
```

Использованные выше ключевые слова LIMIT и OFFSET, а также функция COALESCE ранее еще не упоминались, о них речь пойдет только в Главе 5 «СЛОЖНЫЕ SQL-ЗАПРОСЫ».

Справа от некоторых строк в виде комментария /* */ перечислены ключевые слова, которые могут применяться в этой строке.

Большинство из перечисленных строк не являются обязательными, и, следовательно, могут не использоваться в тех или иных запросах. Обязательными здесь являются только ключевые слова **SELECT** и **FROM**.

16. Арифметические операции

Язык **SQL** поддерживает арифметические операции, представленные в табл. 3.

Таблица 3 – Арифметические операции

Операция	Описание
+	Сложение
-	Вычитание
*****	Умножение
/	Деление
%	Остаток от деления (деление по модулю), например: 18 % 5 = 3

*На самом деле, для записи запроса **SELECT** обязательным не является даже оператор **FROM**. Такой запрос будет выполняться без обращения к какой-либо таблице. Хотя подобные («истинно однострочные») запросы годятся разве что для проверки работы операторов и функций в учебных целях, и не применяются в реальной работе.* Проверим работу арифметических операций (рис. 14):

```
SELECT 18+5, 18-5, 18*5, 18/5, 18%5;
```

18+5	18-5	18*5	18/5	18%5
23	13	90	3.6000	3

Рис. 14. Арифметические операции

Арифметические операции, как ясно из названия, работают с числами и не работают со строковыми значениями.

В отличие от операторов сравнения или логических операторов, которые используются только в условиях **WHERE** (или **HAVING**), арифметические операции могут применяться практически в любом месте запроса (любом, где вообще имеет смысл написать число, например там, где корректно будет написать числовую константу 5).

Арифметические операции могут применяться в следующих местах запросов:

- в условиях **WHERE** (или **HAVING**), например:
  WHERE a > 2*b + 1;
  где a и b – имена сравниваемых полей (столбцов);

- в списке выводимых столбцов (в первой строке, после оператора **SELECT**). Например, рассчитаем сколько лабораторных работ (из 4-х) осталось выполнить каждому студенту (рис. 15):

  SELECT FIO **AS** "Ф.И.О.", 4-Lab_Count **AS** "Осталось сдать работ"
  FROM Students;

Ф.И.О.	Осталось сдать работ
Иванов Иван Иванович	2
Петров Петр Петрович	4
Сидоров Сидор Сидорович	3
Николаев Николай Николаевич	4
Синицын Синиц Синицевич	1
Новиков Александр Игоревич	0

Рис. 15. Расчетный столбец

Здесь мы как бы создали «виртуальный» столбец, называющийся «Осталось сдать работ», который был рассчитан из реального столбца Lab_Count («Сдано работ») и константы 4.

Далее имеет смысл не выводить в списке должников тех, кому осталось сдать 0 работ, а также вывести самых злостных должников в начало списка.

Доработаем, соответственно, предыдущий запрос (рис. 16):

```
SELECT FIO AS "Ф.И.О.", 4-Lab_Count AS "Осталось сдать работ"
FROM Students
WHERE 4-Lab_Count > 0
ORDER BY 4-Lab_Count DESC;
```

Ф.И.О.	Осталось сдать работ
Петров Петр Петрович	4
Николаев Николай Николаевич	4
Сидоров Сидор Сидорович	3
Иванов Иван Иванович	2
Синицын Синиц Синицевич	1

Рис. 16. Список должников

Здесь мы задействовали арифметические операции не только в **SELECT**, но и в **WHERE** и **ORDER BY**.

Операции производятся не обязательно со столбцом и константой, можно, например, сложить значения из нескольких столбцов:

```
SELECT a+b+c AS "Сумма"
```

где a, b и c – имена полей (столбцов);

- также арифметические операции могут применяться в строке **SET** запроса на обновление **UPDATE**;
- и др.

17. Обновление (изменение) данных в таблице

```
UPDATE имя_таблицы
SET обновляемый_столбец = новое_значение
WHERE условие_отбора;
```

Обычно имя обновляемого столбца присутствует в запросах на обновление дважды:

- Оба раза в **SET** (до и после равенства). Например, студент Иванов сдал еще одну лабораторную работу (рис. 17):

```
UPDATE students
SET Lab_Count = Lab_Count + 1   /* Увеличить на единицу */
WHERE FIO LIKE 'Иванов %';
```

No	FIO	Lab_Count
1	Иванов Иван Иванович	3
2	Петров Петр Петрович	0
3	Сидоров Сидор Сидорович	1
4	Николаев Николай Николаевич	0
5	Синицын Синиц Синицевич	3
6	Новиков Александр Игоревич	4

Рис. 17. Увеличение значения

- Один раз в **SET** (до равенства), а второй раз в условии **WHERE**:

  UPDATE имя_таблицы
  SET обновляемый_столбец = новое_значение
  WHERE обновляемый_столбец = старое_значение;

  Например, требуется заменить в некоторой таблице «markets» все записи «СПб» на «Санкт-Петербург»:

  UPDATE markets
  SET City = 'Санкт-Петербург'
  WHERE City = 'СПб'

18. Удаление части записей (строк) из таблицы

DELETE
FROM имя_таблицы
WHERE условие_отбора

Если необходимо удалить часть строк из таблицы 1, связанной при этом с таблицей 2 (из которой ничего не удаляется), то:

DELETE имя_таблицы_1
FROM имя_таблицы_1, имя_таблицы_2
WHERE связи_таблиц AND условие_отбора

19. Удаление таблицы

DROP TABLE имя_таблицы;

!!! *В **MySQL** и **Microsoft SQL Server** очистка содержимого таблицы перед ее удалением не требуется. Хотя в некоторых других СУБД может понадобиться выполнить вначале **DELETE**, и только потом **DROP**.*

1.2. Система управления базами данных Microsoft Access

Microsoft Access (или полностью, **Microsoft Office Access**)[3] – реляционная система управления базами данных (СУБД, РСУБД) корпорации **Microsoft**. Входит в состав пакета **Microsoft Office** и является проприетарным (т.е. платным) программным обеспечением. Благодаря встроенному языку **VBA** позволяет разрабатывать экранные формы и целые приложения для работы с базами данных. Язык **SQL** в Microsoft Access не соответствует стандартам **ANSI** или **ISO**.

Microsoft Access предназначен для работы с базами данных на локальном компьютере (один пользователь) или в качестве файл-серверной СУБД (небольшое число пользователей, работающих с базой данных одновременно) и не рассчитан на многопользовательский (клиент-серверный) режим работы.

У корпорации Microsoft есть и другая СУБД – это **Microsoft SQL Server**, которая соответствует стандартам **ISO** и **ANSI** и позволяет работать в многопользовательском режиме.

Стоит обратить внимание на особенности сохранения в **Microsoft Access**, которое сильно отличается от сохранения в других программах **Microsoft Office** (таких как **Word** или **Excel**). В **Access** вносимые изменения сохраняются постоянно, без нажатия кнопки «Сохранить». Кроме того, имя базы данных и папка для ее сохранения задаются сразу в момент создания базы данных (рис. 18).

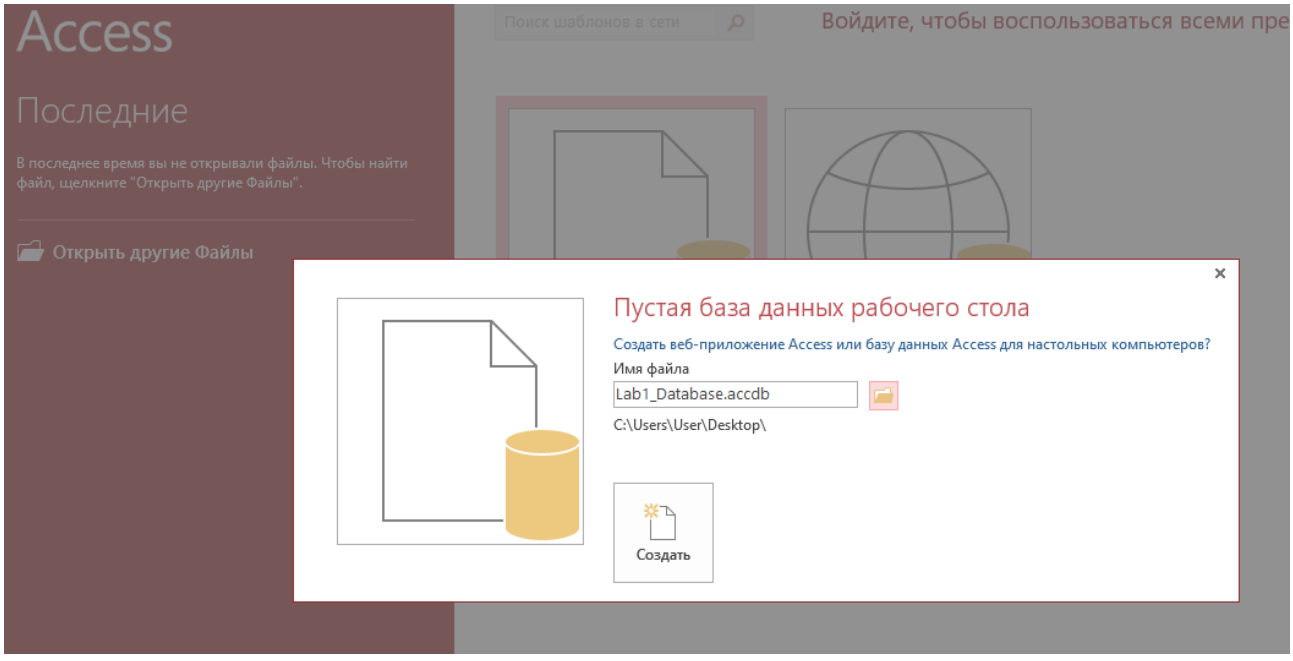


Рис. 18. Создание базы данных в Microsoft Access

!!! В процессе работы, даже если в базу не вносятся новые данные, размер файла **Access** постоянно увеличивается. Для сокращения объема памяти, занимаемого базой данных, необходимо периодически выполнять команду «**Сжать или восстановить базу данных**» (рис. 19) из меню «Работа с базами данных».

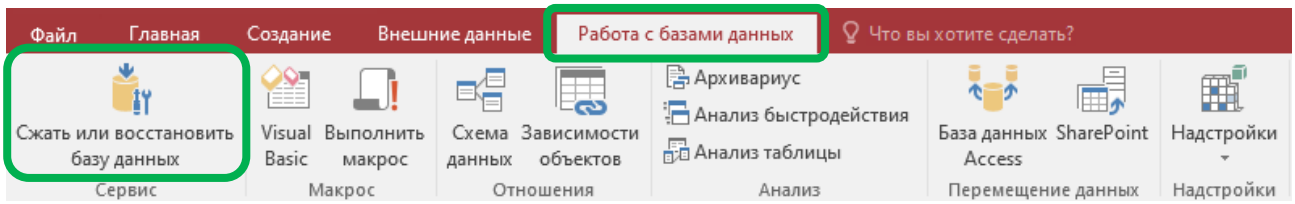


Рис. 19. Сжатие базы данных Microsoft Access

1.3. Создание таблиц в Access

Создание таблицы в **Access** происходит из меню «Создание» (рис. 20).

!!! В связи с тем, что предметом изучения является **SQL**, а не **Access**, здесь рассматривается только вариант создания простейших таблиц и не рассматривается работа с «Конструктором таблиц».

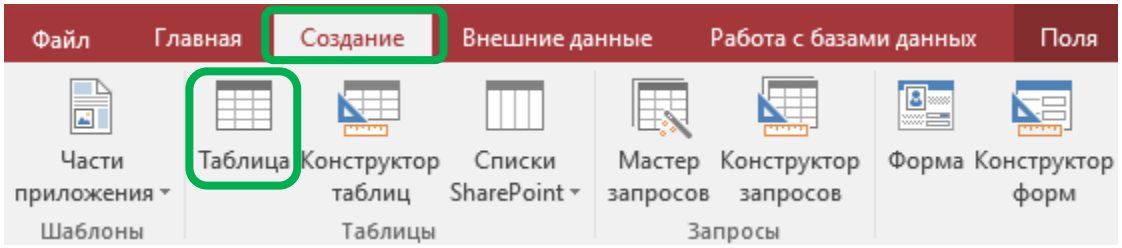


Рис. 20. Создание таблицы в Microsoft Access

После появления «пустой» таблицы в нее необходимо добавить требуемые столбцы (колонки). Это производится из выпадающего списка (рис. 21) в конце таблицы. В выпадающем списке требуется выбрать тип данных для добавляемого столбца (текст, число, логический и т. д.).

Создадим для примера таблицу (рис. 22), содержащую три текстовых столбца: «Фамилия», «Имя» и «Отчество». Также в таблице должен присутствовать числовой столбец «Код», который является первичным ключом (обычно создается автоматически, вместе с созданием новой таблицы).

Созданную таблицу необходимо заполнить не менее чем 10-ю строками с фамилиями, именами и отчествами (ФИО) студентов. Для этого можно используются ФИО ваших реальных одногруппников (если их меньше, то оставшиеся ФИО нужно придумать). В этой таблице фамилии располагаются не по алфавиту.

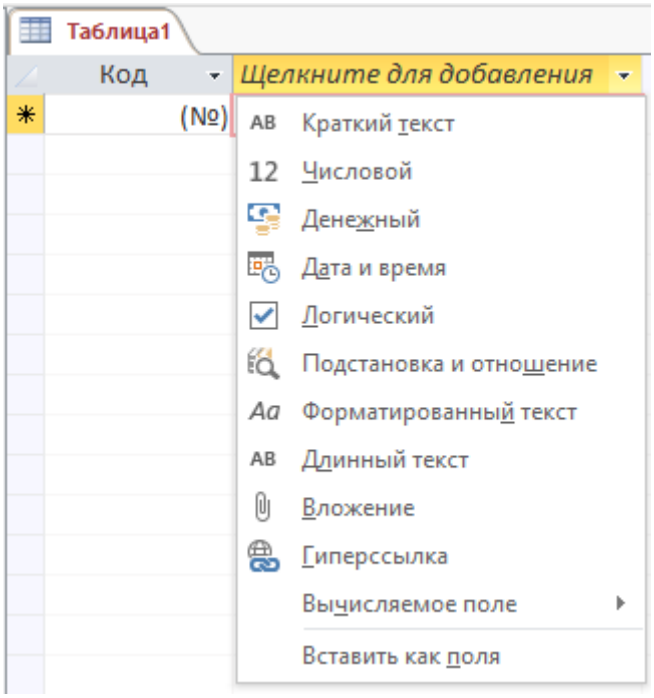


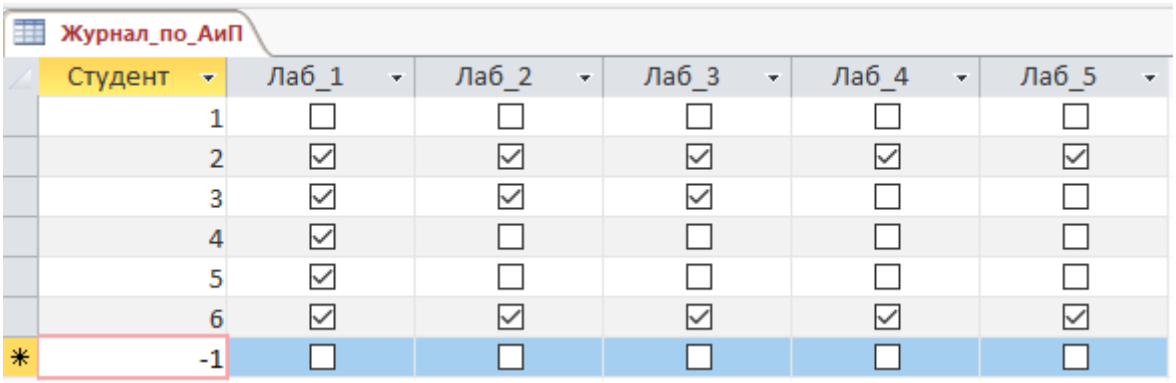
Рис. 21. Добавление столбцов в таблицу Microsoft Access

Студенты

Код	Фамилия	Имя	Отчество	Щелкните для добавления
1	Иванов	Иван	Иванович	
2	Петров	Петр	Петрович	
3	Сидоров	Сидор	Сидорович	
4	Николаев	Николай	Николаевич	
5	Александрова	Александра	Александровна	
6	Новиков	Александр	Игоревич	
(№)				

Рис. 22. Пример таблицы «Студенты»

Далее создадим таблицу «Журнал_по_АиП» (рис. 23), содержащую числовой столбец «Студент» (с кодом студента из прошлой таблицы), а также 5 логических столбцов («Лаб_1» .. «Лаб_5»), в которых будет отмечаться выполнение студентами лабораторных работ по дисциплине «СУБД». В данной таблице должно быть такое же количество строк, как и в предыдущей.

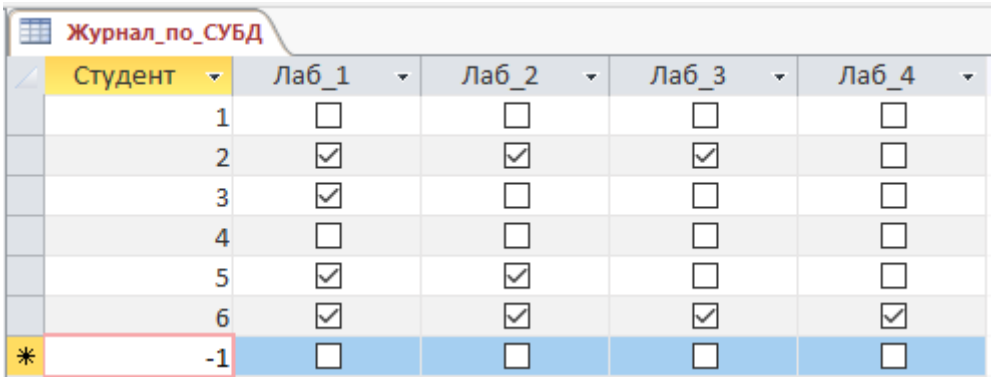


Журнал_по_АиП

Студент	Лаб_1	Лаб_2	Лаб_3	Лаб_4	Лаб_5
1	☐	☐	☐	☐	☐
2	☑	☑	☑	☑	☑
3	☑	☑	☑	☐	☐
4	☑	☐	☐	☐	☐
5	☑	☐	☐	☐	☐
6	☑	☑	☑	☑	☑
* -1	☐	☐	☐	☐	☐

Рис. 23. Пример таблицы «Журнал_по_АиП»

Аналогично создадим таблицу «Журнал_по_СУБД» (рис. 24), но содержащую 4, а не 5 лабораторных работ.



Журнал_по_СУБД

Студент	Лаб_1	Лаб_2	Лаб_3	Лаб_4
1	☐	☐	☐	☐
2	☑	☑	☑	☐
3	☑	☐	☐	☐
4	☐	☐	☐	☐
5	☑	☑	☐	☐
6	☑	☑	☑	☑
* -1	☐	☐	☐	☐

Рис. 24. Пример таблицы «Журнал_по_СУБД»

В итоге в **Microsoft Access**, мы получили 3 таблицы (рис. 25).

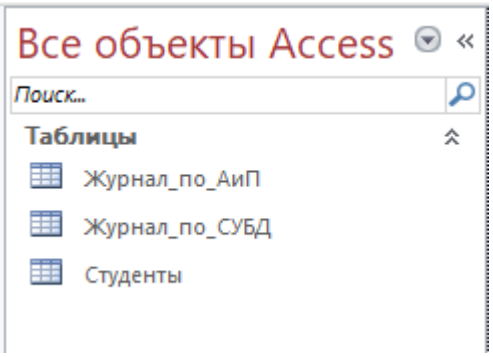


Рис. 25. Список таблиц в Microsoft Access

1.4. SQL-запросы в Access

Создание **SQL**-запросов в **Access** происходит через «**Конструктор запросов**» из меню «Создание» (рис. 26).

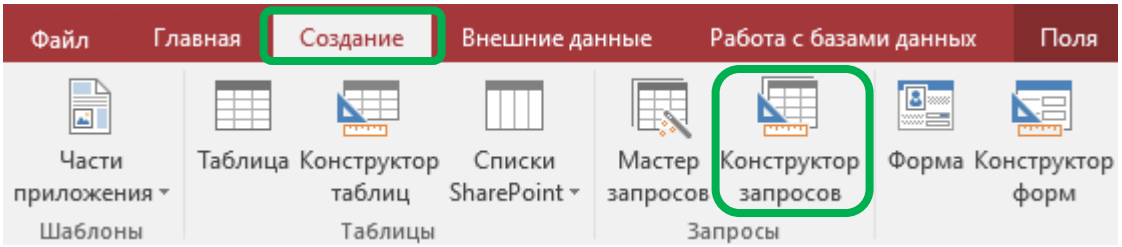


Рис. 26. Создание запроса в Microsoft Access

В появившемся далее окне, необходимо нажать кнопку «Закрыть» (рис. 27).

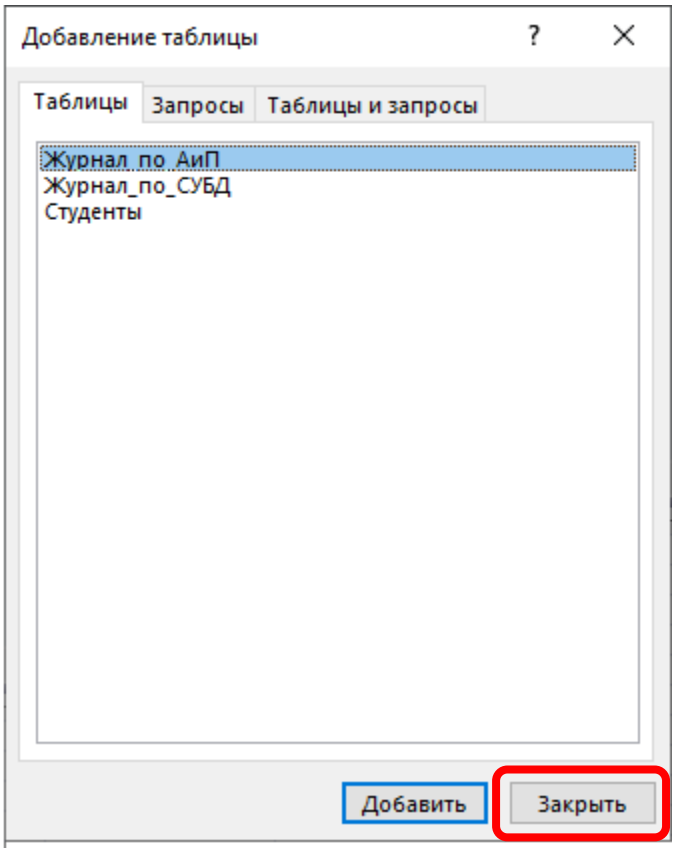


Рис. 27. Добавление таблиц к запросу

Далее перейдем в «**режим SQL**» (рис. 28).

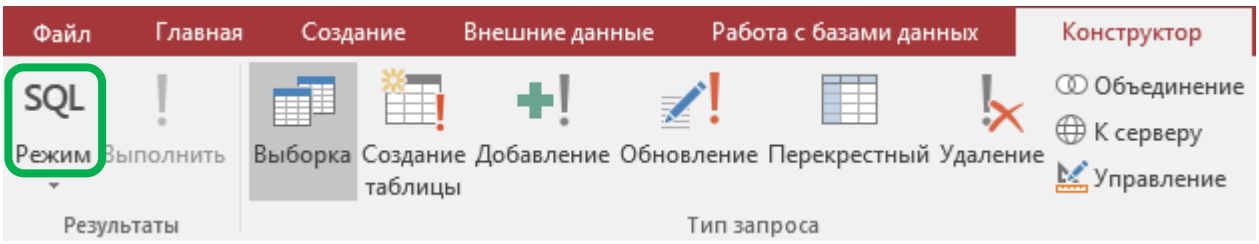


Рис. 28. Переход в режим SQL

Здесь введем **SQL**-запрос, представленный на рис. 29. Данный запрос позволяет нам вывести (рис. 30) содержание сразу двух таблиц. Этот запрос мы назовем «1) Журнал по АиП (с фамилиями)».

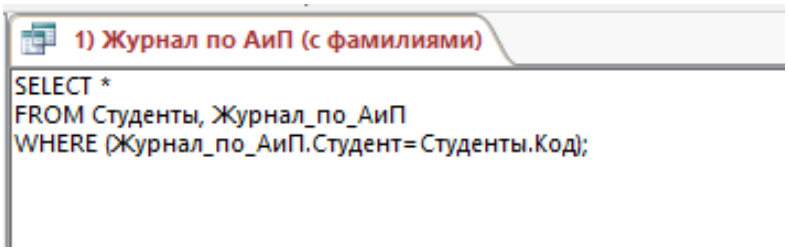


Рис. 29. Запроса «1) Журнал по АиП (с фамилиями)»

1) Журнал по АиП (с фамилиями)

Код	Фамилия	Имя	Отчество	Студент	Лаб_1	Лаб_2	Лаб_3	Лаб_4	Лаб_5
1	Иванов	Иван	Иванович	1	☐	☐	☐	☐	☐
2	Петров	Петр	Петрович	2	☑	☑	☑	☑	☑
3	Сидоров	Сидор	Сидорович	3	☑	☑	☑	☐	☐
4	Николаев	Николай	Николаевич	4	☑	☐	☐	☐	☐
5	Александрова	Александра	Александровна	5	☑	☐	☐	☐	☐
6	Новиков	Александр	Игоревич	6	☑	☑	☑	☑	☑

Рис. 30. Пример запроса «1) Журнал по АиП (с фамилиями)»

Недостатком выполненного ранее запроса является то, что были выведены все возможные колонки, в том числе повторяющиеся значения «Код» и «Студент». Создадим аналогичный запрос «2) Журнал по СУБД (с фамилиями)» (рис. 31), но без этих двух лишних колонок. Для этого в запросе вместо звездочки (*) необходимо перечислить имена всех отображаемых столбцов.

2) Журнал по СУБД (с фамилиями)

Фамилия	Имя	Отчество	Лаб_1	Лаб_2	Лаб_3	Лаб_4
Иванов	Иван	Иванович	☐	☐	☐	☐
Петров	Петр	Петрович	☑	☑	☑	☐
Сидоров	Сидор	Сидорович	☑	☐	☐	☐
Николаев	Николай	Николаевич	☐	☐	☐	☐
Александрова	Александра	Александровна	☑	☑	☐	☐
Новиков	Александр	Игоревич	☑	☑	☑	☑

Рис. 31. Пример запроса «2) Журнал по СУБД (с фамилиями)»

На данном этапе в **Microsoft Access** мы получили 3 таблицы и 2 запроса (рис. 32).

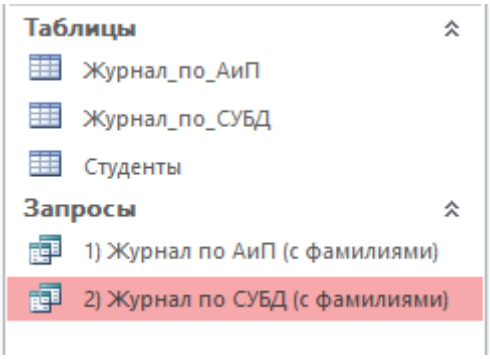


Рис. 32. Список объектов в Microsoft Access

1.5. ЛАБОРАТОРНАЯ РАБОТА № 1

«SQL-запросы в Access»

Необходимо выполнить работу в **Microsoft Access**, состоящую из 3 таблиц и 7 **SQL**-запросов (рис. 33). Создание таблиц описано в разделе **1.3**, создание первых двух запросов рассматривалось в разделе **1.4**.

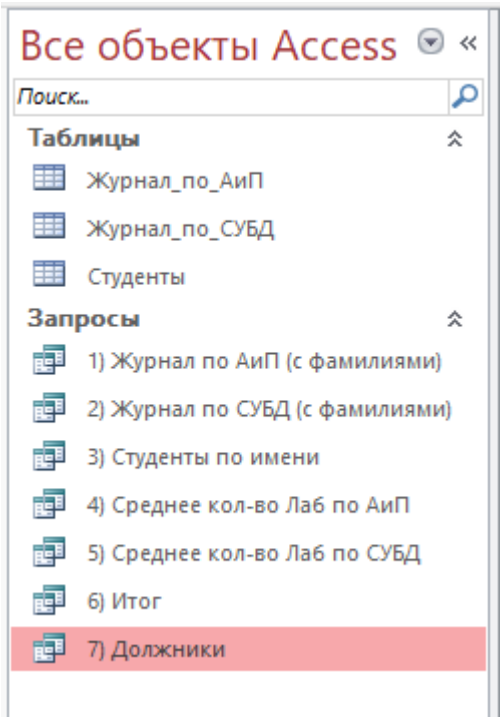


Рис. 33. Итоговый список объектов в Microsoft Access

Запрос «3) Студенты по имени» (рис. 34) выводит список всех студентов, чье имя начинается с заданных символов (в примере это все имена, начинающиеся с «Алекс»).

3) Студенты по имени

Код	Фамилия	Имя	Отчество
5	Александрова	Александра	Александровна
6	Новиков	Александр	Игоревич
(№)			

Рис. 34. Пример запроса «3) Студенты по имени»

Запрос «4) Среднее кол-во Лаб по АиП» (рис. 35) подсчитывает среднее количество лабораторных работ, выполненных студентами по дисциплине АиП, а также вычисляет процент выполненных работ (от общего числа работ, которое должна выполнить вся группа студентов). Здесь применяются агрегатные функции.

!!! Логические значения, как и числа, можно суммировать используя операцию сложение «+». Стоит обратить внимание, что в **Access** установленная галочка ☑ дает значение «-1», а не «+1», как может показаться на первый взгляд**!**

4) Среднее кол-во Лаб по АиП

АиП (в среднем Лаб)	АиП (% выполнения)
2,5	50%

Рис. 35. Пример запроса «4) Среднее кол-во Лаб по АиП»

Аналогично, запрос «5) Среднее кол-во Лаб по СУБД» (рис. 36) подсчитывает среднее количество лабораторных работ, выполненных студентами по дисциплине СУБД. Стоит обратить внимание, что в одном случае лабораторных было 4, а в другом 5.

5) Среднее кол-во Лаб по СУБД

СУБД (в среднем Лаб)	СУБД (% выполнения)
1,667	41,7%

Рис. 36. Пример запроса «5) Среднее кол-во Лаб по СУБД»

Запрос «6) Итог» (рис. 37) подсчитывает количество лабораторных работ, выполненных каждым студентом. Здесь для получения результата используются сразу все 3 таблицы.

6) Итог

Фамилия	Имя	Отчество	СУБД (сдано Лаб)	АиП (сдано Лаб)
Иванов	Иван	Иванович	0	0
Петров	Петр	Петрович	3	5
Сидоров	Сидор	Сидорович	1	3
Николаев	Николай	Николаевич	0	1
Александрова	Александра	Александровна	2	1
Новиков	Александр	Игоревич	4	5

Рис. 37. Пример запроса «6) Итог»

Запрос «7) Должники» (рис. 38), наоборот, выводит количество несданных лабораторных работ. Студенты, сдавшие все лабораторные работы, исключаются из таблицы. Список сортируется по фамилиям.

7) Должники

Фамилия	Имя	Отчество	СУБД (осталось сдать Лаб)	АиП (осталось сдать Лаб)
Александрова	Александра	Александровна	2	4
Иванов	Иван	Иванович	4	5
Николаев	Николай	Николаевич	4	4
Петров	Петр	Петрович	1	0
Сидоров	Сидор	Сидорович	3	2

Рис. 38. Пример запроса «7) Должники»

Отчет должен содержать:

- цель и задачи;
- описание хода работы (что, как и в какой программе делалось? Со скриншотами);
- скриншоты каждой из созданных таблиц (минимум 10 строк в каждой таблице);
- текст SQL-запросов (7 шт.) и скриншоты результатов;
- описание использованных команд SQL;
- титульный лист, номера страниц, оглавление, список использованной литературы (включая Интернет-ресурсы и данную методичку), выводы/заключение и т. п.

2. СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MySQL

MySQL – свободная реляционная система управления базами данных (СУБД, РСУБД)[4], одна из самых популярных реализаций языка и стандарта SQL. В настоящее время разработку и поддержку MySQL осуществляет корпорация **Oracle**. В 2008 году Sun Microsystems приобрела компанию MySQL AB (изначального разработчик MySQL) за 1 млрд долларов, в 2010 году Oracle приобрела Sun Microsystems за 7,4 млрд долларов. Продукт распространяется как под лицензией GNU GPL, так и под собственной коммерческой лицензией.

MySQL поддерживает множество платформ, в том числе работает на операционных системах Windows, Linux и MacOS. MySQL (как впрочем и SQL вообще) поддерживается многими языками программирования, например такими как Delphi, Си, Си++, Java, PHP. *Примеры работы с MySQL на языке PHP будут продемонстрированы далее (в Главе 4).*

MariaDB – ответвление от системы управления базами данных MySQL, разрабатываемое сообществом под лицензией GNU GPL. Разработку и поддержку MariaDB осуществляет компания MariaDB Corporation Ab и фонд MariaDB Foundation[5]. Создана в противовес политике лицензирования MySQL компанией Oracle. Ведущий разработчик MariaDB – Микаэль Видениус, автор оригинальной версии MySQL *(MySQL назван в честь его старшей дочери «Мю», а MariaDB в честь младшей – Марии)*.

MariaDB поддерживает высокую совместимость с MySQL. Несмотря на то, что у **MySQL** и **MariaDB** существует ряд серьезных отличий, в рамках данного курса мы будем считать их полностью идентичными, и говоря про MySQL, в равной мере будем подразумевать также и MariaDB. Но работать мы будем с MySQL.

2.1. Администрирование MySQL

Администрирование MySQL осуществляется при помощи программы PhpMyAdmin.

PhpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных[6]. Проект локализован на более чем 62 языках.

Для запуска **PhpMyAdmin** необходимо открыть любой браузер и ввести адрес **localhost** (или, что равносильно, адрес **127.0.0.1**). Данный адрес означает, что подключение производится к этому же компьютеру, т. е. серверная часть должна быть установлена на том же ПК. *Про установку серверной части подробнее рассказано в Главе 4, здесь же предполагается, что работа выполняется в компьютерном классе, где требуемое ПО уже установлено.*

После ввода адреса в браузере отобразится главная страница WAMP-сервера (рис. 39).

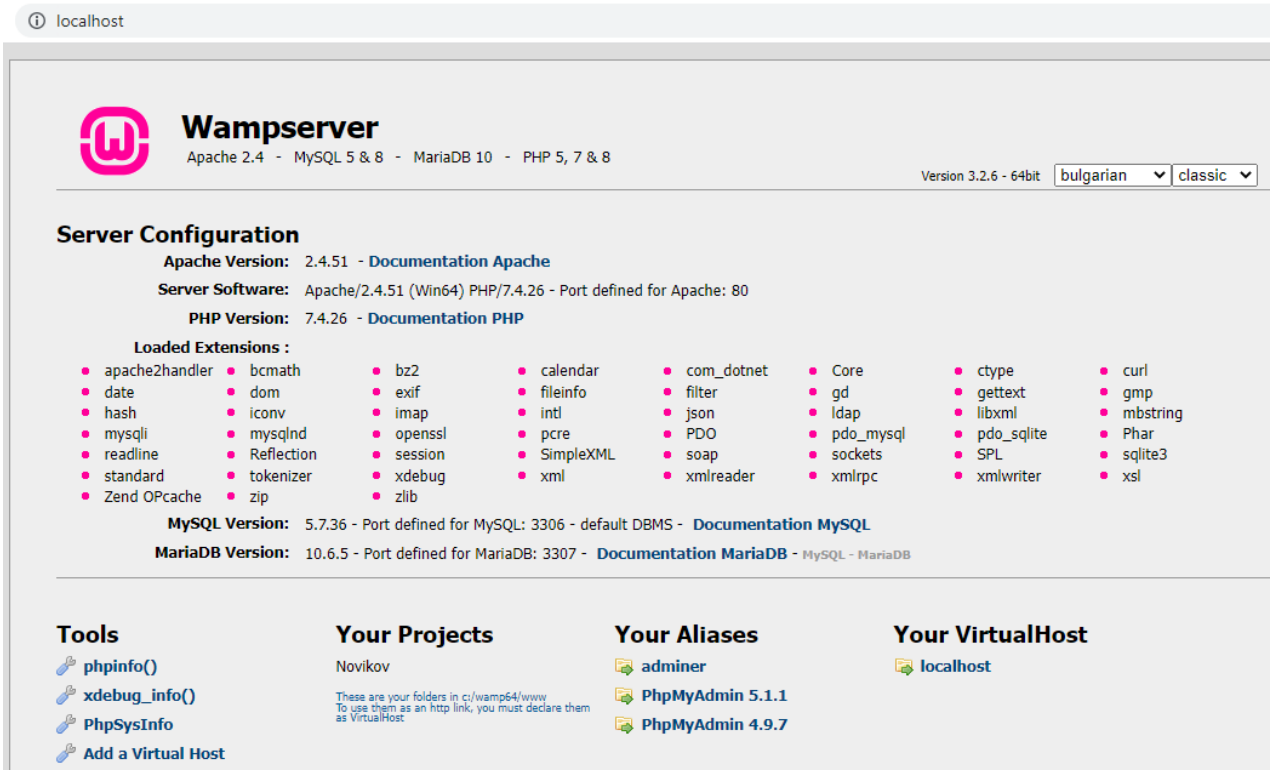


Рис. 39. Главная страница WAMP-сервера

Далее нужно перейти по ссылке **PhpMyAdmin** внизу страницы (рис. 40), если имеется несколько версий PhpMyAdmin, то лучше выбрать самую новую.

Рис. 40. Ссылки на администрирование СУБД

После чего мы попадаем в окно авторизации PhpMyAdmin (рис. 41), в котором нужно:

- указать имя пользователя **root**;
- поле для ввода пароля оставить пустым;
- выбрать базу данных (в нашем случае это **MySQL**, но здесь же можно было выбрать и **MariaDB**);
- нажать кнопку «**Вперёд**».

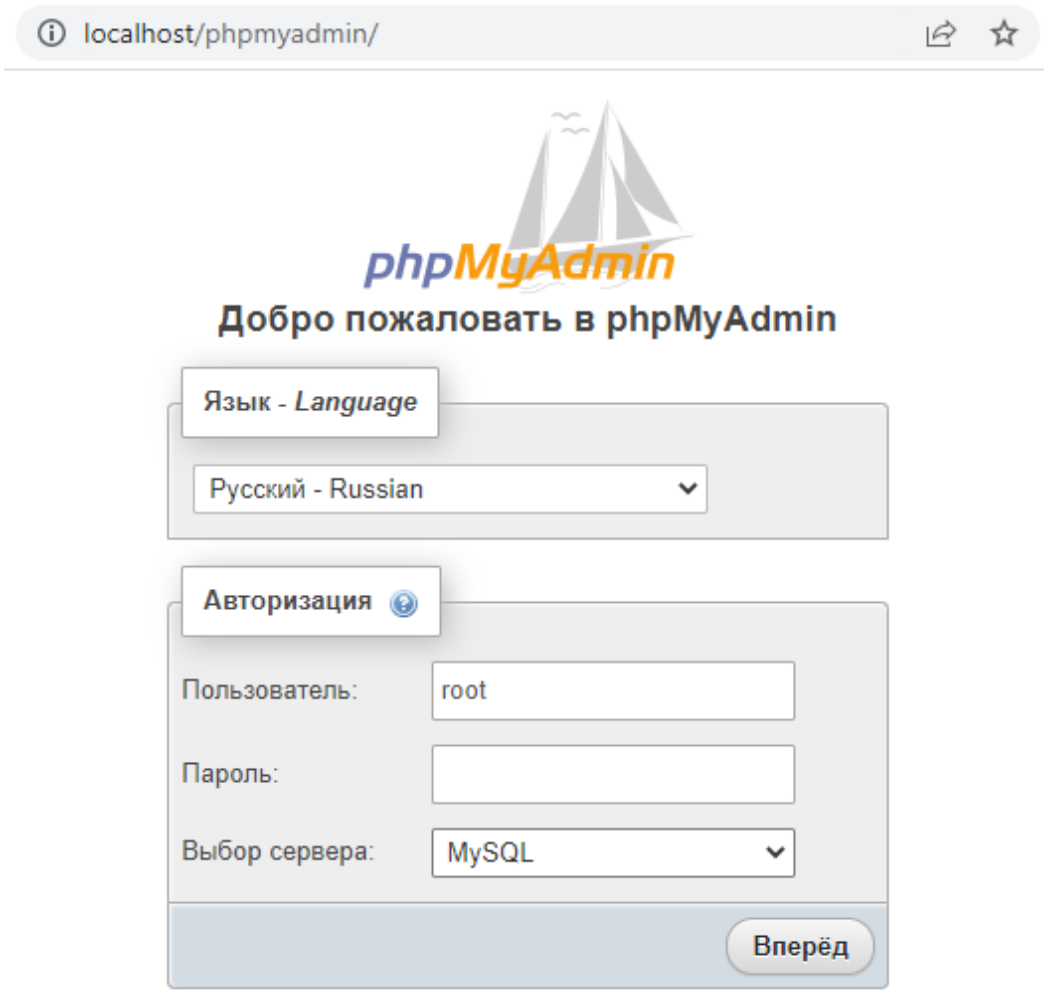


Рис. 41. Авторизации PhpMyAdmin

В результате мы попадаем на главную страницу PhpMyAdmin (рис. 42).

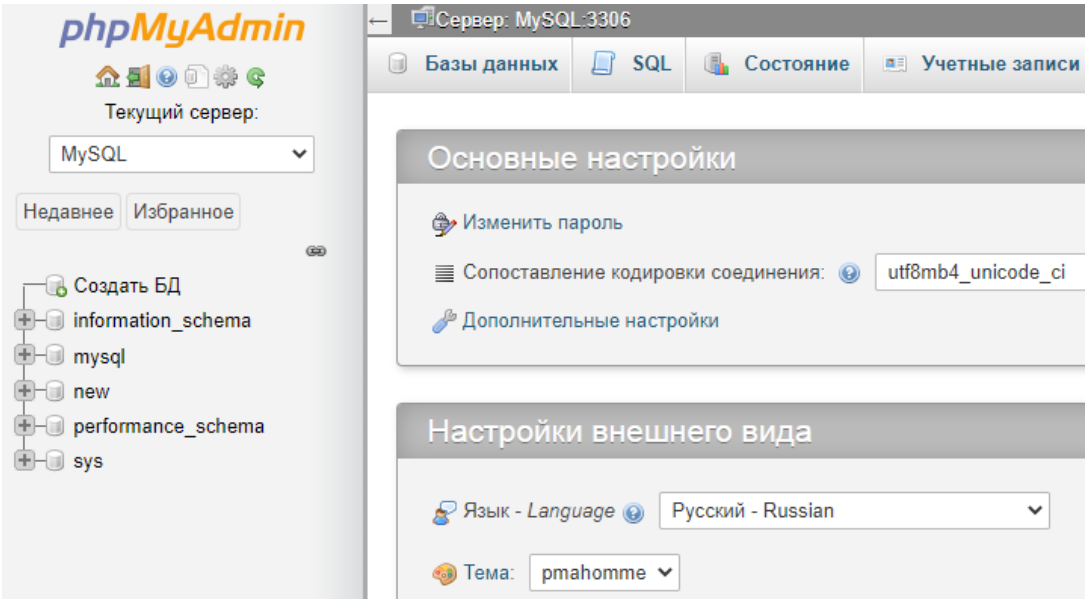


Рис. 42. Главная страница PhpMyAdmin

2.2. Создание базы данных

Выберем пункт «Создать БД» из меню слева (рис. 43).

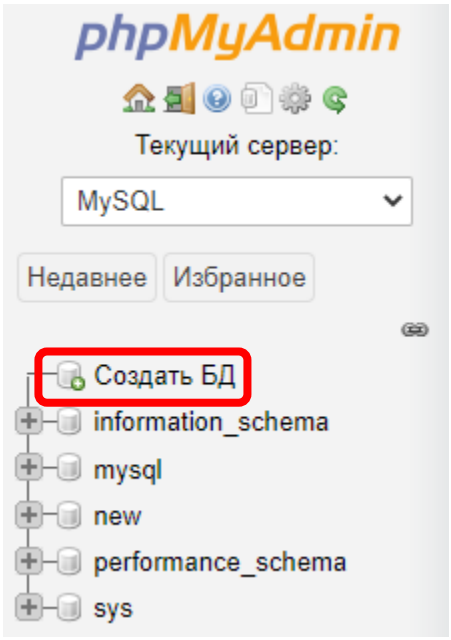


Рис. 43. Создание базы данных

Далее необходимо указать параметры (рис. 44) создаваемой базы данных: имя базы данных и кодировку символов.

!!! В качестве имени базы данных студенты указывают свою фамилию и текущий год! *В моих примерах это будет только фамилия «**Novikov**»*. Далее работа осуществляется только в этой базе, другие таблицы менять запрещено!

!!! В качестве кодировки следует выбрать «**cp1251**», которая поддерживает русские символы (символы Кириллицы).

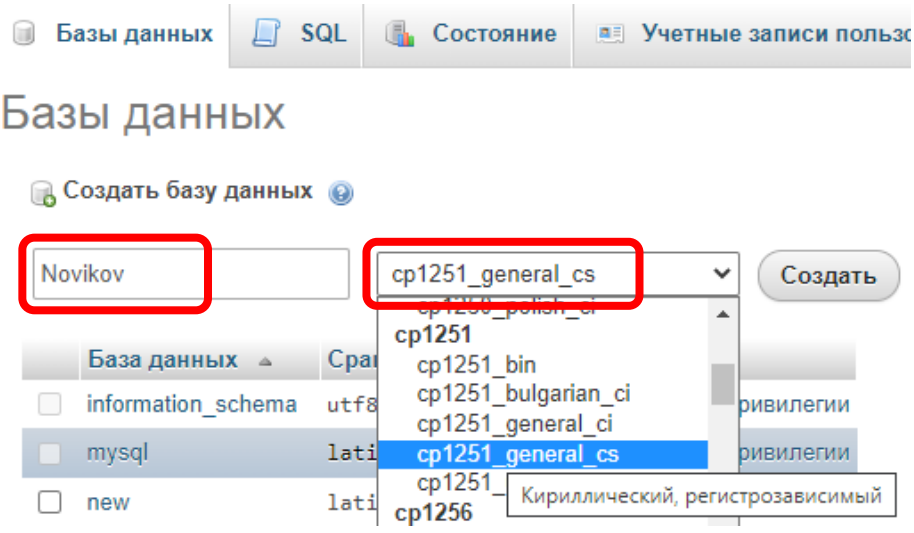


Рис. 44. Параметры создаваемой базы данных

После создания базы данных появится страница с предложением создать таблицу в новой базе данных (рис. 45). Но нас больше интересует создание таблиц с помощью запросов на SQL.

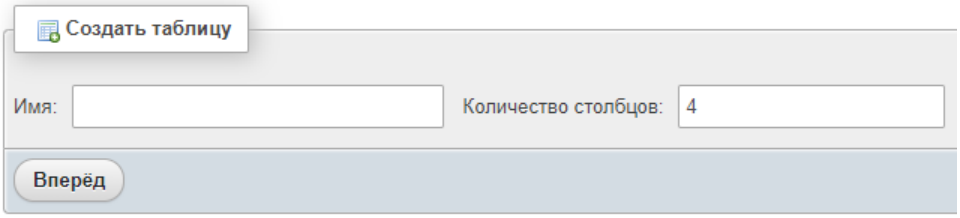


Рис. 45. Предложение создать таблицу

2.3. SQL-запросы в PhpMyAdmin

Для выполнения запросов на языке SQL, требуется перейти на вкладку «SQL» (рис. 46).

1. Создание таблицы

Для создания таблицы необходимо написать соответствующий SQL-запрос **CREATE** (рис. 46).

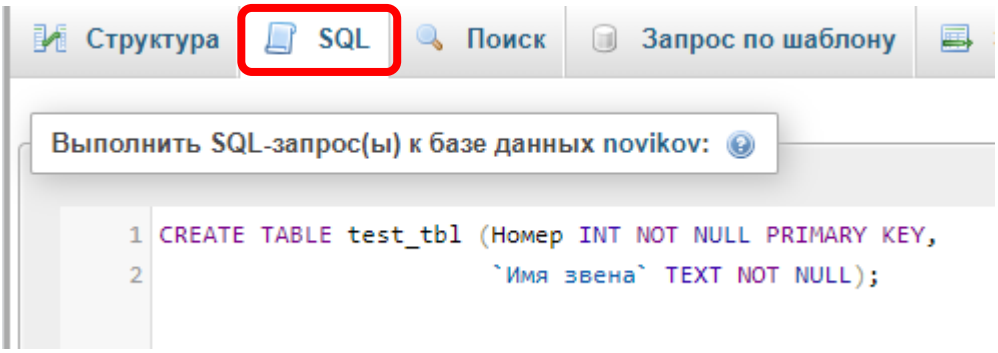


Рис. 46. Создание таблицы через SQL

В данном примере создается таблица с именем «**test_tbl**», имеющая два столбца, с именами «**Номер**» и «**Имя звена**» (рис. 47).

Т.к. имя второго столбца содержит пробел, его обязательно нужно писать в кавычках. При этом для имен используются «косые» кавычки, которые вводятся с клавиатуры клавишей «ё» в английской раскладке. Стоит обратить внимание, что для ввода текстовых значений используются обычные одинарные кавычки, вводимые клавишей «э» в английской раскладке. Имя первого столбца «Номер» и имя таблицы «test_tbl» также могли быть взяты в косые кавычки, но в данном случае это было не обязательно.

Первый столбец имеет тип «**INT**», т.е. в нем будут храниться целые числа (номера). Второй столбец имеет тип «**TEXT**». Для обоих столбцов указано ключевое слово «**NOT NULL**», которое говорит, что соответствующие поля обязательны для заполнения и не могут быть оставлены пустыми. Кроме того, для первого столбца также указано ключевое слово «**PRIMARY KEY**» (первичный ключ), говорящее, что значения в этом поле должны быть уникальными, т.е. не могут повторяться.

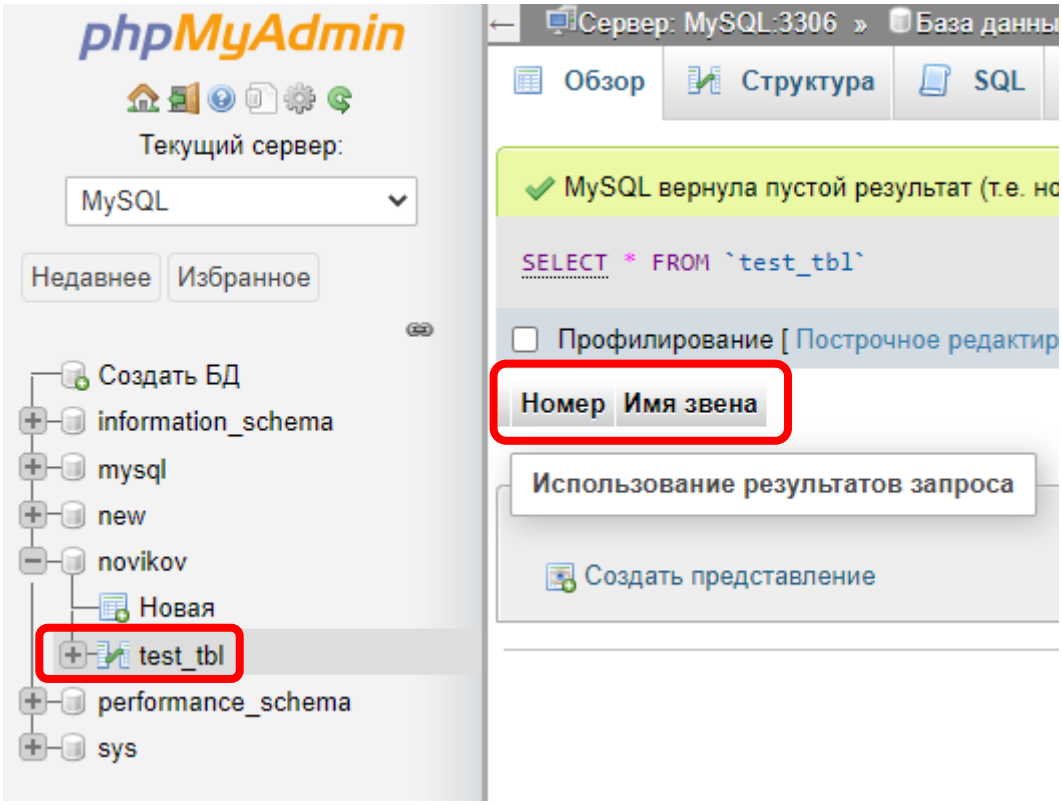


Рис. 47. Созданная таблица

2. Заполнение таблицы

Для заполнения таблицы строками, используется ключевое слово «**INSERT**» (рис. 48).

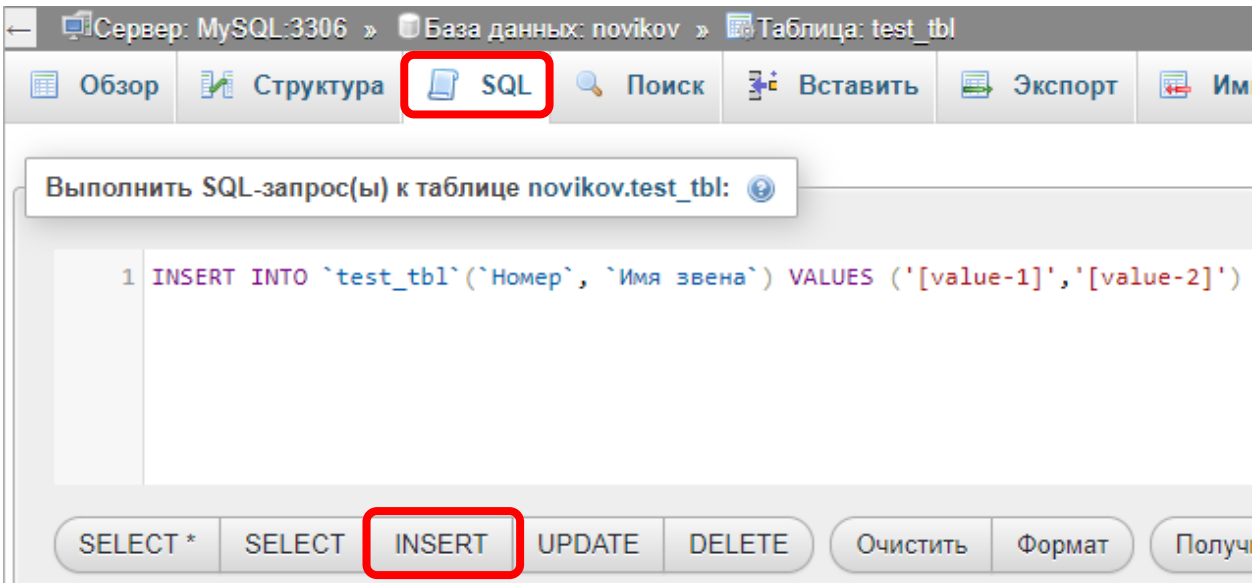


Рис. 48. Шаблон запроса

При этом **PhpMyAdmin** уже содержит удобные шаблоны для разных запросов, вызываемые нажатием соответствующей кнопки, в нашем случае кнопки «**INSERT**» (см. рис. 48), расположенной под полем ввода запроса.

Шаблон запроса уже содержит все имена столбцов из созданной ранее таблицы *(в нашем случае имена столбцов не понадобятся, т. к. мы заполняем все значения по порядку)*. Также в шаблон уже добавлены черновые значения в формате «**[value-1]**», вместо которых необходимо подставить свои значения.

Стоит отметить, что шаблон не содержит точки с запятой в конце запроса. **MySQL** допускает отсутствие точки с запятой в конце запроса, а если пишется сразу несколько запросов, то только в конце последнего из них.

!!! В студенческих работах использование точки с запятой обязательно в конце каждого запроса!

Исправим шаблон и скопируем его несколько раз для заполнения таблицы 5-ю строками (рис. 49).

Рис. 49. Добавление строк в таблицу

При этом, если цифры были введены в точности как на рис. 49, то мы получим сообщение об ошибке (рис. 50).

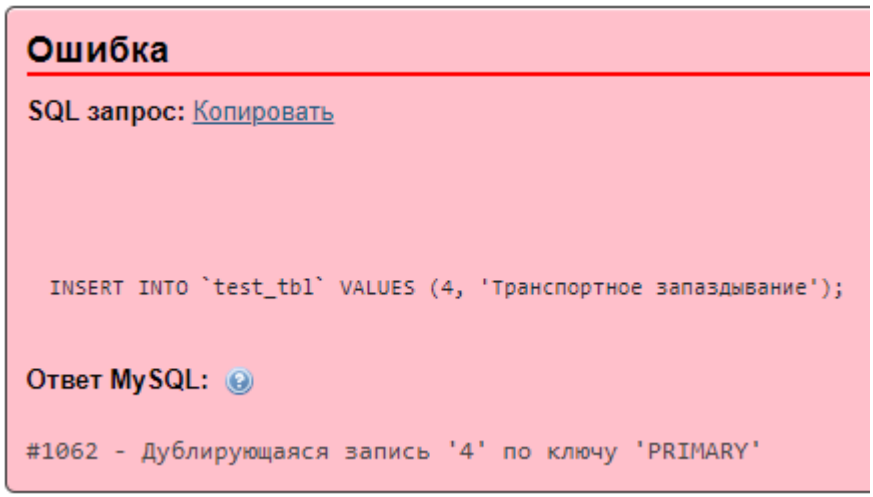


Рис. 50. Ошибка при добавлении строк в таблицу

Это связано с тем, что в уникальное поле («**PRIMARY KEY**») мы пытаемся записать два одинаковых номера 4. Перейдя на вкладку «Обзор» мы увидим, что в таблицу были добавлены только первые 4 строки (рис. 51).

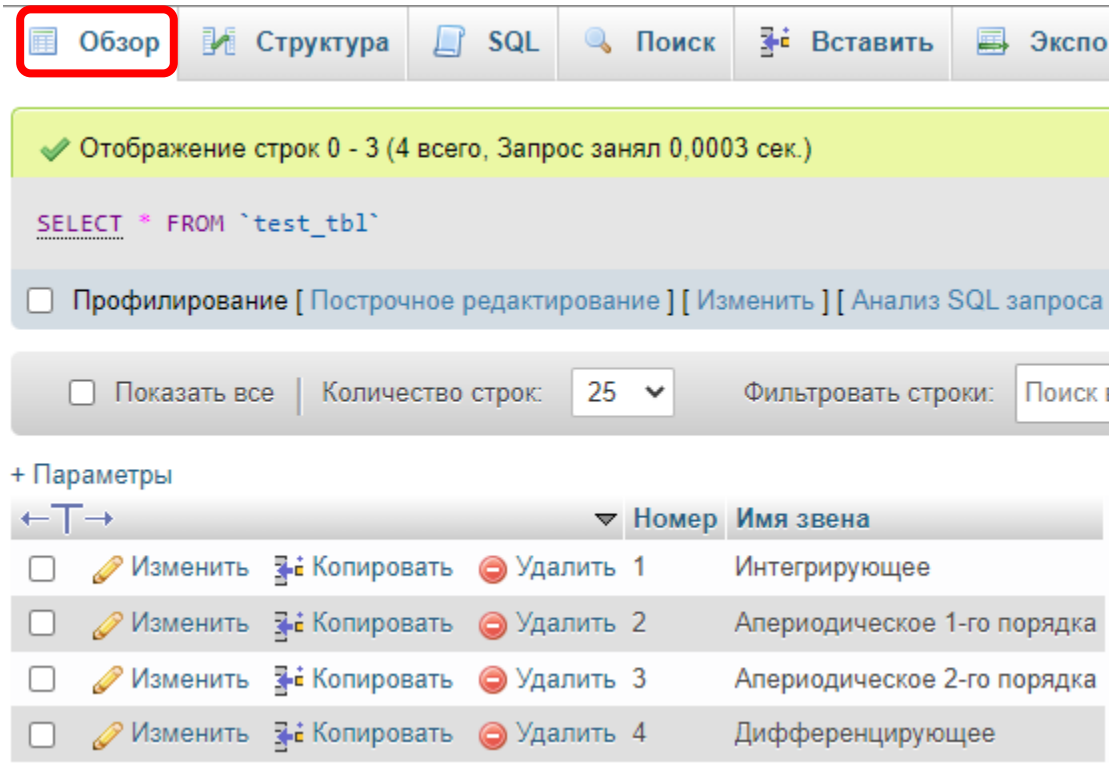


Рис. 51. Таблица с добавленными строками

3. Вывод таблицы

Как видно из рис. 51, вызов вкладки «Обзор» автоматически формирует запрос «**SELECT**» на вывод всего содержимого таблицы, в данном случае это запрос:

SELECT * **FROM** `test_tbl`

Если мы хотим выбрать только часть строк, то необходимо вновь перейти на вкладку «SQL» и ввести соответствующий запрос с **WHERE** (рис. 52).

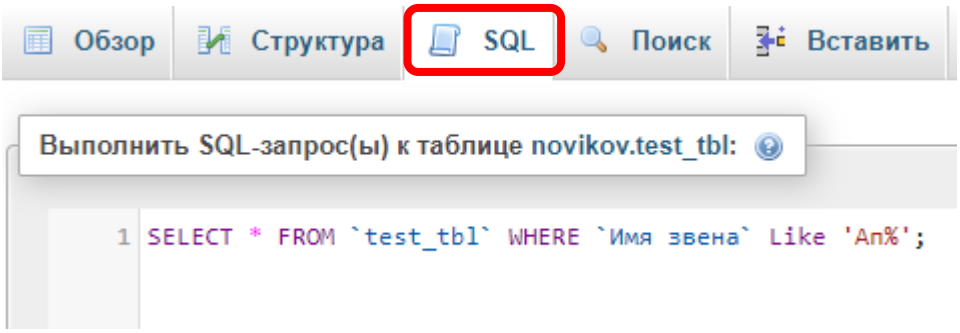


Рис. 52. Запрос SELECT с условием

В результате мы найдем 2 подходящие строки (рис. 53), из 4 исходных.

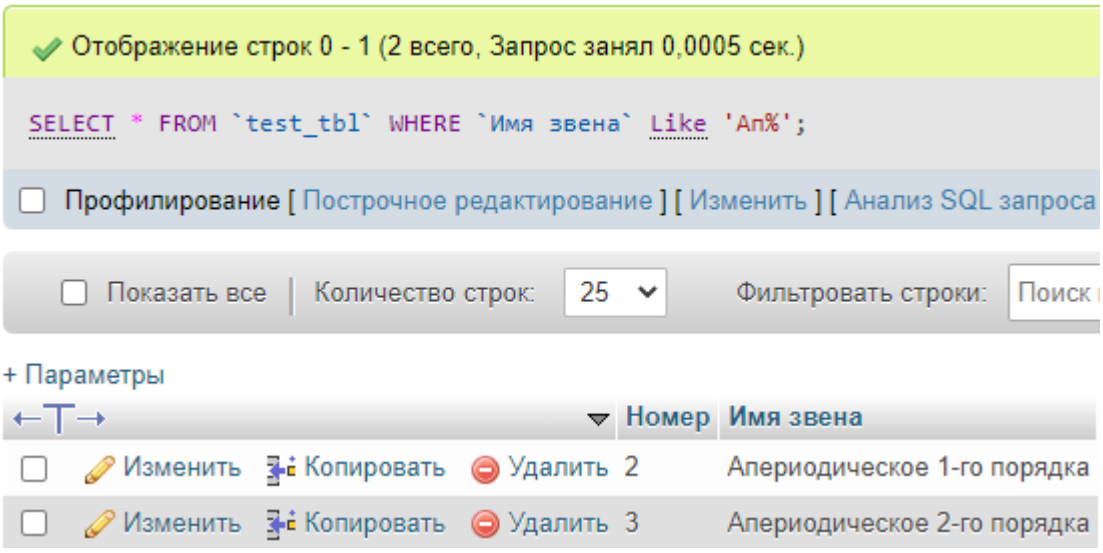


Рис. 53. Результат запроса SELECT

4. Удаление таблицы

Для удаления таблицы необходимо выполнить запрос «**DROP**» (рис. 54).

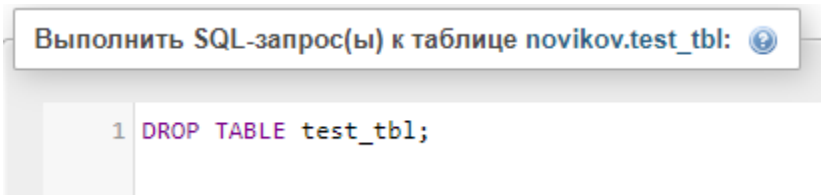


Рис. 54. Запрос на удаление таблицы

2.4. ЛАБОРАТОРНАЯ РАБОТА № 2

«Базы данных MySQL в PhpMyAdmin»

В данной лабораторной работе студенту предстоит выбрать **10** датчиков (или др. устройств), объединенных общей мыслью (например, 10 разных датчиков температуры), и внести их в таблицу, разработанную в **MySQL**.

Выполнение данной работы рассматривается на примере *датчиков температуры*. Студенту необходимо выбрать свой тип датчиков (или др. устройств), которыми он будет заполнять таблицы. При этом нельзя выбирать датчики температуры (т. к. они уже есть в примере), а также необходимо, чтобы у студентов в группе были выбраны разные типы устройств.

Оборудование можно найти, например, посетив сайты ведущих российских производителей оборудования для автоматизации, таких как: https://owen.ru/, https://segnetics.com/, https://www.mzta.ru/. *Не обязательно брать все устройства с одного сайта, и не обязательно пользоваться именно этими сайтами.*

!!! Необходимо выбрать оборудование, для которого имеются фотографии! Они еще понадобятся нам в дальнейшем (в главах 3 и 4).

Далее приводится далеко не полный перечень устройств, которые можно выбрать: датчики давления, датчики уровня, датчики влажности, программируемые контроллеры, модули для контроллеров, панели оператора (HMI), измерители-регуляторы (например, такие как Овен ТРМ1), реле и контакторы электромагнитные, твердотельные реле, программируемые реле, автоматические выключатели (автоматы), счетчики (на воду, на газ, на электричество – предпочтительно электронные, а не механические), частотные преобразователи, электроприводы заслонок и клапанов, насосы, блоки питания 24В, а также различное программное обеспечение для автоматизации (например, SCADA-системы).

Шапка таблицы с датчиками должна быть примерно следующей: наименование датчика, тип датчика, производитель, нижний предел измерения, верхний предел измерения, погрешность, выходной сигнал, описание, гарантия, цена, рейтинг *(последний можно придумать самостоятельно)*. Пример содержимого таблицы для выбранных датчиков температуры представлен в табл. 4 и на рис. 55. *Если выбраны не датчики, а другие устройства, то некоторые из перечисленных параметров могут быть к ним не применимы, но у них появятся свои параметры, например, количество и типы входных/выходных сигналов для контроллеров или модулей к контроллерам.*

Кроме того, необходимо придумать вторую таблицу (**8-10** строк), в которой содержится информация о складах, на которых хранятся данные датчики (или магазины, в которых их можно приобрести). В этой таблице должны быть отражены: название магазина, адрес магазина, телефон магазина, время работы магазина и т. п. Пример содержимого таблицы со складами/магазинами представлен на рис. 56.

Связь между этими двумя таблицами организована через третью таблицу, отрывок которой представлен на рис. 57. В этой таблице также содержится количество датчиков, имеющихся в наличии в указанном магазине. Необходимо предусмотреть варианты, при которых некоторые датчики имеются сразу в нескольких магазинах, другие только в одном магазине, а нескольких датчиков нет в наличии ни в одном магазине. В этой таблице должно быть не менее **15** строк.

Отчет должен содержать:

- цель и задачи;
- описание хода работы (что, как и в какой программе делалось? Со скриншотами);
- текст SQL-запросов на создание и заполнение таблиц (10 устройств) и скриншоты каждой из созданных таблиц (достаточно только части каждой таблицы);
- пример запроса Select, выводящего содержимое всех таблиц в одну (текст SQL-запроса и скриншот результата);
- расшифровку терминов, которые встречаются в отчете (такие как SQL, PhpMyAdmin и т. п.);
- описание использованных команд SQL;
- титульный лист, номера страниц, оглавление, список использованной литературы (включая Интернет-ресурсы и данную методичку), выводы/заключение и т. п.

Таблица 4 – Пример таблицы с датчиками

Номер	1	2
Наименование	ДТС015-PT1000.B2.200	ДТС035М-50М.1,0.120.МГ.И [3]
Тип	Термометр сопротивления	Термометр сопротивления
Изображение	dts015.png	dts035m.png
Рейтинг	4,7	4,9
Производитель	Овен	Овен
Страна	Россия	Россия
Гарантия, месяцев	24	24
Цена, руб	2490,00	7998,00
Описание	ДТСхх5 с коммутационной головкой позволяют измерят...	Датчики изготавливаются на базе термометров сопрот...
Диапазон измеряемых температур	–60…+500 °C	0…+150 °C
Погрешность	0,5%	1%
Выходной сигнал	Сопротивление (Pt1000)	4...20 мА
Схема подключения	Двухпроводная	Двухпроводная
Напряжение питания (номинальное)	Не требуется	24 В
Диапазон допустимых напряжений питания	Не требуется	12...36 В
Ссылка на документацию	https://owen.ru/uploads/292/kratkoe_rukovodstvo_dtshh5.pdf	https://owen.ru/uploads/324/re_oven_dts-i_1-ru-18399-1.10.pdf
Температура окружающей среды	–60…+85 °C	–40…+85 °C
Количество чувствительных элементов	1	1
Длина погружаемой части L, мм	200	120
Материал коммутационной головки	Пластмасса	Металл
Класс защиты	IP54	IP65
Среда измерения	Твердые, жидкие и газообразные среды (неагрессивны...)	Твердые, жидкие, газообразные и сыпучие среды
Сопротивление изоляции	не менее 100 МОм	
Средний срок службы	не менее 10 лет	не менее 10 лет
...	...	...

Номер	Наименование	Тип	Изображение	Рейтинг	Производитель	Страна	Гарантия	Цена
1	ДТС015-РТ1000.В2.200	Термометр сопротивления	dts015.png	4.7	Овен	Россия	24	2490
2	ДТС035М-50М.1,0.120.МГ.И [3]	Термометр сопротивления	dts035m.png	4.9	Овен	Россия	24	7998
3	ДТС035М-50М.1,0.120.И [3]	Термометр сопротивления	dts035m.png	4.9	Овен	Россия	24	7398

Рис. 55. Пример таблицы с датчиками

Номер	Город	Адрес	Телефон	E-mail
1	г. Санкт-Петербург	ул. Александра Матросова, д. 4, корп. 2, литер Д	+7 (812) 677-56-..	sales@овенэюя.рф
2	г. Екатеринбург	ул. Малышева, 164	+7 (343) 228-18-..	sale@uralenergoxyz.ru
3	г. Москва	ул. Кусковская, дом 20А, офис В706	+7 (495) 777-83-..	zakaz@e-xyz.ru
4	г. Санкт-Петербург	пр. Стачек, д. 41	+7(812) 628-28-..	info@owen-xyz.ru
5	г. Псков	улица Советская, дом 49, помещение 4	+7 (911) 157-32-..	info@owen-xyz.ru

Рис. 56. Пример таблицы со складами

Номер записи	Номер датчика	Количество	Номер магазина
1	1	400	1
2	1	123	2
3	2	2	2
4	2	3	3
5	2	7	4

Рис. 57. Пример связи таблиц

2.5. *Работа с таблицами средствами PhpMyAdmin

PhpMyAdmin позволяет создавать и удалять таблицы, а также изменять их содержимое, не прибегая к написанию запросов на языке SQL.

1. Создание таблицы

Для создания таблицы необходимо в базе данных со своей фамилией нажать кнопку «Новая» (рис. 58).

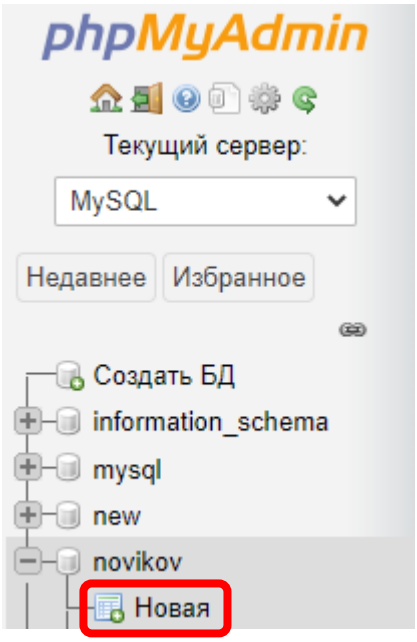


Рис. 58. Создание таблицы средствами PhpMyAdmin

В появившемся окне (рис. 59) требуется ввести имя создаваемой таблицы («test2_tbl») и указать имена столбцов (в нашем случае их три: «Номер», «Имя звена» и «Комментарий»), а также указать типы данных для этих столбцов.

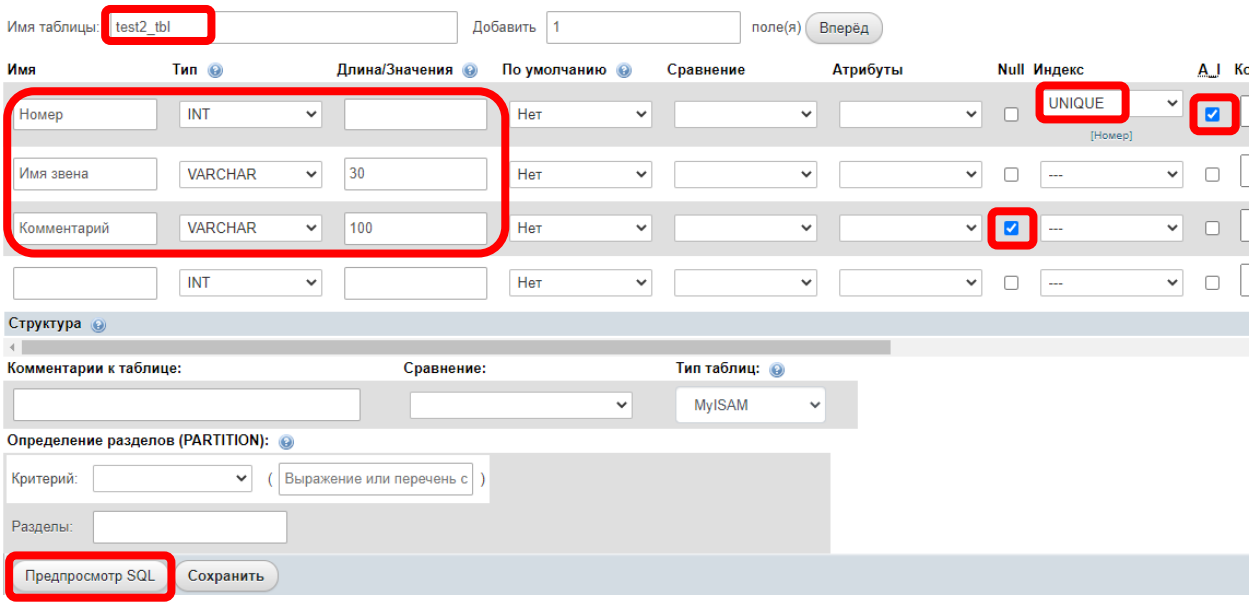


Рис. 59. Параметры создаваемой таблицы

Как и в прошлом примере, столбец «**Номер**» имеет тип «**Int**», т. е. будет целым числом. Для «**Имя звена**» и «**Комментарий**» укажем тип «**VarChar**», а не «**Text**» (как в прошлый раз). *Это связано с тем, что при добавлении строк в таблицу через PhpMyAdmin (о котором будет рассказано далее), тип Text будет восприниматься как многострочный текст, и поля для ввода будут занимать весь экран, а для VarChar мы получим компактные поля для ввода.* При этом VarChar также хранит текстовые значения. Для него необходимо задать максимальное количество символов, которое можно ввести в это поле (в нашем случае это будет «**30**» и «**100**»).

Для столбца «**Номер**» необходимо выбрать «**Unique**» («Уникальный») и поставить галочку «**Auto_Increment**» (**A_I**, «Автоматическое увеличение на единицу», которое в дальнейшем позволит не заполнять поле номера вручную). Для столбца «**Комментарий**» установим галочку «**Null**».

Перед тем как сохранить таблицу, нажмем кнопку «**Предпросмотр SQL**» и увидим автоматически сгенерированное содержимое SQL-запроса (рис. 60).

!!! В Лабораторных работах не принимаются запросы, сгенерированные автоматически, а соответствующая работа считается <u>невыполненной</u>!

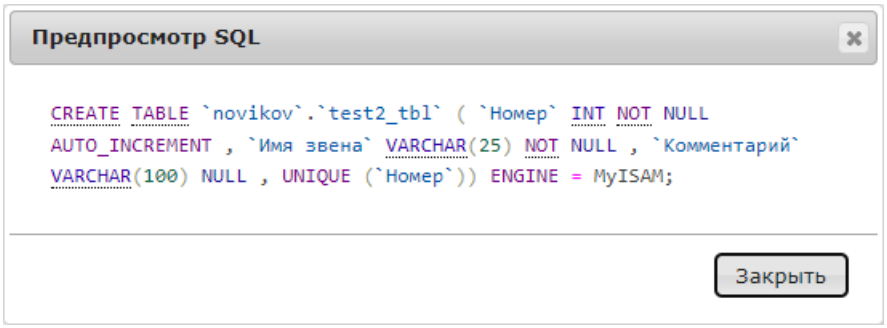


Рис. 60. Автоматически сгенерированный код SQL

2. Добавление строк в таблицу

Для добавления строк в таблицу необходимо перейти на вкладку «**Вставить**» (рис. 61). Здесь требуется заполнить «**Имя звена**» (в нашем случае это «Интегрирующее»). Т. к. для поля «**Номер**» мы ранее указали «**Auto_Increment**», то теперь его можно оставить пустым, оно будет заполнено автоматически. Можно заполнить сразу и вторую строку («Апериодическое 1-го порядка»), после чего необходимо нажать на любую из кнопок «**Вперед**».

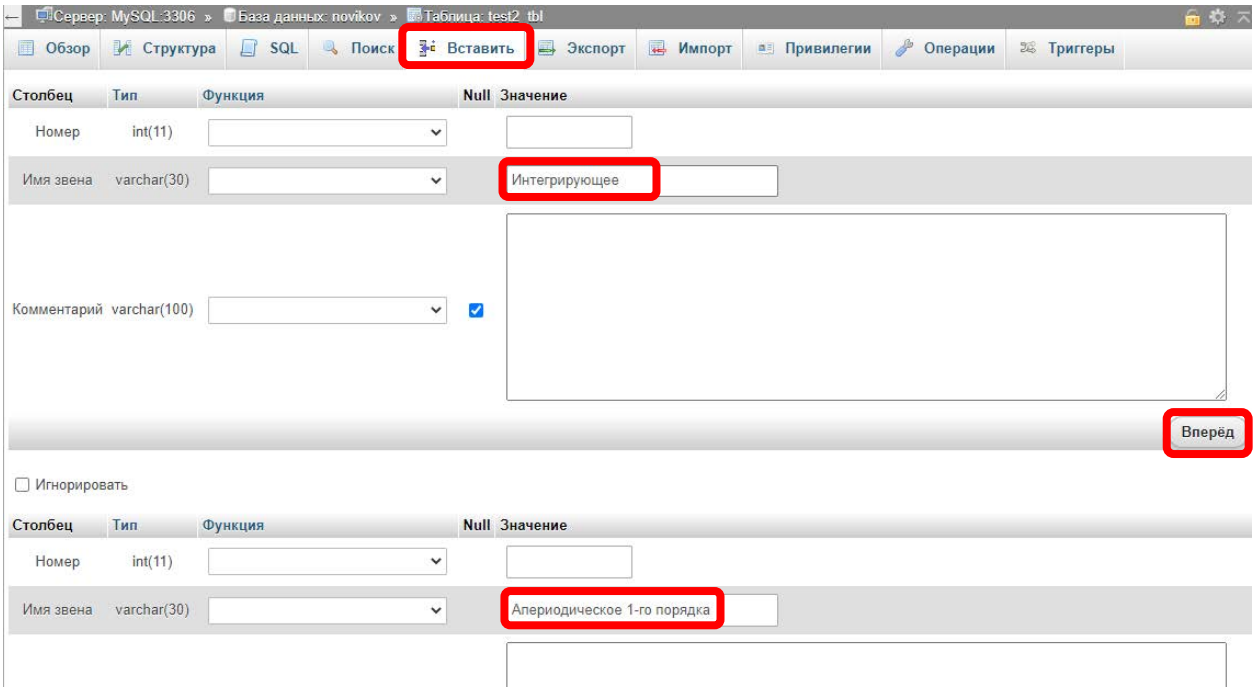


Рис. 61. Добавление строк в таблицу

3. Изменение, копирование и удаление строк таблицы

Для изменения, копирования или удаления строки таблицы необходимо перейти на вкладку «**Обзор**» и нажать соответствующую кнопку в выбранной строке (рис. 62).

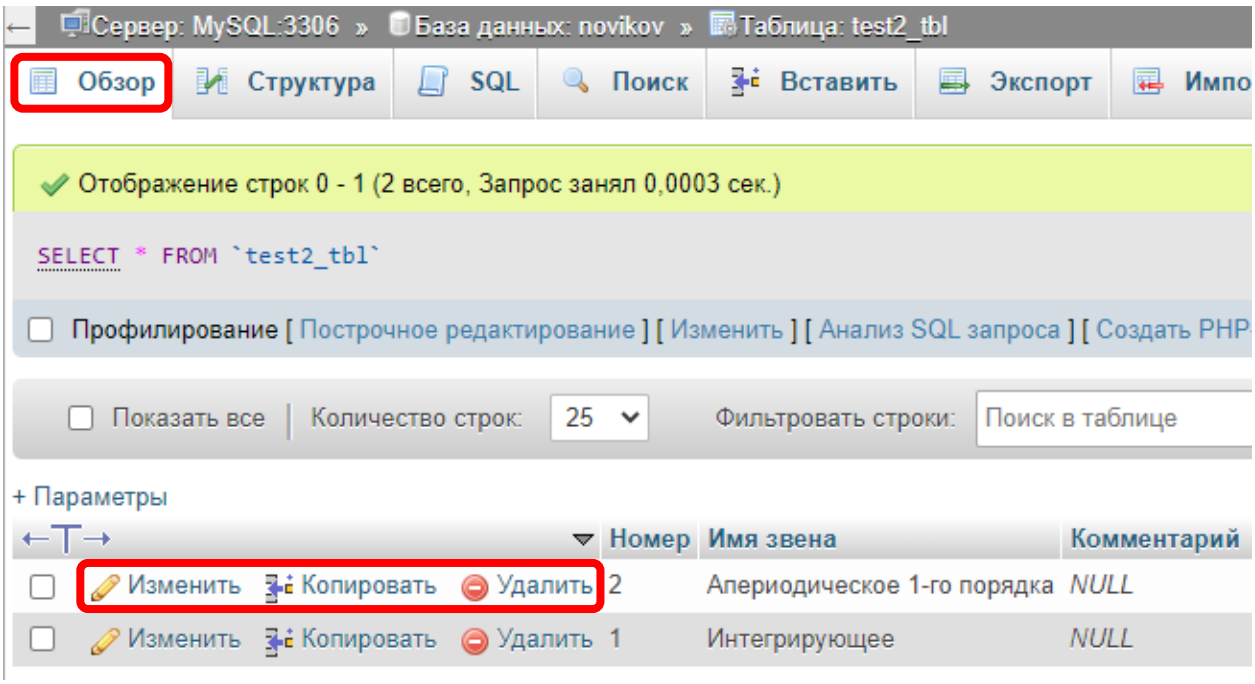


Рис. 62. Изменение, копирование или удаление строки таблицы

При выборе «**Изменить**» или «**Копировать**», будет открыто окно, аналогичное окну добавления строк, рассмотренному ранее (рис. 61).

Можно работать и с несколькими строками таблицы одновременно, для этого необходимо проставить галочки в требуемых строках, и нажать одну из кнопок под таблицей (рис. 63).

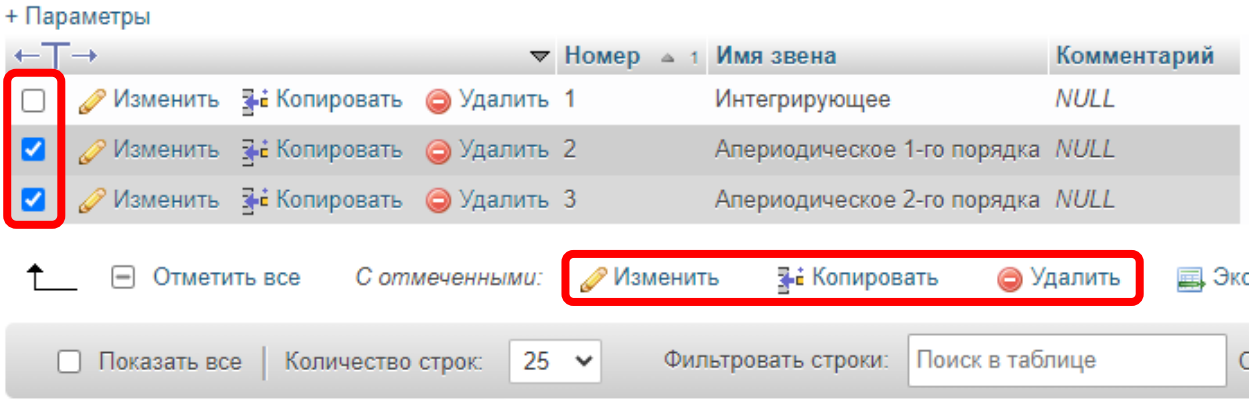


Рис. 63. Изменение, копирование или удаление нескольких строк таблицы

Кроме того, можно изменять значения отдельных полей прямо в режиме «**Обзор**». Для этого достаточно дважды кликнуть по соответствующему полю (рис. 64).

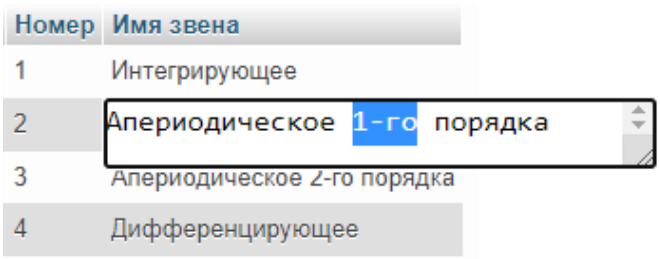


Рис. 64. Редактирование значения отдельного поля

4. Редактирование структуры таблицы

Помимо редактирования строк таблицы, возможно редактирование и столбцов для уже созданной таблицы. Для этого необходимо перейти на вкладку «**Структура**» (рис. 65). Здесь нам будут доступны такие функции, как добавление и удаление столбцов, изменение имени и типа столбца, изменение порядка (перемещение) столбцов.

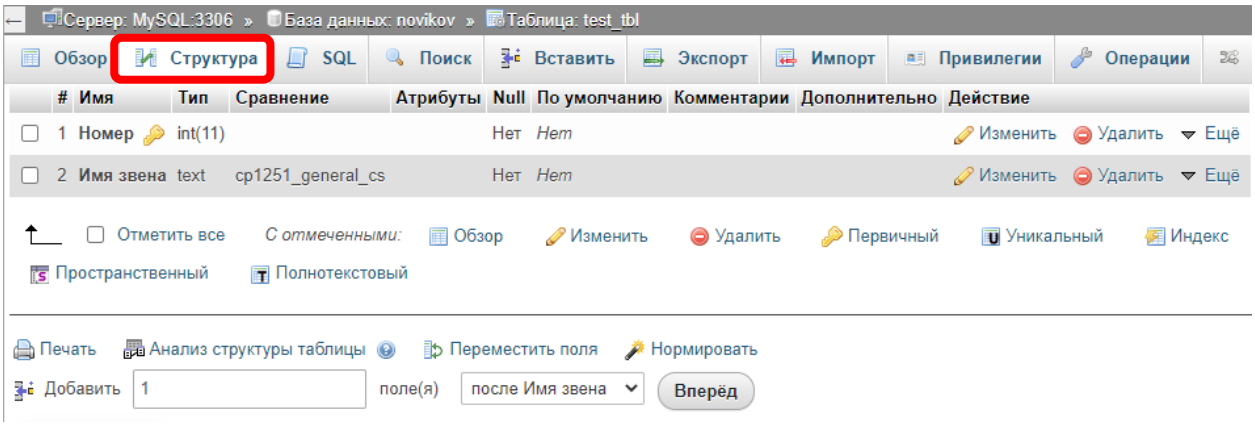


Рис. 65. Редактирование структуры таблицы

5. Копирование, перемещение и переименование таблицы

Копирование, перемещение и переименование таблиц осуществляется на вкладке «**Операции**». Переименование таблицы производится в разделе «**Move table**» (рис. 66), при этом нужно ввести новое имя таблицы, а имя базы данных оставить без изменения.

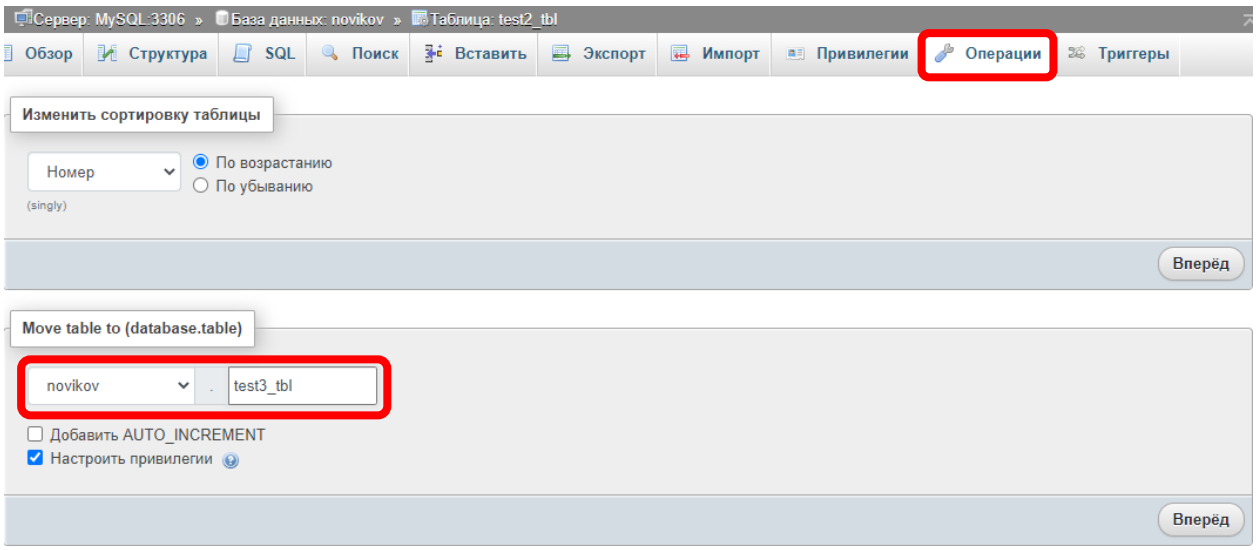


Рис. 66. Переименование таблицы

Перемещение в другую базу данных производится аналогично (в разделе «**Move table**»), только нужно выбрать из выпадающего списка имя базы данных, в которую производится перемещение. При это можно как задать новое имя для перемещаемой таблицы, так и оставить его прежним.

Копирование таблицы осуществляется из раздела «**Copy table**» (рис. 67). Копирование можно осуществлять как внутри этой же базы данных, так и выбрать из выпадающего списка другую базу данных. Также возможно копирование таблицы как вместе с данными (заполненными строками), так и только структуры таблицы («шапки», без заполненных строк).

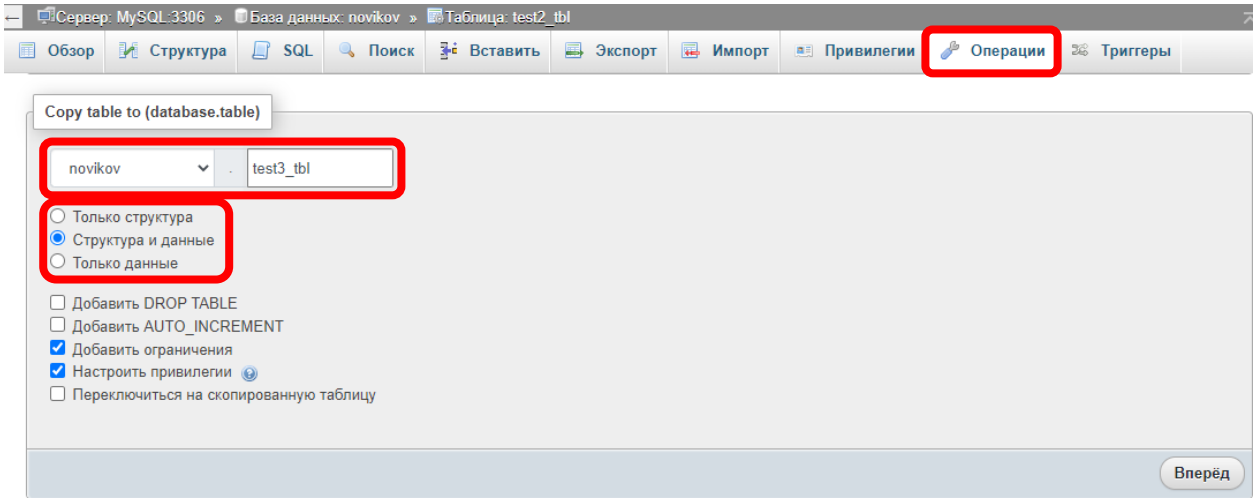


Рис. 67. Копирование таблицы

6. Очистка и удаление таблицы

Очистка и удаление таблицы, также производится на вкладке «**Операции**». Для этого необходимо перейти в раздел «**Удалить данные и таблицу**» (рис. 68),

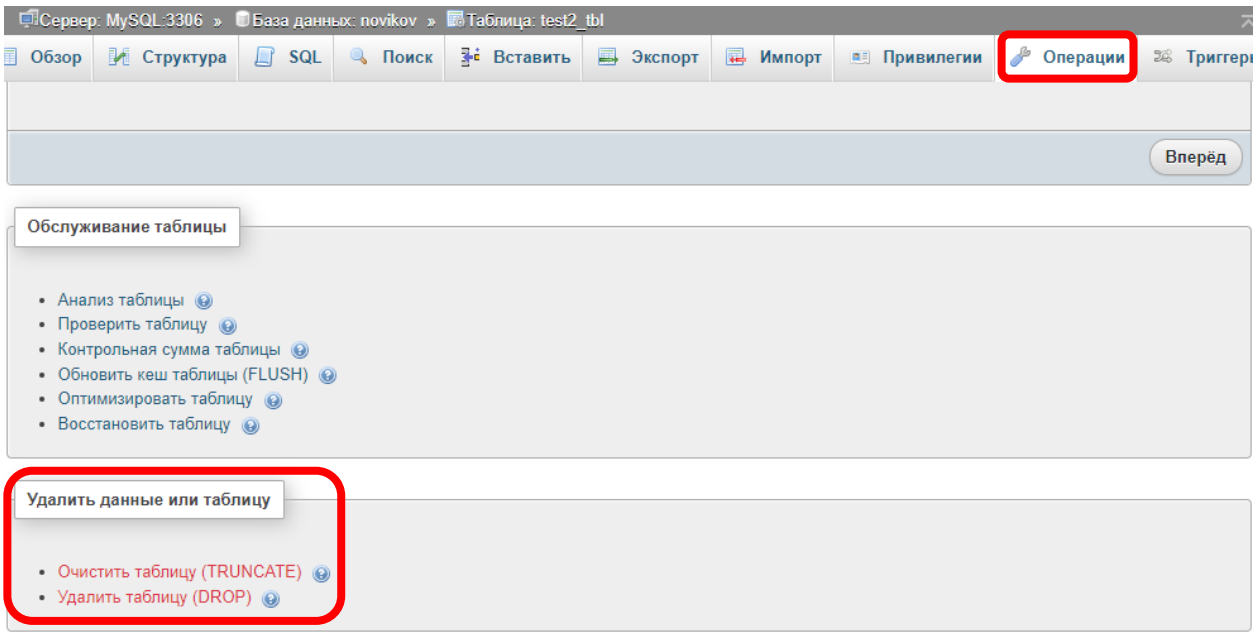


Рис. 68. Очистка и удаление таблицы

2.6. *Экспорт таблиц из PhpMyAdmin в Excel и «*.sql»

Бывает необходимо перенести созданные в **MySQL** базы данных на другой компьютер (сервер). Для этого необходимо экспортировать (рис. 69) базу данных в файл «***.sql**», а потом импортировать его на другом компьютере. Файлы ***.sql** – это текстовые файлы, содержащие **SQL**-запросы на создание и заполнение соответствующих таблиц. Их можно просмотреть через блокнот (рис. 70).

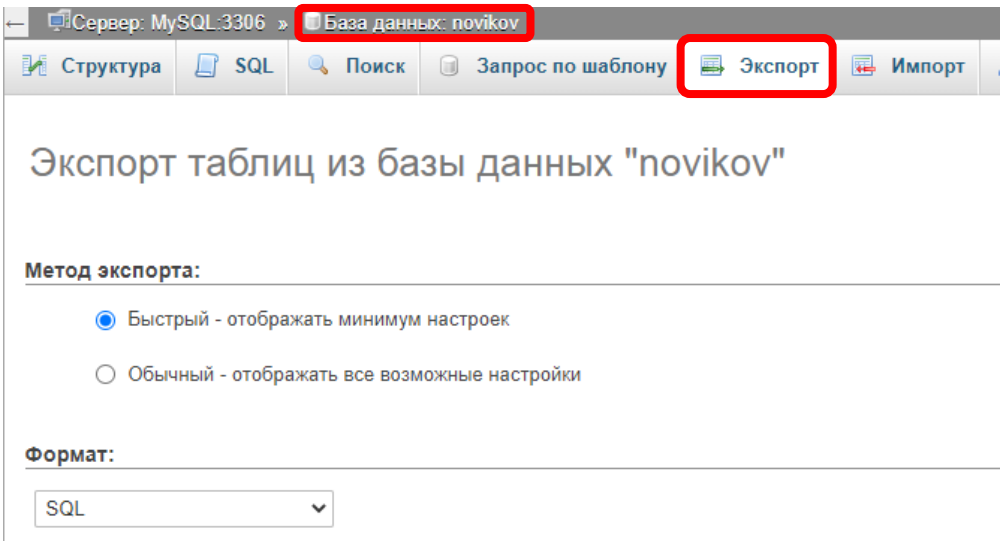


Рис. 69. Экспорт базы данных

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База данных: `novikov`
--

-- --------------------------------------------------------

--
-- Структура таблицы `lab2`
--

DROP TABLE IF EXISTS `lab2`;
CREATE TABLE IF NOT EXISTS `lab2` (
  `Номер` int(11) NOT NULL AUTO_INCREMENT,
  `Наименование` varchar(100) COLLATE cp1251_general_cs NOT NULL,
  `Тип` varchar(100) COLLATE cp1251_general_cs NOT NULL,
  `Изображение` varchar(200) COLLATE cp1251 general cs NOT NULL,
```

Рис. 70. Пример файла *.sql

Для переноса содержимого **SQL**-таблицы в **Excel** требуется:

1. Через вкладку «**Обзор**» поместить содержимое выбранной таблицы в буфер обмена (рис. 71);

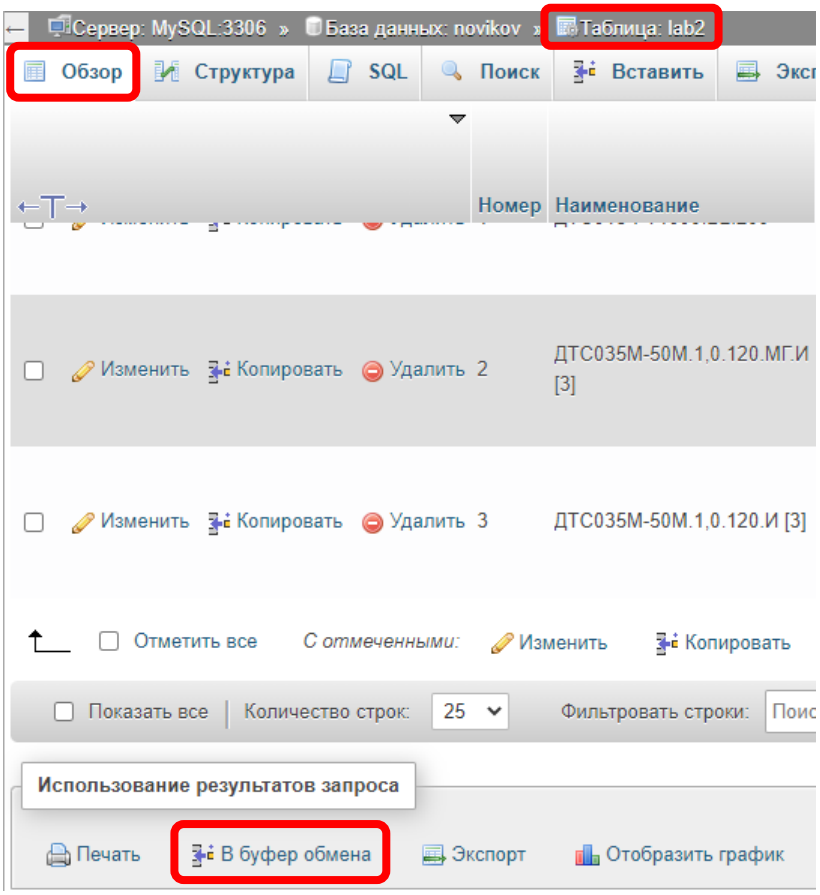


Рис. 71. Копирование таблицы в буфер обмена

2. На новом **Excel**-листе выделить все ячейки (Ctrl+A) и изменить их формат на «Текстовый» (рис. 72);

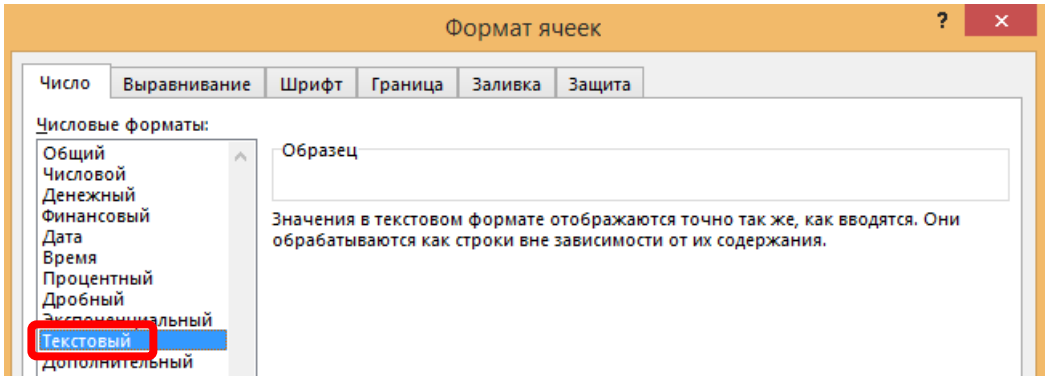


Рис. 72. Формат ячеек в Excel

3. Вставить на лист содержимое буфера обмена (Ctrl+V);
4. *Изменение формата ячеек требовалось для правильной вставки числовых значений, например, значений «рейтинга», которые записаны как числа через точку, но для* ***Excel*** *требуется запятая, поэтому необходимо* выделить ячейки с числовыми значениями и

нажать Ctrl+F, где перейдя на вкладку «**Заменить**» (рис. 73), произвести замену точек на запятые;

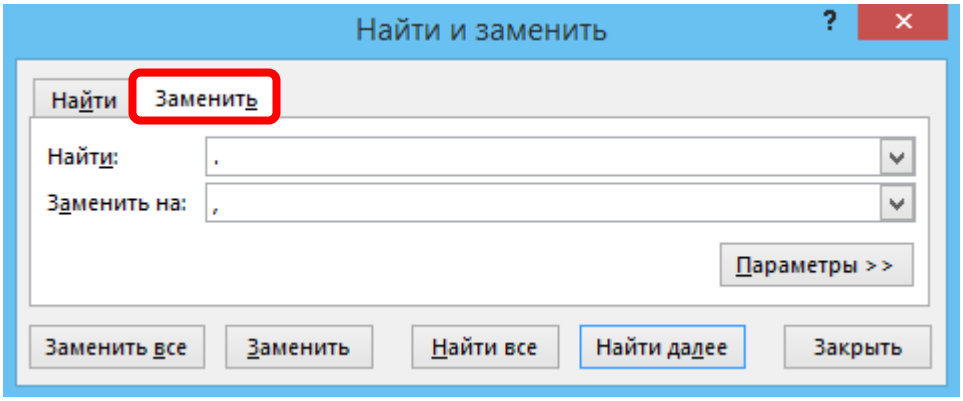


Рис. 73. Замена точек на запятые

5. Далее вновь выделить все ячейки (Ctrl+A) и изменить их формат обратно на «Общий».

Пример итоговой таблицы **Excel** представлен на рис. 74.

	A	B	C	D	E	F	G	H	I
1	MySQL:3306/novikov		http://localhost/phpmyadmin/index.php?route=/sql&db=novikov&tab						
2									
3	Отображение строк 0 - 2 (3 всего, Запрос занял 0,0004 сек.)								
4									
5									
6	SELECT * FROM `lab2`								
7									
8									
9	Номер	Наименов	Тип	Изображе	Рейтинг	Производ	Страна	Гарантия	Цена
10	1	ДТС015-Р	Термоме	dts015.pn	4,7	Овен	Россия	24	2490
11	2	ДТС035М-	Термоме	dts035m.p	4,9	Овен	Россия	24	7998
12	3	ДТС035М-	Термоме	dts035m.p	4,9	Овен	Россия	24	7398

Рис. 74. Итоговая таблица в Excel

Таким способом можно перенести в **Excel** не только таблицу целиком, но и результаты выполнения отдельных запросов.

3. ОСНОВЫ ЯЗЫКА HTML

HTML (читается как «Эйчь-Ти-Эм-Эл» или «Эйчь-Ти-Эм-Эль») – язык разметки интернет-страниц (они-же «Web-страницы», или «HTML-страницы»). Базовый пример кода на языке HTML выглядит следующим образом:

```
<HTML>
  <HEAD> <TITLE>СУБД - Новиков</TITLE> </HEAD>
  <BODY>
     <B>Жирный текст</B><BR>
     <I>Курсив</I>
  </BODY>
</HTML>
```

Язык состоит из «тегов», которые могут быть парными, т. е. иметь закрывающийся тег (как например <B> и </B>), и непарными (как например
).

!!! В студенческих работах написание имен тегов большими буквами обязательно!

Заголовок страницы, отображаемый как название соответствующей закладки в браузере, указывается между тегами **<TITLE>** и **</TITLE>**.

!!! В студенческих работах, необходимо указывать в качестве заголовка название предмета, фамилию студента и текущий год!

Целью данного курса не является изучение всей структуры языка HTML, поэтому далее мы будем работать только с телом интернет-страницы, расположенным между тегами **<BODY>** и **</BODY>**.

Примеры основных тегов языка HTML:

- выделение текста **жирным**, *курсивом* или подчеркиванием:
  ```
  <B>Жирный текст</B>, <I>Курсив </I>,
  <B><U>Жирный подчеркнутый</U></B>
  ```
- переход на новую строку:
  ```
  <BR>
  ```
- гиперссылка:
  ```
  <A href="http://gturp.spb.ru">Ссылка на сайт</A>
  ```
- вставка изображения:
  ```
  <IMG src="Img/logo_green.png" align=right>
  ```
- размер и цвет шрифта:
  ```
  <FONT size = "7" color = "red">Большой красный текст</FONT>
  <FONT color = "#FF00FF">Розовый текст</FONT>
  ```
  Размер шрифта задается цифрой от 1 до 7. Цвет можно задать по имени (на английском) или по цветовым координатам в формате

RRGGBB, где RR – красный (red), GG – зеленый (green), BB – синий (blue). Каждый цвет задается в диапазоне от 00 до FF (в 16-ричной системе).

- вставка кнопки:
  <BUTTON>Нажми меня!</BUTTON>

3.1. Создание Web-страницы

Для создания Web-страницы необходимо:

1. Создать (рис. 75) на рабочем столе текстовый файл («**Текстовый документ**»).

!!! После окончания работы необходимо удалять за собой файлы с рабочего стола!

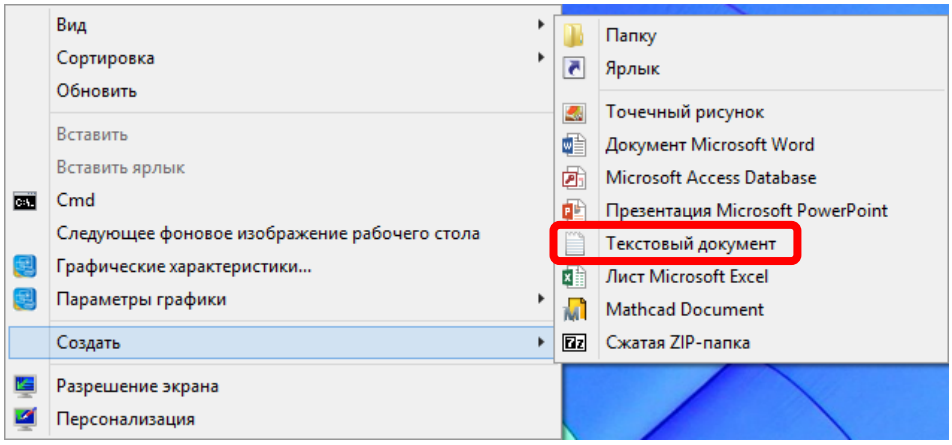


Рис. 75. Создание текстового документа

2. Открыть созданный документ в блокноте и напечатать в нем код HTML-страницы (базовый пример кода, рассмотренный ранее).
3. Сохранить напечатанный код и закрыть блокнот.
4. Переименовать данный файл в «**Demo.html**».
5. Открыть файл с помощью браузера (рис. 76).

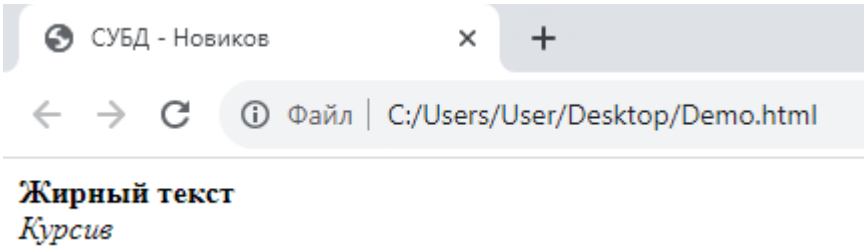


Рис. 76. Пример Demo.html, открытый в браузере

Операционная система Windows позволяет открывать файл двойным кликом мышкой. При этом файл открывается только в какой-либо одной программе. В зависимости от настроек системы, это могут быть разные программы. Если требуется открывать один файл из разных программ (как в нашем случае из блокнота, и из браузера), то необходимо использовать меню «Открыть с помощью» (рис. 77).

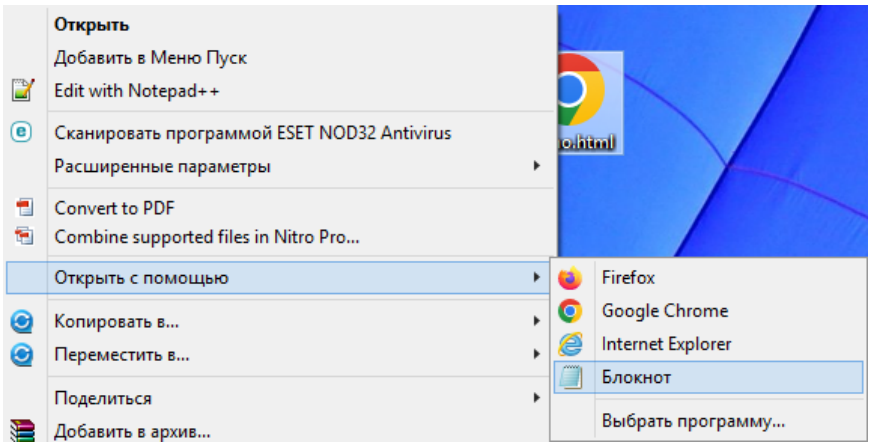


Рис. 77. Меню «Открыть с помощью»

В Windows может быть отключено отображение расширений файлов (рис. 78). В этом случае будет невозможно правильно переименовать файл, сменив расширение файла (с *.txt на *.html).

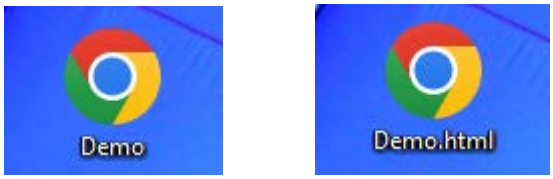


Рис. 78. Отображение файла без расширения и с расширением

Для решения этой проблемы нужно воспользоваться одним из двух способов:

1. Перенастроить Windows на отображение расширений файлов через «Параметры папок». Для этого в любой папке необходимо открыть меню «**Вид**» (рис. 79) и нажать кнопку «**Параметры**». Далее в появившемся окне «**Параметры папок**» (рис. 80), необходимо снять галочку «**Скрывать расширения для зарегистрированных типов файлов**».
2. Либо открыть требуемый файл в блокноте и, воспользовавшись функцией «Сохранить как», задать имя файла вместе с правильным расширением.

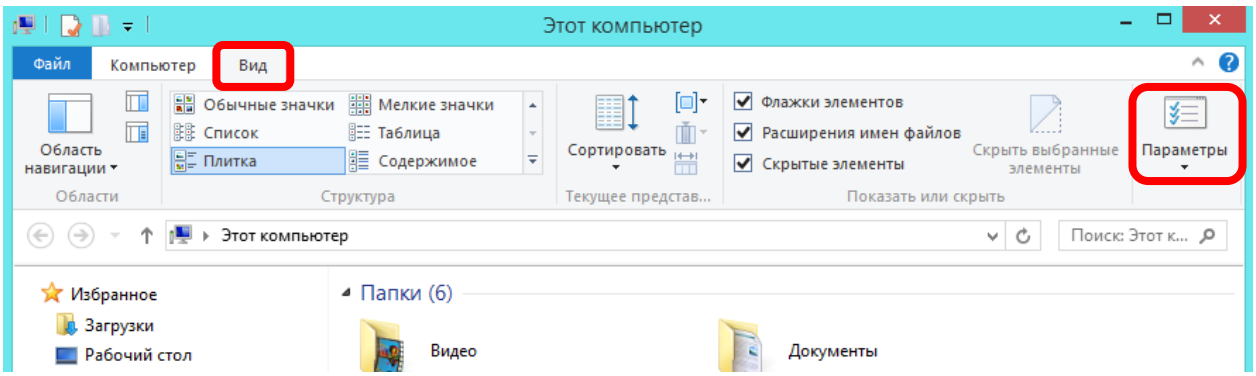


Рис. 79. Вызов «Параметры папок»

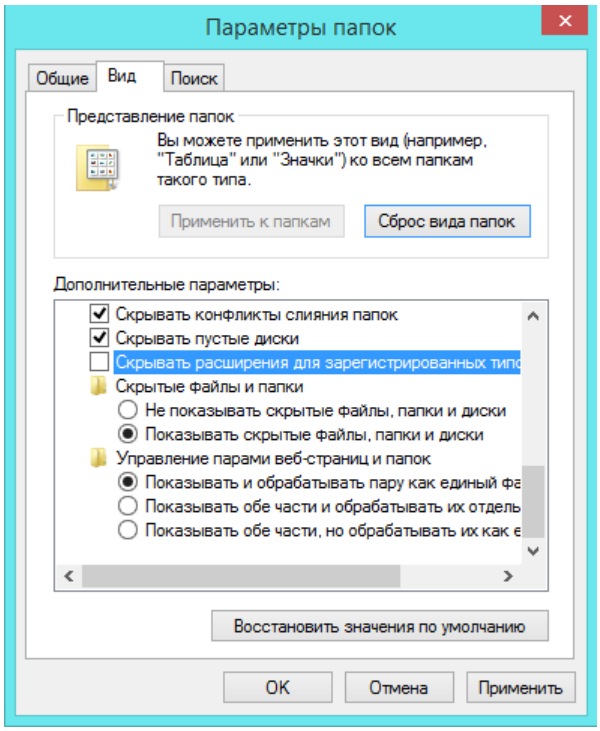


Рис. 80. Настройка отображения расширения файла

3.2. ЛАБОРАТОРНАЯ РАБОТА № 3 (часть 1)

Необходимо разработать **HTML**-страницу для одного из датчиков (устройств, приборов), который использовался в предыдущей Лабораторной работе. Страница должна содержать все параметры, использованные ранее в таблицах, в том числе: наименование датчика, его описание и список характеристик, рейтинг товара, информацию о цене, гарантии и наличие устройства, количество магазинов (складов), где это устройство имеется, список этих магазинов (складов) и их адреса. Кроме того, **HTML**-страница должна содержать изображение описываемого устройства и кнопку «Купить» (или «Заказать»), или несколько кнопок «Купить», по одной для каждого из магазинов.

Для достижения поставленной цели необходимо:

1. Переименовать ранее созданный файл «Demo.html» (или его копию) в «**Lab3.html**».
2. Создать на рабочем столе папку «**Img**» для размещения в ней изображений.
3. Найти в Интернете изображение требуемого датчика и поместить его в папку «**Img**».
4. Задать для этого изображения понятное имя, например, «**dts015.png**» (т. к. в моем случае используется датчик «Овен ДТС-015»). *В дальнейшем в теге <IMG>, адрес этого изображения будет «**src="Img/dts015.png"**»*.
5. Открыть в блокноте файл «**Lab3.html**» и, редактируя его между тегами **<BODY>** и **</BODY>**, используя при этом текст и теги <B>, <I>, <U>,
, <A>, <IMG>, <FONT> и <BUTTON>, добиться требуемого результата.

!!! Можно использовать и другие теги, если автор работы в состоянии объяснить их назначение и принцип работы.

Для более удобного редактирования **HTML**-документов, необходимо использовать блокнот с подсветкой синтаксиса, например, **Notepad++**.

Пример итоговой **HTML**-страницы представлен на рис. 81.

Файл | C:/Users/Alex/Desktop/Lab3.html

Термометр сопротивления ДТС015-PT1000.B2.200

Рейтинг: **4,7**
Производитель: Овен
Страна: Россия
Гарантия: **24 месяца**
Цена: **2 490,00 руб.**
Купить

Наличие: **523 шт.** (много)

- (•) г. Санкт-Петербург, ул. Александра Матросова, д. 4, корп. 2, литер Д - **400 шт.**
- ( ) г. Екатеринбург, ул. Малышева, 164 - **123 шт.**

Описание

ДТСxx5 с коммутационной головкой позволяют измерять температуру до 500 °С (ДТС с платиновым ЧЭ) и до 180 °С (ДТС с медным ЧЭ). Подключение к измерительной линии производится медным кабелем (кабель в комплекте не идет, заказывается отдельно).

Отличительные особенности:
- Бюджетная цена датчиков.
- Имеют сертификат средств измерений и проходят первичную поверку на заводе-изготовителе.

Номинальные статические характеристики (НСХ) по ГОСТ 6651-2009:
- 50М и 100М (W100 = 1,428, α = 0,00428 °С-1)

Основные характеристики

Диапазон измеряемых температур: –60…+500 °С
Погрешность: 0,5%
Выходной сигнал: Сопротивление (Pt1000)
Схема подключения: Двухпроводная
Напряжение питания (номинальное): Не требуется
Диапазон допустимых напряжений питания: Не требуется
Ссылка на документацию: https://owen.ru/uploads/292/kratkoe_rukovodstvo_dtshh5.pdf

Прочие характеристики

Температура окружающей среды: –60…+85 °С
Количество чувствительных элементов: 1
Длина погружаемой части L, мм: 200
Материал коммутационной головки: Пластмасса
Класс защиты: IP54
Среда измерения: Твердые, жидкие и газообразные среды (неагрессивные)
Сопротивление изоляции: не менее 100 МОм
Средний срок службы: не менее 10 лет

Рис. 81. Пример итоговой HTML-страницы

3.3. *Блочная структура HTML-документа

В приведенном выше примере (рис. 81) была рассмотрена **HTML**-страница, в которой рисунок и текст разбиты на две колонки. Добиться такого результата можно окружив соответствующие часть **HTML**-документа тегами **<DIV>** и **</DIV>**. Хотя рамки для этих элементов и не отображаются, продемонстрируем их пунктирными линиями на рис. 82.

Рис. 82. Блочная структура документа

Положение блоков задается через стиль элемента, также в нем можно указать и ширину блока. Например, можно написать следующий **HTML**-код:

```
<div style="display: inline-block; width: 350;">
  <IMG width="300" src="Img/dts015.png">
</div>
<div style="display: inline-block;">
  Рейтинг: <b>4,7</b>
  <BR>Производитель: Овен
  <BR>Страна: Россия
  <BR>...
</div>
```

4. APACHE-СЕРВЕР

Apache HTTP-сервер (произносится «**апа́ч**») – программное обеспечение, позволяющее создать HTTP-сервер (он же Web-сервер) для запуска страниц на языке HTML. Клиентом для подключения к такому серверу является Web-браузер.

Сервер Apache и язык HTML позволяют запускать статические страницы. Для запуска динамических страниц[7] совместно с ними используется язык **PHP**[8] (читается как «Пи-Аш-Пи» или «Пи-Эйчь-Пи»). Фактически **PHP** является «препроцессором» для **HTML**, т. е. осуществляет его обработку, прежде чем отправить его клиенту. Динамизация страниц бывает двух видов – на стороне сервера и на стороне клиента. **PHP** отвечает за динамизацию на сервере. Для динамизации на стороне клиента используются **JavaScript** и **JSON** *(о которых здесь рассказываться не будет)*. Возможен и смешанный вариант создания динамической страницы, которая вначале обрабатывается на стороне сервера, а потом на стороне клиента.

WampServer (**WAMP**-сервер) – сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, phpMyAdmin и другие дополнения, предназначенная для web-разработки под Windows[9]. Имеет автоматический инсталлятор. Для управления сервером и его настройками WampServer создает иконку в трее (у часов). Позволяет выбирать различные версии Apache, MySQL, MariaDB и PHP. Название расшифровывается как:

WAMP = Windows + Apache + MySQL + PHP.

4.1. Установка WAMP-сервера

Для установки **WAMP**-сервера необходимо вначале скачать дистрибутив программы с официального сайта[10]:

https://www.**wampserver**.com/

При написании данного пособия использовалась версия **Wampserver 3.2.6 x64**, включающая следующие версии программ:

- Apache 2.4.51;
- PHP 5.6.40/7.4.26/8.0.13/8.1.0;
- MySQL 5.7.36|8.0.27;
- MariaDB 10.5.13|10.6.5;
- PhpMyAdmin 4.9.7 & 5.1.1.

Также рекомендуется установить блокнот с подсветкой синтаксиса языков программирования, например **Notepad++**[11], который можно скачать по адресу:

https://**notepad-plus-plus**.org/downloads/

Установка **WAMP**-сервера, в основном, производится с настройками по умолчанию. При этом стоит обратить внимание, что установка должна производиться в папку «**C:\wamp64**» (рис. 83), т. е. только в корень диска (нельзя устанавливать данную программу в «C:\Program Files …»!). Путь не должен содержать пробелов и других специальных знаков, а также русских букв. Установка возможна только от имени Администратора.

!!! Нельзя устанавливать **WAMP**-сервер поверх имеющейся версии! Если ранее в данную папку уже был установлен **WAMP**-сервер, то необходимо вначале полностью удалить предыдущую версию при помощи стандартной функции Windows «Удалить или изменить программу», а при необходимости (если папка осталась) также удалить папку «C:\wamp64» вручную.

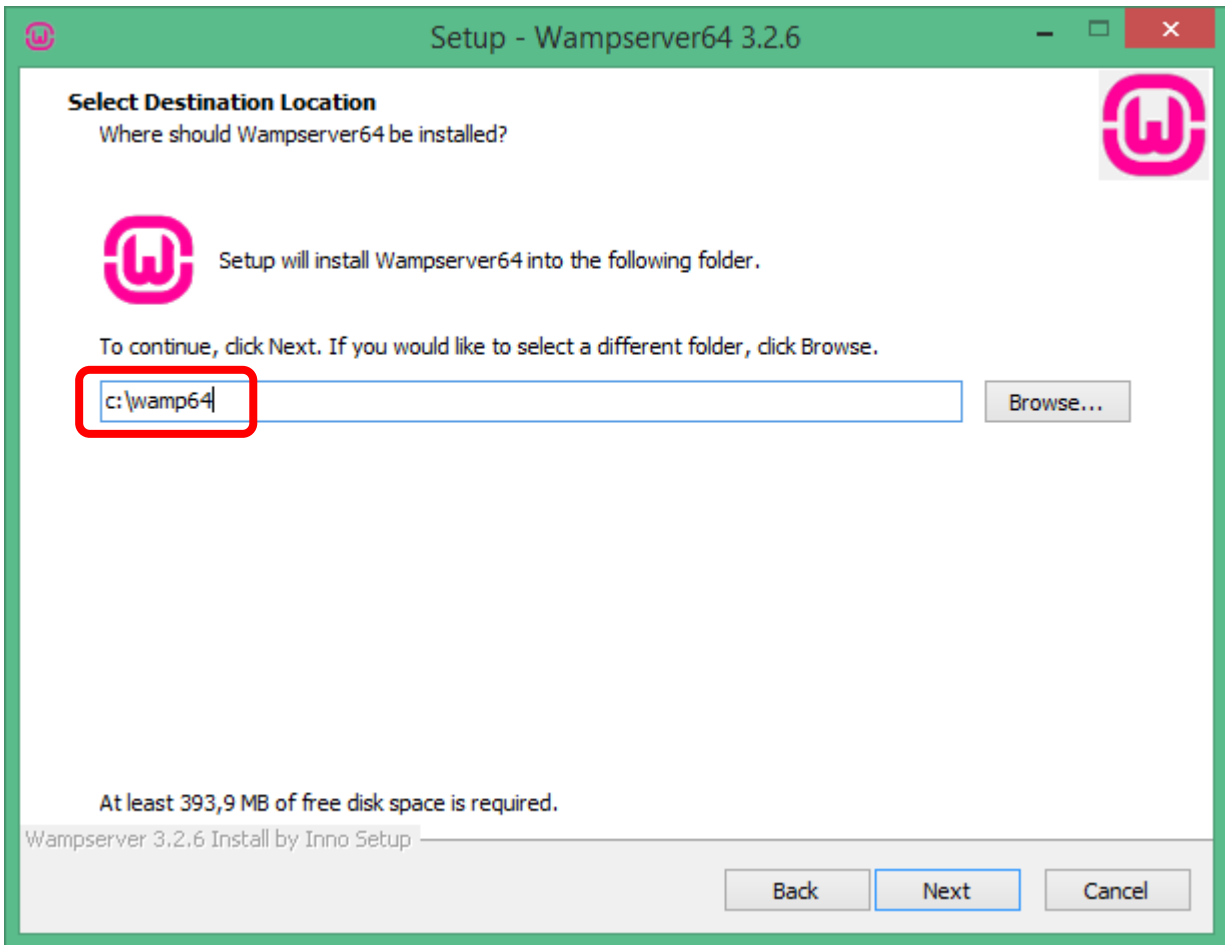


Рис. 83. Выбор папки для установки

При установке на **Windows 8** (и более ранние версии) необходимо выбрать для установки MySQL версии 5. При установке на **Windows 10** и более новые версии необходимо выбрать для установки MySQL версии 8 (рис. 84).

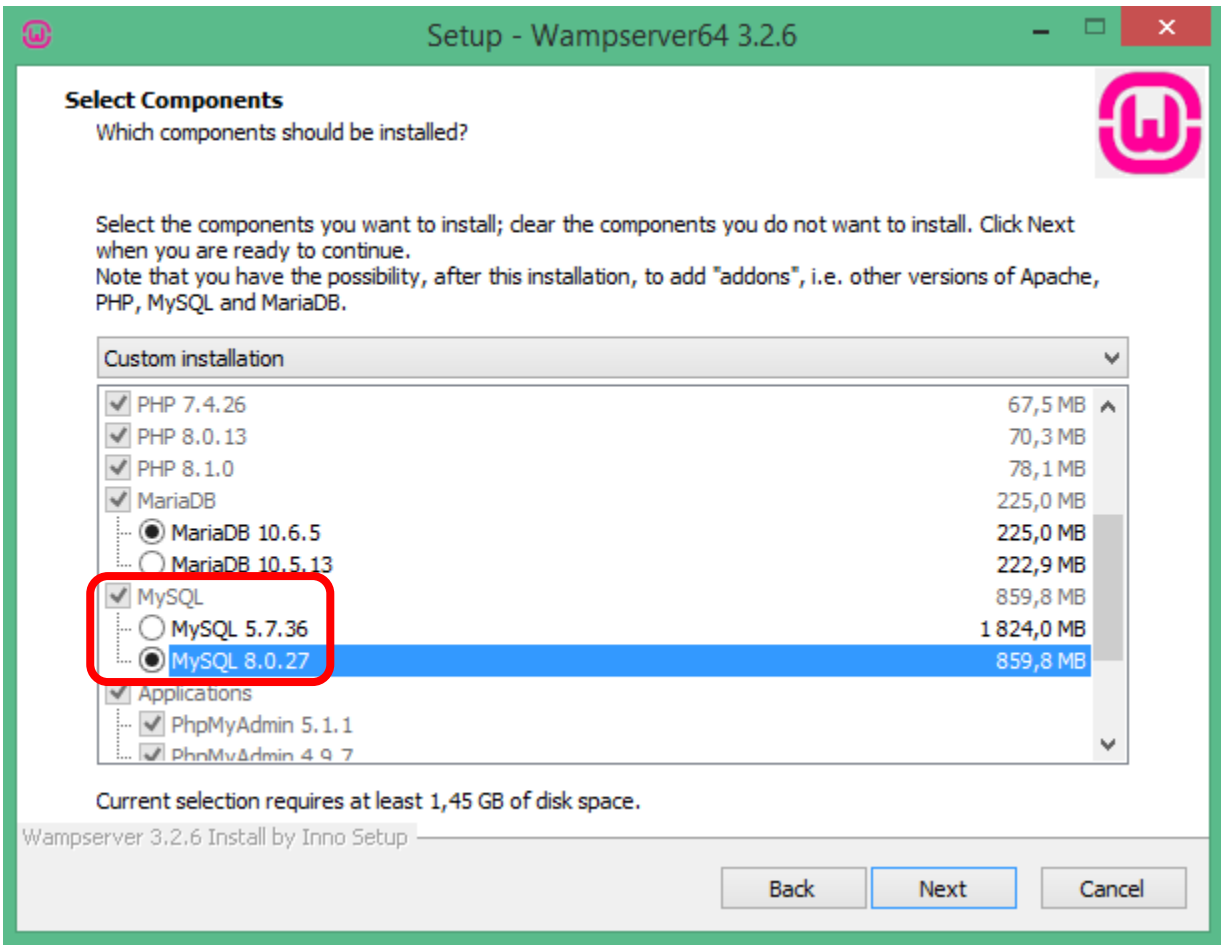


Рис. 84. Выбор версии MySQL для установки

После установки необходимо:

1. Запустить появившийся на рабочем столе ярлык «**Wampserver64**» (рис. 85).

Рис. 85. Ярлык WAMP-сервера на Рабочем столе

2. Щелкнуть правой кнопкой мыши по соответствующему значку у часов и переключить «Язык» («**Language**») на «**russian**» (рис. 86). *Сервер стартует не мгновенно, может потребоваться подождать загрузки сервера до 1 минуты. После запуска значок у часов должен стать зеленым*.

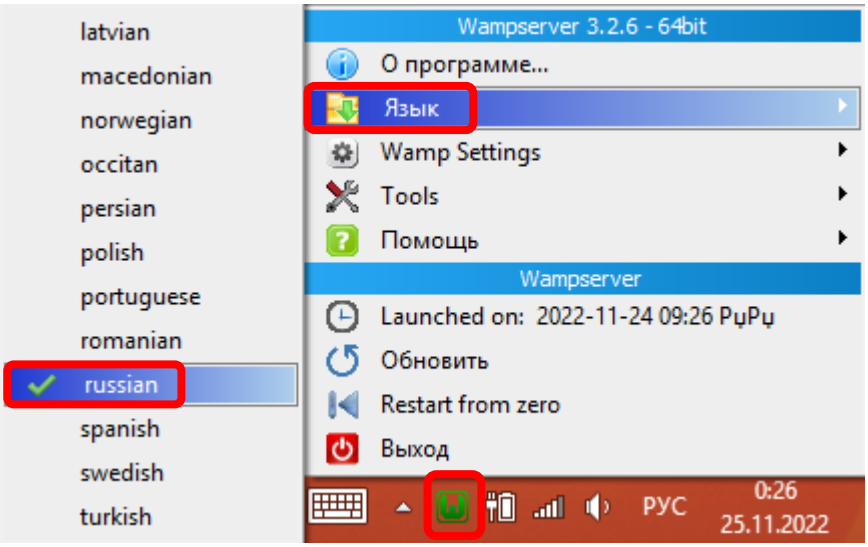


Рис. 86. Выбор языка

3. Проверить работу сервера, запустив браузер и введя адрес **localhost** (или, что равносильно, адрес **127.0.0.1**). Далее выбрать «**PhpMyAdmin**» (рис. 87).

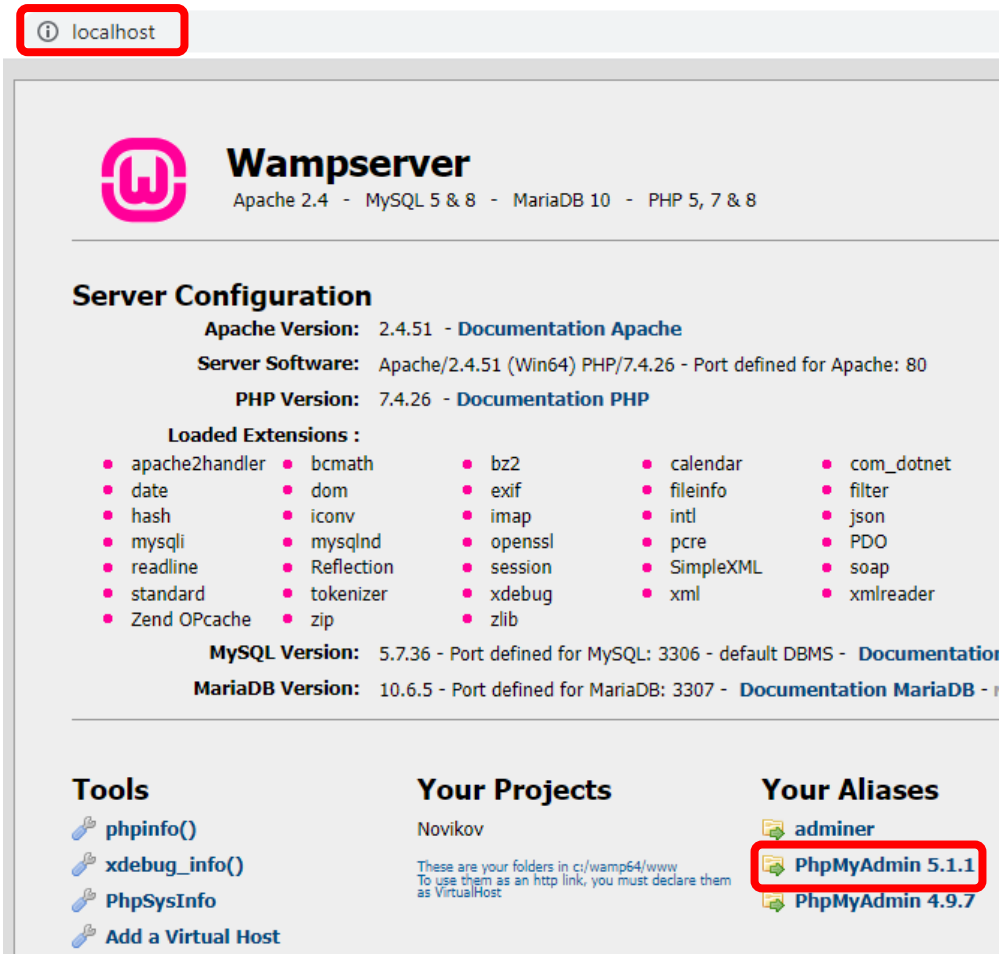


Рис. 87. Подключение к WAMP-серверу через браузер

4. Ввести имя пользователя **root** (без пароля) и подключиться к серверу **MySQL** (рис. 88).

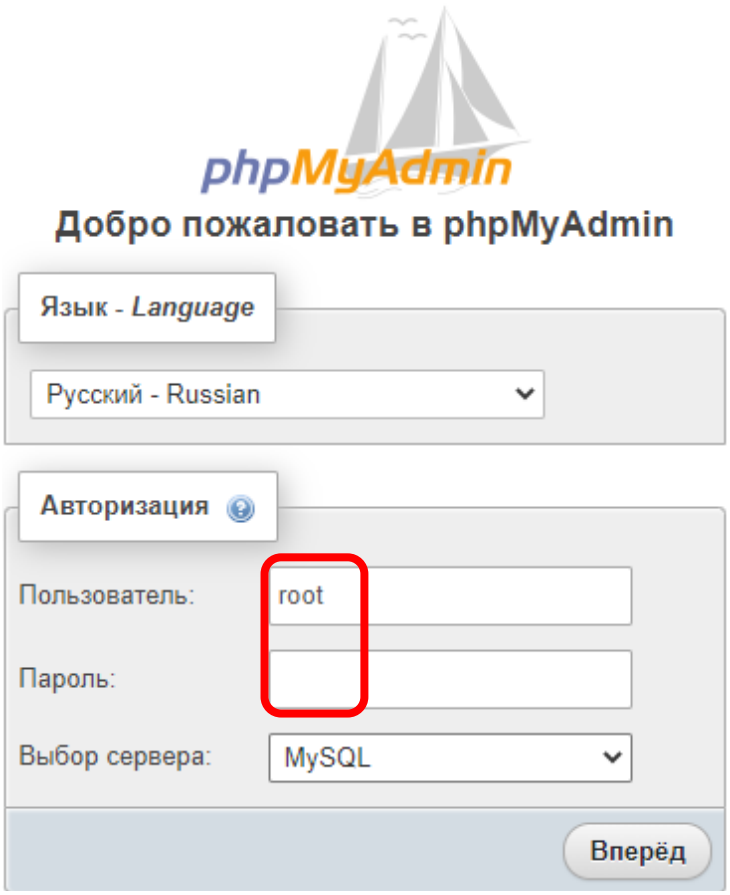


Рис. 88. Подключение к серверу MySQL

Возможные проблемы:

По умолчанию в Windows может быть включен «**IIS Windows**» (Internet Information Services), тогда он будет перехватывать вход по адресу **localhost** (он же **127.0.0.1**). Тогда в браузере вместо страницы **WAMP**-сервера отобразится страница **IIS Windows** (рис. 89).

Для отключения **IIS Windows** необходимо:

Зайти в «Установку и удаление программ» → «**Включение или отключение компонентов Windows**»

1. рис. 90).
2. Отключить «**Службы IIS**» (рис. 91).
3. Если в браузере все еще видна страница **IIS Windows**, нажать в нем **Ctrl+F5** (обновить).

Кроме **IIC Windows**, перехватывать вход по адресу **localhost** могут и другие программы (например, **Skype**). Команда **netstat** позволяет проверить отсутствие других программ на порту **80**. Для ее запуска необходимо нажать комбинацию клавиш **Win+R** (рис. 92) и ввести «**cmd**». В появившемся окне ввести (рис. 93):

netstat -a

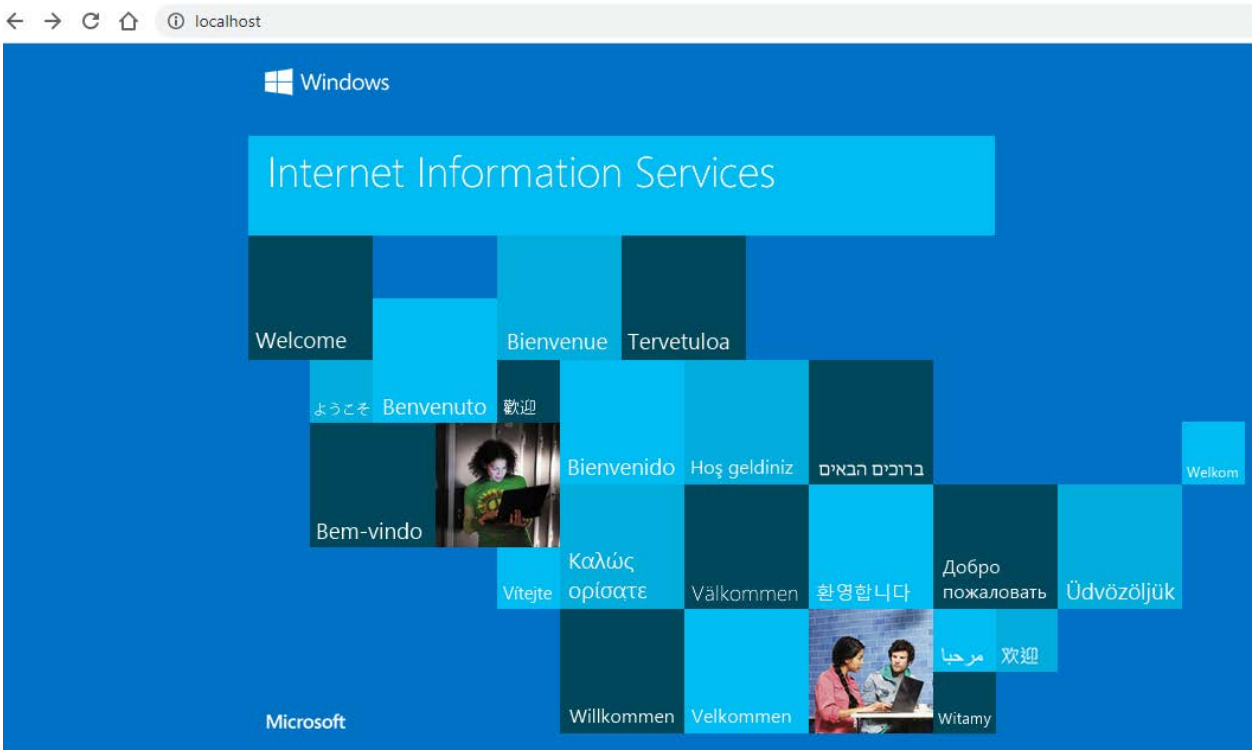


Рис. 89. Главная страница Windows IIS

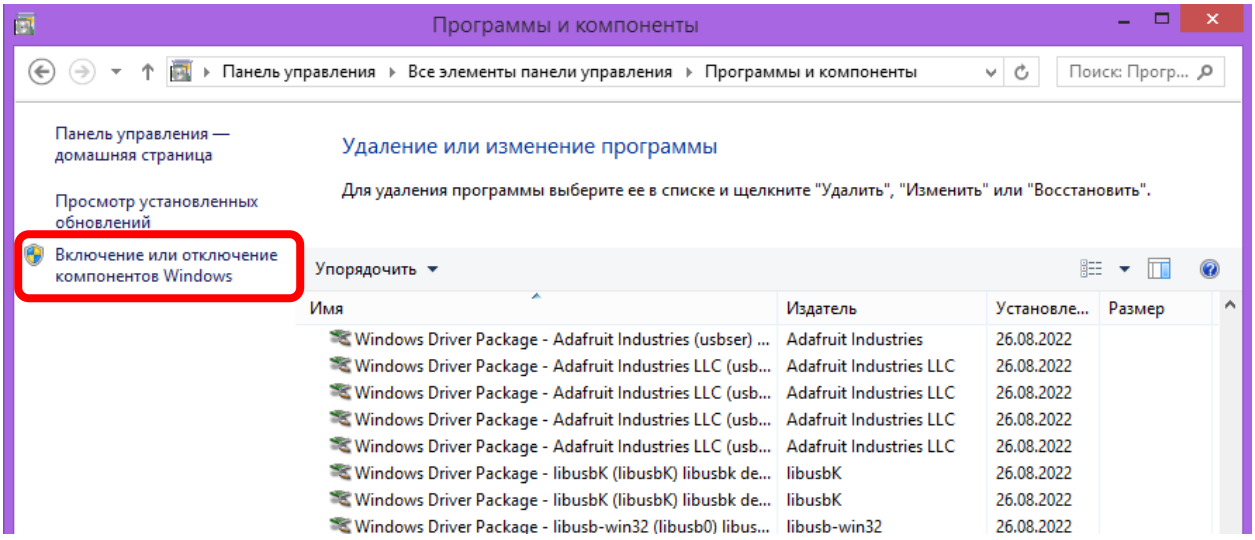


Рис. 90. Включение и отключение компонентов Windows

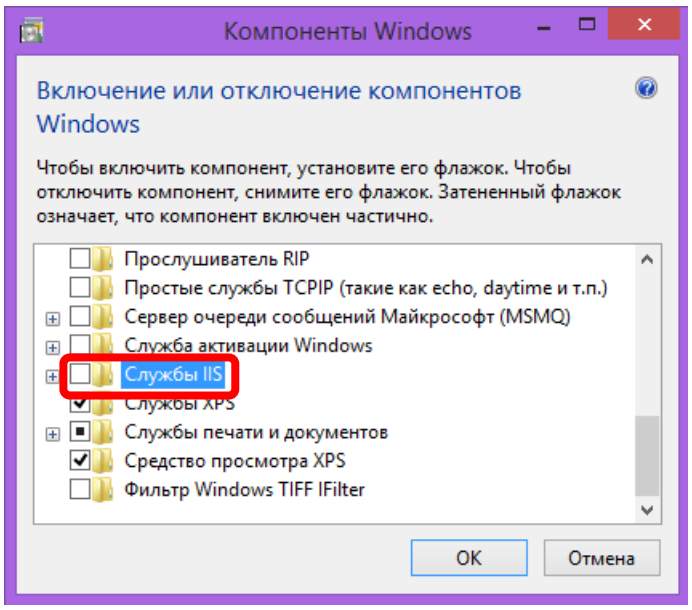


Рис. 91. Отключение служб IIC

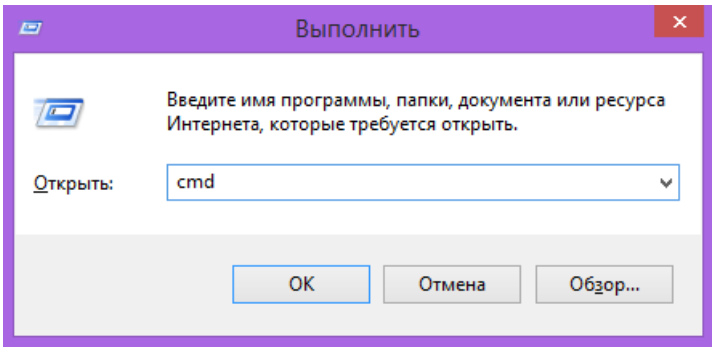


Рис. 92. Вызов «cmd»

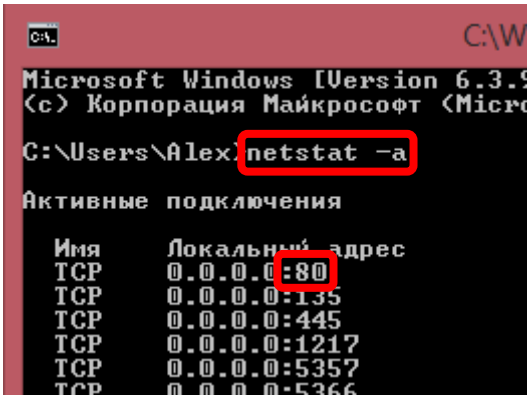


Рис. 93. Результат работы команды netstat

4.2. Подготовка к работе

Перед началом работы необходимо проделать следующее:

1. Зайти в папку «**C:\wamp64\www**» (это папка Web-сервера, отображаемая при вводе в браузере адреса «**localhost**»).
2. В папке «**www**» создать папку со своей фамилией и текущим годом (необходимо использовать только латинские буквы, а вместо пробела использовать знак подчеркивания «_»).
   *В моих примерах это будет только фамилия «**Novikov**». Адрес для работы с этой папкой в браузере будет, соответственно, «**localhost/Novikov**».*

!!! Допускается изменять и удалять файлы только внутри этой своей папки!

3. Внутри своей папки создать еще две папки «**Admin**» и «**Img**»;
4. Переместить созданные ранее (в **Главе 3**) файлы «**Demo.html**» и/или «**Lab3.html**», а также папку «**Img**» с изображениями (если имеется) в папку со своей фамилией;
5. Открыть скопированные **HTML**-файлы из браузера. Их адреса будут, соответственно, «**localhost/Novikov/Demo.html**» и «**localhost/Novikov/Lab3.html**».

!!! Если после открытия страницы в браузере вместо русских символов отображаются «крякозябры», то необходимо сменить кодировку соответствующего HTML-документа. Для этого необходимо:

1. Открыть требуемый файл в блокноте **Notepad++**.
2. Выбрать меню «Кодировки» → «**Преобразовать в UTF-8 с BOM**» (рис. 94).
3. Сохранить файл.
4. Обновить страницу в браузере (**F5**).

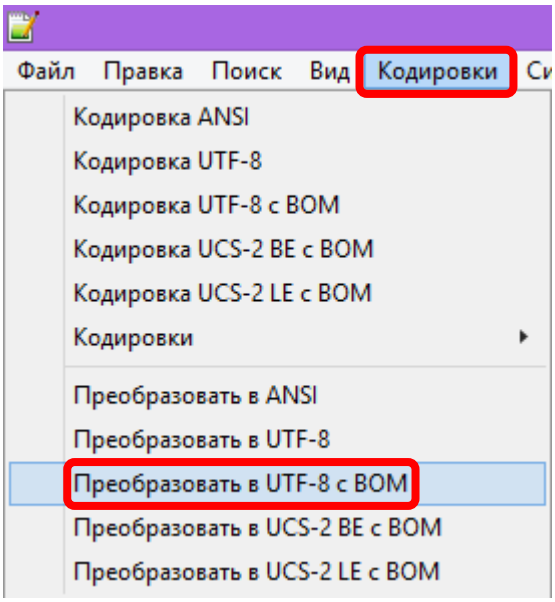


Рис. 94. Смена кодировки

4.3. Основы языка PHP

Основной задачей языка **PHP** является «подстановка значений» в код **HTML**, т. е. процесс частичной автоматической генерации кода (хотя возможна и полная генерация).

Рассмотрим простейший пример **HTML**-документ:

```
<HTML>
  <HEAD> <TITLE>СУБД - Новиков</TITLE> </HEAD>
  <BODY>
      Дата обращения: 17.10.2022<BR>
      Время обращения: 13:50:43
  </BODY>
</HTML>
```

В данном случае страница просто отображает (рис. 95) две заранее прописанные строчки текста. Но невозможно постоянно вручную менять содержимое страницы, поэтому необходимо, чтобы сервер сам подставлял текущие дату и время. Для этого и существует **PHP**.

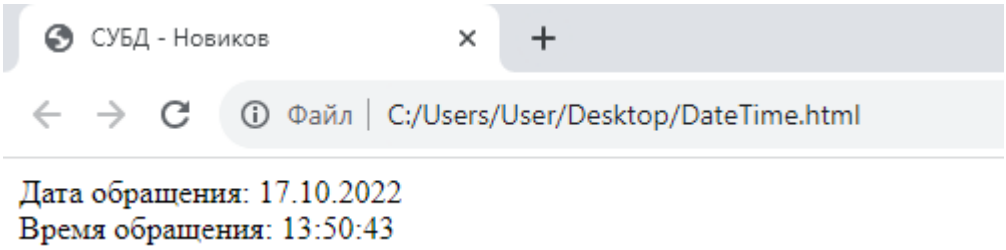


Рис. 95. Пример **HTML**-страницы, открытой в браузере

На месте всех подставляемых значений требуется написать код:

```
<?php echo $Имя_переменной;?>
```

Этот код выводит значение переменной при помощи команды «эхо». В нашем случае таких переменных две, это:

```
<?php echo $StartDate;?>
```

и

```
<?php echo $StartTime;?>
```

Конечно, чтобы что-то вывести, нужно вначале это «что-то» в переменную записать, поэтому выше в коде необходимо задать значение переменной:

```
$Имя_переменной = ...;
```

Стоит обратить внимание, что имена переменных в PHP всегда начинаются со знака доллара «**$**».

Во всех наших **PHP**-файлах мы будем придерживаться структуры из двух частей:

1. Вначале – код на языке **PHP**, содержащий запись значений всех переменных внутри тега «**<?php** … **?>**». Здесь же будут производиться все необходимые вычисления для получения этих значений (в том числе запрос их из базы данных);
2. Во второй части – код на языке **HTML** с выводом в нужных местах значений командой «**echo**». В команде «**echo**» используются непосредственно имена переменных и не проводятся вычисления, максимум, речь может идти лишь о выполнении здесь простейших операций, таких как взятие элемента массива, округление, указание формата отображения чисел с плавающей точкой (запятой), и т. п.

!!! Без веской на то причины стоит всегда придерживаться именно такой структуры **PHP**-документа. *Вескость причины доказывает автор работы***!**

Тогда рассмотренный вначале **HTML**-документ примет следующий вид **PHP**-документа:

```
<?php
  //Запоминаем текущие дату и время в переменные,
  //чтобы вывести их дальше
  $StartDate = date('d.m.Y');
  $StartTime = date('H:i:s');
?>

<HTML>
  <HEAD> <TITLE>СУБД - Новиков</TITLE> </HEAD>
  <BODY>
      Дата обращения: <?php echo $StartDate;?><BR>
      Время обращения: <?php echo $StartTime;?>
  </BODY>
</HTML>
```

В языке **PHP** имеется несколько вариантов записи комментариев. Мы будем пользоваться комментарием, начинающимся с символов «//» и заканчивающимся в конце строки (или в конце PHP-тега, символами «**?>**»):

```
//Однострочный комментарий
```

Стоит отметить, что несмотря на сказанное вначале (про «подстановку значений»), **PHP** является полноценным языком программирования общего назначения, поддерживающим все типовые средства для программирования, такие, например, как условия, циклы, или массивы.

4.4. SQL-запросы на языке PHP

На самом деле, **SQL** не предназначен для постоянного ручного ввода запросов к базе данных. Он предназначен для автоматизации процесса проведения запросов и используется в составе какого-либо из языков программирования (в нашем случае это будет язык **PHP**).

1. Подключение к серверу

Для выполнения запроса необходимо вначале установить подключение к серверу, которое выполняется командой **mysqli_connect** (Адрес, Логин, Пароль, База_Данных), например:

```
//Подключиться к базе данных
$connect = mysqli_connect("localhost", "root", "", "Novikov");
```

Далее необходимо проверить наличие ошибок соединения:

```
//Завершить работу в случае ошибки
if ($connect == false) {
    print("Невозможно подключиться к MySQL");
    exit;
}
```

Кроме того, необходимо установить кодировку для правильного отображения русских символов:

```
//Установить кодировку
mysqli_set_charset($connect, 'utf8');
```

Естественно, что в конце данное соединение необходимо закрыть командой:

```
//Завершить соединение с базой данных
mysqli_close($connect);
```

2. Выполнение запроса

Запросы **SQL** пишутся между команд на подключение и на отключение (после команды установки кодировки). В одном подключении можно выполнить и несколько **SQL**-запросов.

Рассмотрим простейший запрос к таблице «**Test_tbl**», созданной ранее (см. **раздел 2.3**):

```
//Запросить данные из БД
$sql = "SELECT * FROM Test_tbl WHERE Номер=1";
$result = mysqli_query($connect, $sql);
//Завершить работу в случае ошибки
if ($result == false) {
    print("Ошибка при выполнении запроса<BR>$sql");
    exit;
}
```

Обычно каждый запрос строится из трех частей: текст запроса, команда на выполнение этого запроса и проверка на ошибку.

Язык **PHP** позволяет объединять несколько строковых значений в одно, это делается через символ точка «.». Данная функция позволит правильно оформить **SQL**-запросы в несколько строк:

```
$sql = "SELECT * ".
       "FROM lab2 ".
       "WHERE Номер=1";
```

3. Получение и вывод результата запроса

Описанный выше запрос, в случае успеха, помещает результат в переменную **$result**, при этом результат – это всегда таблица, т. е. двумерный массив (даже если содержит всего одну строку). Данная таблица всегда имеет шапку. Таблица может быть и пустой (содержать ноль строк).

Например, мы можем получить результат запроса для «Номер=1» с одной строкой (рис. 96). Или результат с нулем строк для «Номер=99» (рис. 97). Если вовсе убрать условие WHERE, то получим всю таблицу с 5-ю строками (рис. 98). *Данные таблицы с результатами не выводятся на экран, они хранятся в памяти в переменной* ***$result****, которую и нужно использовать в дальнейшем для отображения результатов.*

Номер	Имя звена
1	Интегрирующее

Рис. 96. Результат запроса с одной строкой

Номер	Имя звена

Рис. 97. Результат запроса с нулем строк (пустая таблица)

Номер	Имя звена
1	Интегрирующее
2	Апериодическое 1-го порядка
3	Апериодическое 2-го порядка
4	Дифференцирующее
5	Транспортное запаздывание

Рис. 98. Результат запроса с несколькими строками

Для того чтобы узнать количество полученных строк, в PHP используется команда:

```
$result->num_rows
```

Для того чтобы получить только одну строку результата (первую), т. е. одномерный массив, необходимо выполнить команду:

```
//Получаем первую строку с результатом
$r = mysqli_fetch_array($result);
```

!!! Данную команду необходимо выполнять, даже если в таблице с результатом заведомо всего одна строка**!**

Теперь во второй части документа (в коде **HTML**) мы сможем получить любое из значений этой строки, используя команду «**echo**», например:

```
<?php echo $r['Имя звена'];?>
```

Что касается имени переменной **$r**, то оно намеренно выбрано коротким, из одной буквы, чтобы не загромождать в дальнейшем код HTML многочисленным ее использованием. *Стоит обратить внимание, что если выполняются несколько запросов, то и имен переменных (**$result** и **$r**) для них может понадобиться несколько разных, иначе мы рискуем затереть предыдущие значения новыми. При этом имена таким переменным стоит давать более осмысленные, отражающие суть хранящихся в них данных, например, **$markets**, а переменную, содержащую одну строчку, по-прежнему стараться назвать коротким именем, например, просто **$m**.*

Запрос по номеру хорош тем, что мы уверены, что нам будет выдано не более одной строки (т. к. поле «**Номер**» помечено как уникальное). Но при этом нам может быть выдано и менее одной строки, т. е. ноль строк (когда строка с таким номером не существует). Попытка получить значения таких несуществующих полей приведет к множественным ошибкам (рис. 99). Поэтому необходимо проверять количество полученных строк:

```
//Проверяем, что найдена хотя-бы одна строка
if ($result->num_rows < 1) {
    print("Указан неверный номер строки");
    exit;
}
```

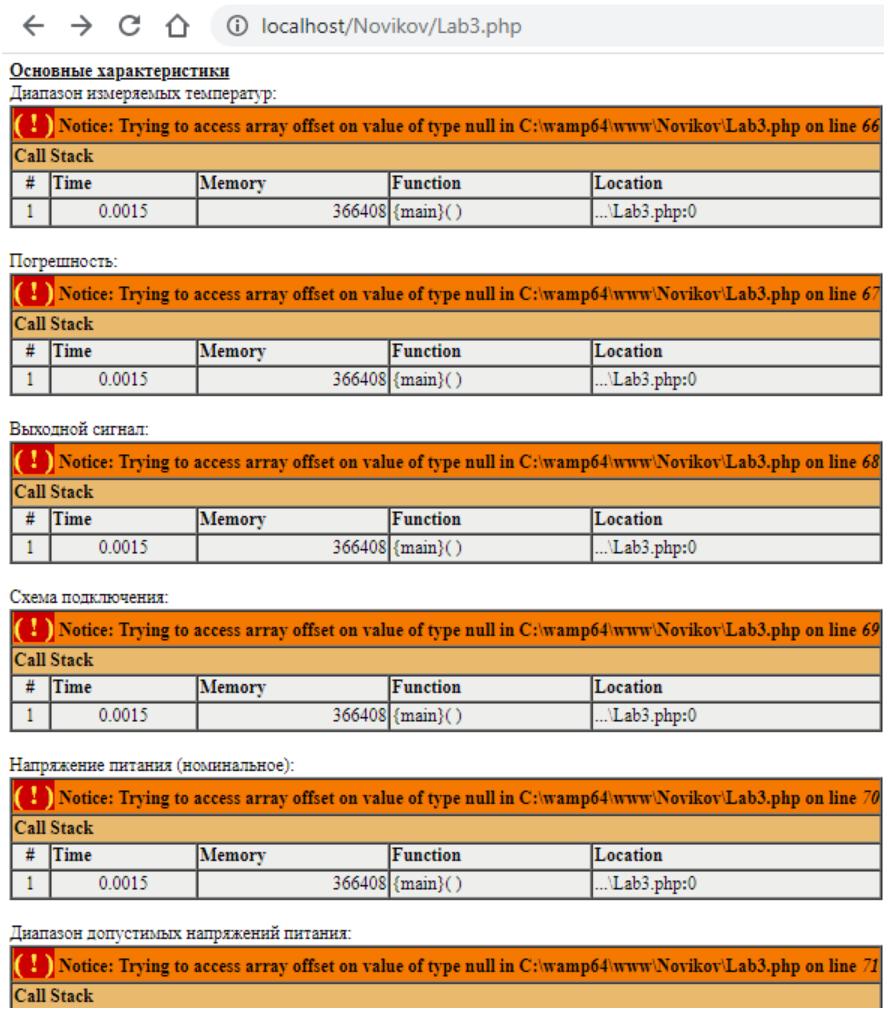


Рис. 99. Множественные ошибки при обращении к пустой таблице

4.5. ЛАБОРАТОРНАЯ РАБОТА № 3 (часть 2)

«Использование SQL в PHP»

!!! *Для дальнейшей работы необходимо вначале завершить выполнение Лабораторной работы 2 и Лабораторной работы 3 (часть 1).*

Создадим копию файла «**Lab3.html**», дав ей имя «**Lab3.php**» (данная операция проводится в папке со своей фамилией, например, «**C:\wamp64\www\Novikov**»).

Далее отредактируем файл «**Lab3.php**» таким образом, чтобы:

– он состоял из двух частей: **PHP** и **HTML**;
– в первой части файла организуем подключение к базе данных (со списком оборудования), созданной в прошлой лабораторной работе;

- разработаем **SQL**-запрос (или несколько запросов), получающий все характеристики для одного из устройств (например, первого в списке) и реализуем его в **PHP**;
- заменим (во второй части файла, в коде **HTML**) все статически выводимые на странице характеристики устройства, соответствующими командами «**echo**», реализовав таким образом получение его характеристик из базы данных. *Данный пункт имеет смысл разбить на два этапа, вначале вывести все характеристики, касающиеся непосредственно устройства (такие как его название, изображение, производитель, погрешность, диапазон измерения, напряжение питания и т. п.), а потом вывести значения из связанных таблиц (такие как, например, магазины и наличие в этих магазинах)*;
- в завершение поменяем в запросе номер устройства (например, на второй), и убедимся в правильности работы и в этом случае;
- кроме того, поменяем номер устройства на заведомо несуществующий (например, 99), и убедимся, что правильно отрабатывается ошибка.

Для проверки работоспособности созданной страницы откроем ее из браузера по адресу «**localhost/Novikov/Lab3.php**». Необходимо выполнять проверку работоспособности страницы на каждом из этапов (и даже чаще!), для этого в браузере необходимо обновить страницу нажатием клавиши **F5**. Стоит напомнить, что перед обновлением страницы в браузере необходимо нажать кнопку «**Сохранить» в блокноте**.

Отчет должен содержать:

- цель и задачи;
- скриншоты таблиц (из Лаб. 2, возможно, немного доработанные), по которым осуществляются запросы;
- описание хода работы (что, как и в какой программе делалось? Со скриншотами);
- код созданных страниц (HTML+PHP+SQL) с выделением синтаксиса разными цветами и скриншоты созданных страниц, в том числе страницы с ошибкой при неправильном вводе id;
- расшифровку терминов, которые встречаются в отчете (но которых не было в предыдущих отчетах, такие как HTML, PHP и т. п.);
- описание использованных команд SQL и PHP;
- титульный лист, номера страниц, оглавление, список использованной литературы (включая Интернет-ресурсы и данную методичку), выводы/заключение и т. п.

4.6. Повторение HTML-кода в PHP-цикле

При выполнении предыдущей часть Лабораторной работы многие могли столкнуться с вопросом «Как вывести список магазинов, когда запрос выдает несколько строк?». Для этого необходимо обрабатывать строки в цикле.

Цикл в **PHP** имеет следующую структуру:

```
<?php while ($m = mysqli_fetch_array($markets)):?>
   <!-- Здесь размещается HTML-код, повторяемый в цикле -->
<?php endwhile;?>
```

!!! Данная конструкция используется во второй части нашего файла! Т. е. в коде на языке **HTML**, а не в коде на языке PHP.

Например, следующий код может вывести результат, представленный на рис. 100:

```
<?php while ($m = mysqli_fetch_array($markets)):?>
  <BR><?php echo $m['Город'];?> -
  <b><?php echo $m['Количество'];?> шт.</b>
<?php endwhile;?>
```

г. Екатеринбург - **2 шт.**
г. Москва - **3 шт.**
г. Санкт-Петербург - **7 шт.**

Рис. 100. Вывод информации о трех магазинах

При этом в переменной **$markets** уже должен содержаться список магазинов, который необходимо получить в первой части файла (в коде на языке **PHP**). Список загружается при помощи **SQL**-запроса командой **mysqli_query** (см. раздел **4.4**).

4.7. *PHP-условия в HTML-коде

Аналогично циклам **while** можно использовать и условия **if-then** или **if-then-else**. Условия имеют следующую структуру:

```
<?php if (/*Условие*/):?>
   <!-- HTML-код при True -->
<?php else:?>
   <!-- HTML-код при False -->
<?php endif;?>
```

Например, следующий код может вывести разные надписи (и разными цветами) при разном количестве имеющегося оборудования

```php
<?php if ($count >= 50):?>
  <FONT color = "#00FF00"><B>(много)</B></FONT>
<?php else:?>
  <FONT color = "#FFC000"><B>(мало)</B></FONT>
<?php endif;?>
```

Использование «**else**» является необязательным.

4.8. Ввод данных на страницу

Для ввода данных на страницу необходимо вначале **PHP**-кода использовать следующую команду:

```php
<?php
  //Номер отображаемого датчика
  //(например: localhost/Novikov/Lab3.php?id=1 )
  $id = !empty($_GET['id']) ? $_GET['id'] : 1
  …
```

Теперь в **SQL**-запросах вместо «Номер=1», «Номер=2», … «Номер=99», необходимо использовать код «**Номер=$id**».

Для вызова страницы с требуемым номером необходимо использовать адрес вида:

localhost/Novikov/Lab3.php**?id=1**

В итоге изменяя **id** в адресе, мы сможем получить разное содержимое страницы (рис. 101-104).

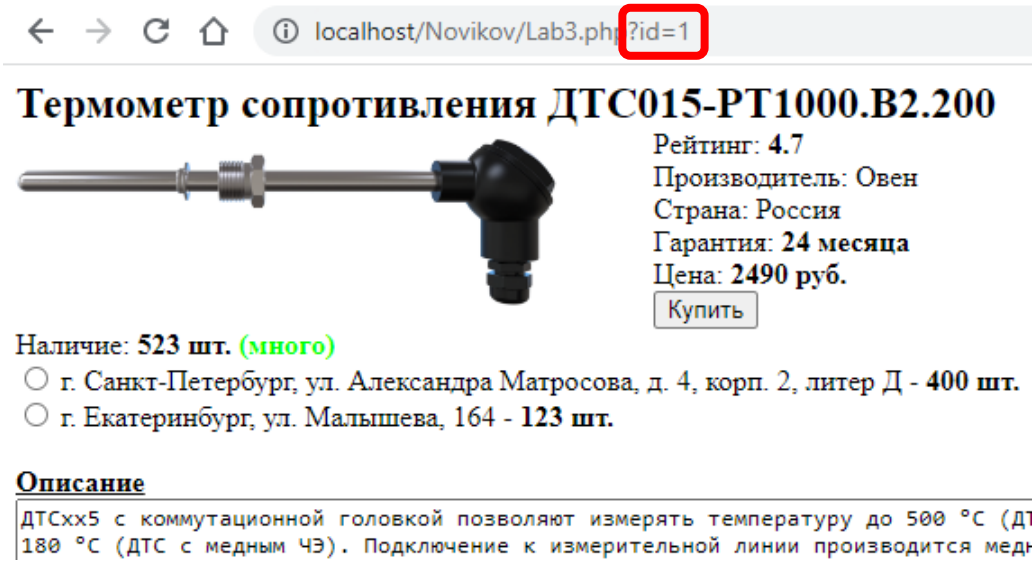


Рис. 101. Страница с id=1

Рис. 102. Страница с id=2

Рис. 103. Страница с id=3

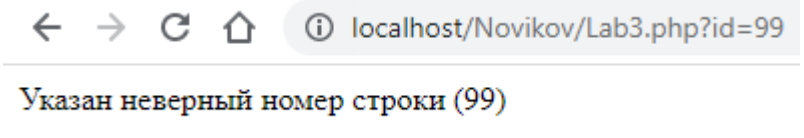


Рис. 104. Страница с id=99

4.9. ЛАБОРАТОРНАЯ РАБОТА № 3 (часть 3)

«Использование SQL в PHP»

В продолжение предыдущей части работы необходимо выполнить следующее:

- доделать вывод списка магазинов (складов) при помощи цикла;
- дополнить страницу, разработанную в частях 1 и 2 данной Лабораторной работы, вводом **id** в адресной строке браузера;
- проверить работоспособность страниц для всех **10**-ти возможных **id**, вводимых в адресную строку, а также проверить правильность отработки ошибки при вводе **id** для несуществующей страницы.

Содержание отчета описано в части 2 данной Лабораторной работы.

4.10. *ЛАБОРАТОРНАЯ РАБОТА № 3 (часть 4)

«Использование SQL в PHP»

!!! *В настоящем разделе содержатся дополнения к Лабораторной работе 3, предназначенные для углубленного изучения, и таким образом, не являющиеся обязательными для выполнения всеми студентами.*

В проделанной ранее Лабораторной работе необходимо модернизировать следующее:

– добавить вывод звезд рейтинга, в том числе нецелых (рис. 105). Для этого необходимо заранее подготовить несколько изображений (например, «Star0.png», «Star25.png», «Star50.png», «Star75.png» и «Star100.png») и разместить их в папке «**Img**»;

Рис. 105. Пример отображения звезд рейтинга

– добавить вывод фразы «Нет в наличии», и при этом скрывать кнопку «Купить»;
– добавить слова «Много» и «Мало» в зависимости от имеющегося в наличии количества оборудования (например, меньше 50 шт.);
– Добавить вывод текста разными цветами в зависимости от условия (например, «**Много**» – зеленым, «**Мало**» – желтым, «**Нет в наличии**» – красным);
– скрывать в списке характеристики, которые для этого устройства не заданы (поля таблицы пустые);
– уменьшать количество товара на единицу после нажатия кнопки «Купить»;
– в папке «**Admin**» создать файл «**index.php**», позволяющий администратору (заходя по адресу «**localhost/Novikov/Admin**») увеличивать количество товара (имитировать поступление товара на склад).

5. СЛОЖНЫЕ SQL-ЗАПРОСЫ

5.1. Создание страницы

Создадим (пока без использования PHP и MySQL) **HTML**-страницу «**Lab4.html**», в которой добавим список из устройств (пока достаточно 3 шт.), аналогичный представленному на рис. 106. По каждому из устройств должны отображаться:

- наименование устройства;
- изображение устройства (изображение должно иметь маленький размер, причем для всех устройств необходимо отображать изображения одинаковых размеров);
- цена устройства;
- наличие устройства;
- рейтинг устройства;
- 1-2 основные характеристики устройства.

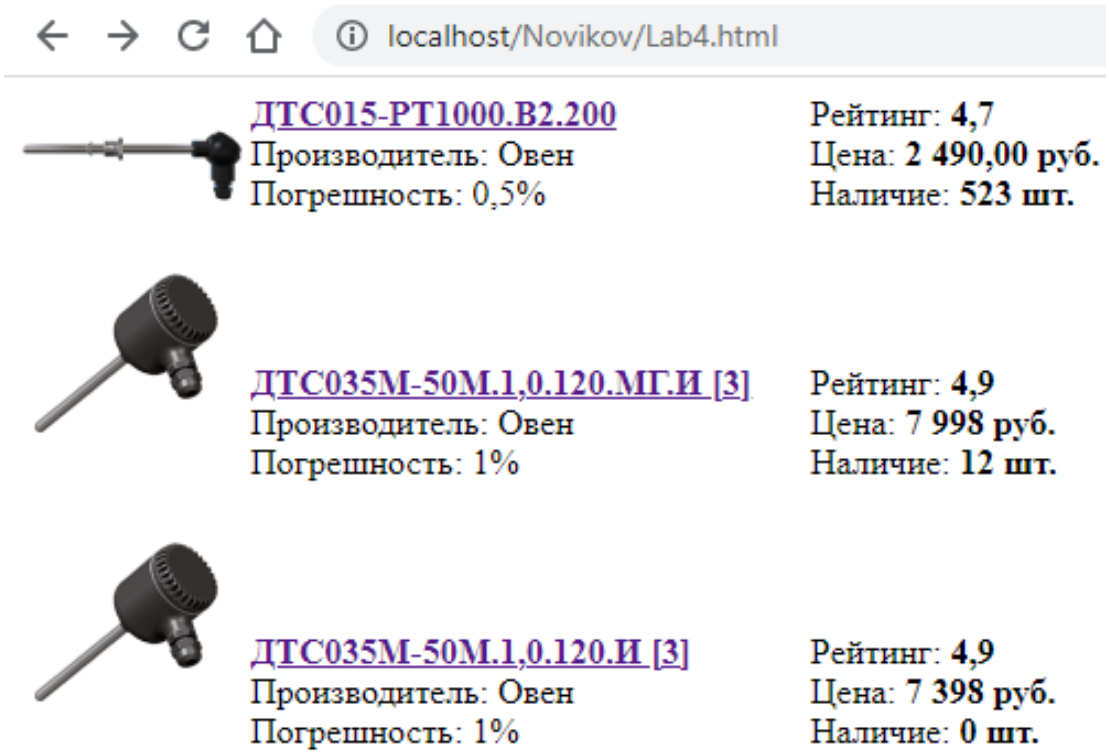


Рис. 106. Страница со списком устройств

Внешний вид может отличаться и выглядеть, например, как на рис. 107. Если требуется оформить список как на рис. 106, то необходимо ознакомиться с разделом **3.3**.

Рис. 107. Вариант отображения списка устройств

Также необходимо создать ссылки для перехода на страницы соответствующих устройств. Переход происходит при нажатии на название устройства или его изображение, например, как представлено на рис. 108.

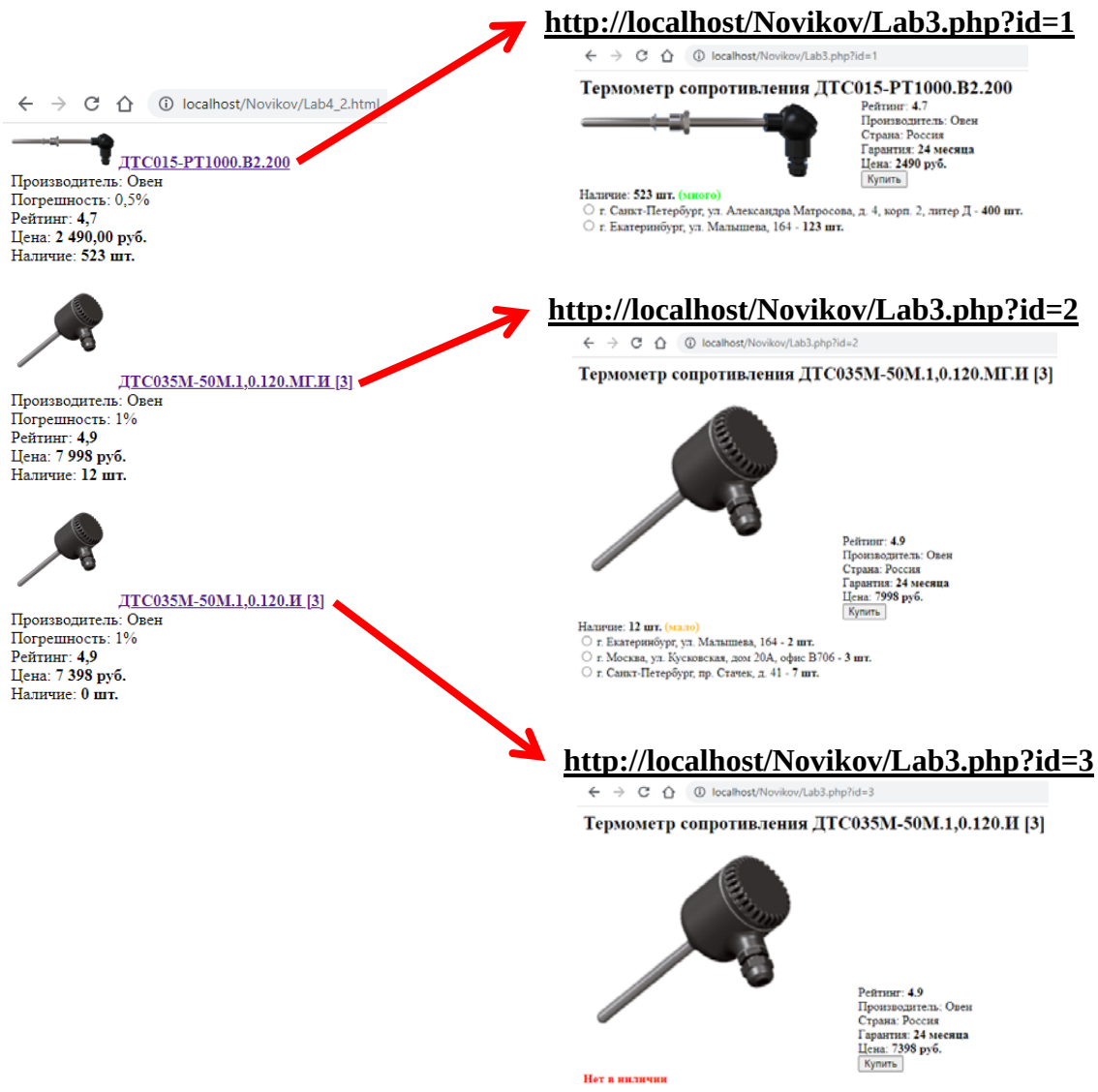


Рис. 108. Переходы на страницы устройств

Когда данная страница создана, переименуем файл в «**Lab4.php**» и добавим в его начало код подключения к базе данных **MySQL** (см. раздел **4.4**). После этого необходимо реализовать запрос, выводящий устройства в заданном диапазоне цен, выстроенные по рейтингу. Для вывода на страницу нескольких строк результата, оформленных в одном стиле, необходимо использовать циклы, описанные в разделе **4.6**.

Стоит обратить внимание, что в отличие от Лабораторной работы 3, здесь на страницу выводятся не все данные, имеющиеся в таблице, следовательно, использовать запрос со звездочкой «**SELECT ***» будет неоправданным расточительством. *Хотя это и не будет заметно при выполнении одного запроса один раз в несколько секунд, но при выполнении сотен и тысяч запросов в секунду от многих пользователей, это значительно снизит производительность сервера.* Требуется вместо звездочки использовать список столбцов, которые участвуют в отображении информации на странице.

Рекомендуется вначале выполнить данную работу без вывода из БД количества шт. в наличии. А потом доработать запрос при помощи операторов **SUM** и **GROUP BY**.

5.2. Агрегатные функции и группировка

Рассмотрим пример таблицы **test3_tbl**, представленный на рис. 109.

Номер	Имя	Количество
1	Wi-Fi роутер Mercusys MW301R	25
2	Wi-Fi роутер HUAWEI WS5200-21	7
3	Wi-Fi роутер HUAWEI WS5200-21	10
4	Wi-Fi роутер Mercusys MW301R	5
5	Wi-Fi роутер TP-LINK TL-WR844N	27

Рис. 109. Исходная таблица

Для определения суммарного значения применяется агрегатная функция **SUM**, которая используется в запросе следующим образом:

```
SELECT SUM(Количество)
FROM test3_tbl;
```

Результат выполнения данного запроса представлен на рис. 110.

SUM(Количество)
74

Рис. 110. Результат суммирования

При таком способе невозможно вывести содержимое других столбцов (без агрегатных функций), например, запрос:

```
SELECT Номер, Имя, SUM(Количество)
FROM test3_tbl;
```

вернет результат, представленный на рис. 111, который не соответствует действительности, т. к. **Количество** посчитано по всем устройствам, а **Имя** и **Номер** отображены только для первого из них, что, конечно же, неверно.

Номер	Имя	SUM(Количество)
1	Wi-Fi роутер Mercusys MW301R	74

Рис. 111. Неверный подсчет количества

Обычно **SUM** используется совместно с **GROUP BY**:

```
SELECT Имя, SUM(Количество)
FROM test3_tbl
GROUP BY Имя;
```

Здесь мы можем вывести и столбцы без агрегатных функций, но только те столбцы, по которым ведется группировка**!** Результат такого запроса представлен на рис. 112.

Имя	SUM(Количество)
Wi-Fi роутер HUAWEI WS5200-21	17
Wi-Fi роутер Mercusys MW301R	30
Wi-Fi роутер TP-LINK TL-WR844N	27

Рис. 112. Суммирование по группам

Агрегатную функцию **SUM** невозможно использовать в условиях **WHERE**, например, нельзя написать:

```
WHERE SUM(Количество) > 20
```

Аналогом **WHERE** для агрегатных функций является **HAVING**:

```
SELECT Имя, SUM(Количество)
FROM test3_tbl
GROUP BY Имя
HAVING SUM(Количество) > 20;
```

В этом случае мы получим результат, представленный на рис. 113.

Имя	SUM(Количество)
Wi-Fi роутер Mercusys MW301R	30
Wi-Fi роутер TP-LINK TL-WR844N	27

Рис. 113. Условие HAVING

Аналогично **SUM**, можно использовать и другие агрегатные функции **MIN**, **MAX**, **AVG** и **COUNT** – для поиска минимального, максимального и среднего значения, а также для определения количества найденных записей (строк).

5.3. Объединение таблиц (JOIN)

Оператор **JOIN** используется для объединения таблиц. Виды объединения таблиц представлены на рис. 114.

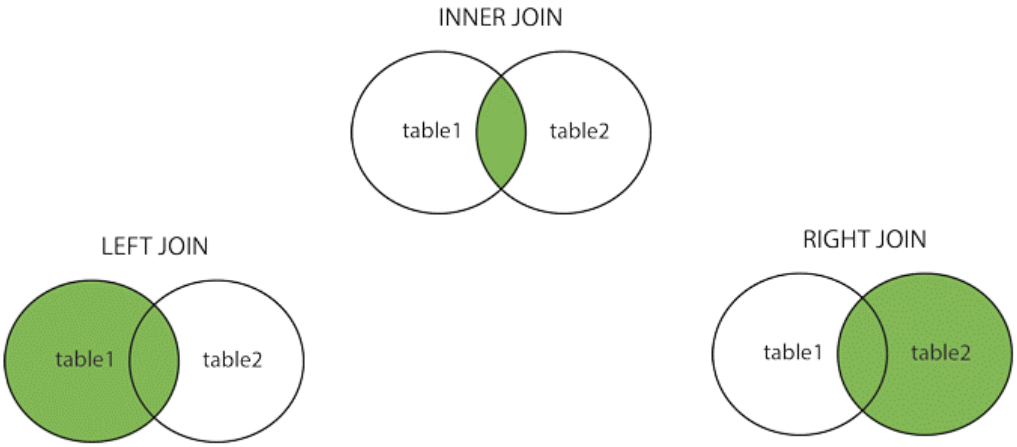


Рис. 114. Виды объединения таблиц

INNER JOIN[12] – выбирает записи, имеющие совпадающие значения в обеих таблицах. *Записи, не имеющие совпадений, отбрасываются.*

FROM Таблица1 **INNER JOIN** Таблица2
ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

Оператор **JOIN** указывается после оператора **FROM** фактически вместо запятой в перечислении таблиц. Ранее мы писали:

FROM Таблица1**,** Таблица2

Оператор **JOIN** используется совместно с оператором **ON**, после которого указываются имена связываемых столбцов (аналогично тому, как это делалось при связывании через **WHERE**).

LEFT JOIN[13] (он же **LEFT OUTER JOIN**) – выбирает все записи из левой таблицы (Таблица 1) и соответствующие ей записи из правой таблицы (Таблица 2). *Записи из правой таблицы (Таблица 2), не имеющие совпадений, отбрасываются.*

FROM Таблица1 **LEFT JOIN** Таблица2
ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

RIGHT JOIN[14] (он же **RIGHT OUTER JOIN**) – выбирает все записи из правой таблицы (Таблица 2) и соответствующие ей записи из левой таблицы (Таблица 1). *Записи из левой таблицы (Таблица 1), не имеющие совпадений, отбрасываются.*

FROM Таблица1 **RIGHT JOIN** Таблица2
ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

5.4. Пример объединения таблиц (JOIN)

Рассмотрим пример двух таблиц, **test4_1** (рис. 115) и **test4_2** (рис. 116).

Номер	Наименование	...
1	Wi-Fi роутер Mercusys MW301R	...
2	Wi-Fi роутер HUAWEI WS5200-21	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...
4	Коммутатор D-Link DES-1005C/B	...
5	Коммутатор Zyxel GS-108S v2	...

Рис. 115. Пример таблицы с устройствами

No	Номер устройства	Город	Количество	Цена	...
1	1	Санкт-Петербург	37	1199	...
2	1	Москва	54	1099	...
3	1	Мурманск	3	1308	...
4	2	Санкт-Петербург	0	2799	...
5	3	Москва	7	1599	...

Рис. 116. Пример таблицы с количеством устройств

Если по этим таблицам выполнить следующий запрос с **INNER JOIN**, то получим таблицу, представленную на рис. 117 (зеленой вертикальной линией показана граница между двумя исходными таблицами):

```
SELECT *
FROM test4_1 INNER JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`;
```

Номер	Наименование	...	No	Номер устройства	Город	Количество	Цена	...
1	Wi-Fi роутер Mercusys MW301R	...	1	1	Санкт-Петербург	37	1199	...
1	Wi-Fi роутер Mercusys MW301R	...	2	1	Москва	54	1099	...
1	Wi-Fi роутер Mercusys MW301R	...	3	1	Мурманск	3	1308	...
2	Wi-Fi роутер HUAWEI WS5200-21	...	4	2	Санкт-Петербург	0	2799	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...	5	3	Москва	7	1599	...

Рис. 117. Пример запроса INNER JOIN

Данная таблица получена путем объединения двух исходных таблиц по схеме, представленной на рис. 118.

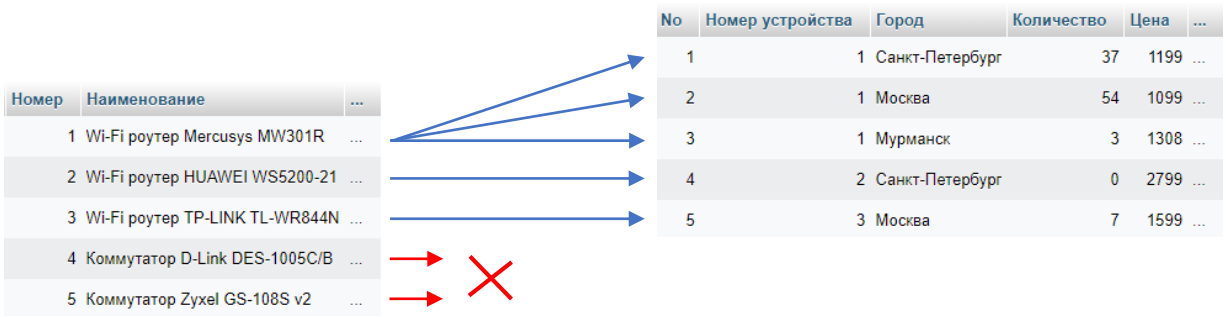


Рис. 118. Схема объединения таблиц через INNER JOIN

Из первой таблицы (**test4_1**) берутся строки с номерами 1, 2 и 3 (некоторые даже не по одному разу), т. к. ссылки на них присутствуют во второй таблице (**test4_2**). Устройства с номерами 4 и 5 отбрасываются, т. к. для них нет соответствующих записей во второй таблице (**test4_2**).

Если выполнить тот же запрос, но с **LEFT JOIN**, то получим таблицу, представленную на рис. 119:

```
SELECT *
FROM test4_1 LEFT JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`;
```

Номер	Наименование	...	No	Номер устройства	Город	Количество	Цена	...
1	Wi-Fi роутер Mercusys MW301R	...	1	1	Санкт-Петербург	37	1199	...
1	Wi-Fi роутер Mercusys MW301R	...	2	1	Москва	54	1099	...
1	Wi-Fi роутер Mercusys MW301R	...	3	1	Мурманск	3	1308	...
2	Wi-Fi роутер HUAWEI WS5200-21	...	4	2	Санкт-Петербург	0	2799	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...	5	3	Москва	7	1599	...
4	Коммутатор D-Link DES-1005C/B	...	NULL	NULL	NULL	NULL	NULL	NULL
5	Коммутатор Zyxel GS-108S v2	...	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 119. Пример запроса LEFT JOIN

Здесь мы видим, что из первой таблицы были подставлены все возможные строки (некоторые даже не по одному разу). Из второй таблицы были подставлены строки, имеющие совпадения с первой таблицей. Недостающие значения в правой части (показаны красной рамкой) были заполнены значениями **NULL**.

Аналогично, используя агрегатные функции и группировку, можно выполнить следующий запрос с **INNER JOIN** (рис. 120):

```
SELECT test4_1.Номер, Наименование,
       sum(Количество) AS `Кол-во`, min(Цена) AS `Мин. цена`
FROM test4_1 INNER JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`
GROUP BY Номер
ORDER BY sum(Количество) DESC;
```

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799

Рис. 120. Пример запроса INNER JOIN с группировкой

Проблемой является то, что устройства с номерами 2, 4 и 5 отсутствуют в наличии в нашем магазине (их 0 шт.), но при этом устройство с номером 2 попало в таблицу с результатами (т. к. в исходных таблицах для него явно указано количество **0**), а устройства с номерами 4 и 5 не попали (т. к. в исходных таблицах для них нет записей с количеством).

Тот же запрос, использующий агрегатные функции и группировку, но с **LEFT JOIN**, будет выглядеть следующим образом (рис. 121):

```
SELECT test4_1.Номер, Наименование,
        sum(Количество) AS `Кол-во`, min(Цена) AS `Мин. цена`
FROM test4_1 LEFT JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`
GROUP BY Номер
ORDER BY sum(Количество) DESC;
```

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799
4	Коммутатор D-Link DES-1005C/B	*NULL*	*NULL*
5	Коммутатор Zyxel GS-108S v2	*NULL*	*NULL*

Рис. 121. Пример запроса LEFT JOIN с группировкой

Были выведены все устройства, но и здесь имеется проблема, часть значений для отсутствующих в продаже устройств отображается как **0**, а часть как **NULL**.

5.5. Функция COALESCE

Функция **COALESCE**[15] принимает на вход два (или более) параметра, возвращая первый ненулевой (≠ **NULL**) из них.

Таким образом, мы можем использовать данную функцию для присвоения значения по умолчанию (например, значения **0**, **'нет'** или пустой строки **''**) в случае, если база данных выдает значение **NULL**. *Причем данные манипуляции осуществляются на стороне **SQL**-сервера, и нет необходимости в дополнительных проверках (**IF**) полученных значений на стороне **PHP**, или на стороне клиента (**JavaScript**).*

Будем использовать эту функцию в шапке запроса **SELECT**, где идет перечисление запрашиваемых столбцов, например, вместо:

SELECT … sum(Количество) … min(Цена) …

теперь используем:

SELECT … **COALESCE** (sum(Количество), 0) … **COALESCE** (min(Цена), '') …

Тогда в итоге доработанный предыдущий запрос вернет нам уже не значения **NULL**, а подставит значения **0** (рис. 122) или пустую строку **''**.

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799
4	Коммутатор D-Link DES-1005C/B	0	
5	Коммутатор Zyxel GS-108S v2	0	

Рис. 122. Итоговая таблица

!!! *Функция **COALESCE** может применяться не только совместно с объединением таблиц (**JOIN**) или агрегатными функциями (**SUM**, **COUNT**, **AVG**, **MIN**, **MAX**). Столбцы, которые не помечены как **NOT NULL**, также могут содержать значения **NULL**. Таким образом, для применения данной функции достаточно и всего одной таблицы.*

5.6. Постраничный вывод и ограничение выборки (LIMIT)

Запрос может выдавать сотни и тысячи строк результатов. При этом все результаты могут не поместиться на одной странице либо вывод всех результатов может быть нецелесообразен (например, товары отсортированы по возрастанию цены, тогда пользователя могут просто не интересовать самые дорогие из товаров, которые будут расположены в конце таблицы с результатами).

Для ограничения выборки используется ключевое слово **LIMIT**. Например, следующий запрос выдаст только первые 3 строки с результатами:

```
SELECT *
FROM test3_tbl
LIMIT 3;
```

Ключевое слово **LIMIT** может использоваться и с двумя параметрами:

```
LIMIT количество_пропущенных_записей, вывести_записей
```
или
```
LIMIT вывести_записей OFFSET количество_пропущенных_записей
```

Например, следующие записи выводят строки результатов с 21 по 30:

```
LIMIT 20, 10
```
или
```
LIMIT 10 OFFSET 20
```

Причем запись с одним параметром, имеющая вид:

LIMIT 10

равносильна записям:

LIMIT 0, 10
или **LIMIT** 10 **OFFSET** 0

Стоит отметить, что ключевое слово **LIMIT** является командой не **SQL**, а только **MySQL** (а также **MariaDB**). *В СУБД других производителей тот же функционал реализуется по-другому, например, командой **SELECT TOP**, или при помощи системной переменной **ROWNUM**, записываемой в условии **WHERE**.*

5.7. ЛАБОРАТОРНАЯ РАБОТА № 4 (часть 1)

«Сложные SQL-запросы»

Требуется создать **PHP**-страницу, аналогичную описанной в разделе **5.1**. При этом страница должна содержать ссылки на страницы, разработанные в Лабораторной работе № 3 (с разными **id**).

Данные для отображения страницы должны запрашиваться из БД (созданной в Лабораторной работе № 2). Необходимо получить все требуемые данные одним запросом.

!!! В этой работе запрещается использовать символ звездочки «*****» в запросах Select**!**

В первой части данной работы достаточно использовать жестко прописанные в коде запросы. При этом необходимо проверить работу как минимум для двух случаев:

- вывести устройства в заданном диапазоне цен, выстроенные по рейтингу;
- вывести устройства с заданной погрешностью (либо указанного производителя, если погрешность для данных устройств неприменима), имеющиеся в наличии, выстроенные по возрастанию цены.

!!! В базе данных должно иметься хотя-бы два устройства, которых нет в наличии (причем для одного должно быть явно указано 0 шт., а для другого должны отсутствовать записи о количестве). *Для этого может понадобиться отредактировать данные в таблицах, созданных в Лабораторной работе № 2.*

5.8. Элементы управления в HTML

Кроме элемента <button> – кнопка (рис. 123а), язык **HTML** содержит и другие элементы управления, перечисленные ниже.

1. Поле для ввода текста – рис. 123б: <**input** type="text">.
2. Флажок (галочка, чекбокс) – рис. 123в: <**input** type="checkbox">.
3. Радиокнопка (переключатель) – рис. 123г: <**input** type="radio">.
4. Текстовая область с многострочным текстом – рис. 123д: <**textarea**>.
5. Однострочный список – рис. 123е: <**select** size=1>.
6. Многострочный список – рис. 123ё: <**select** size=3>.
7. Многострочный список с множественным выбором – рис. 123ж: <**select** size=3 **multiple**>.

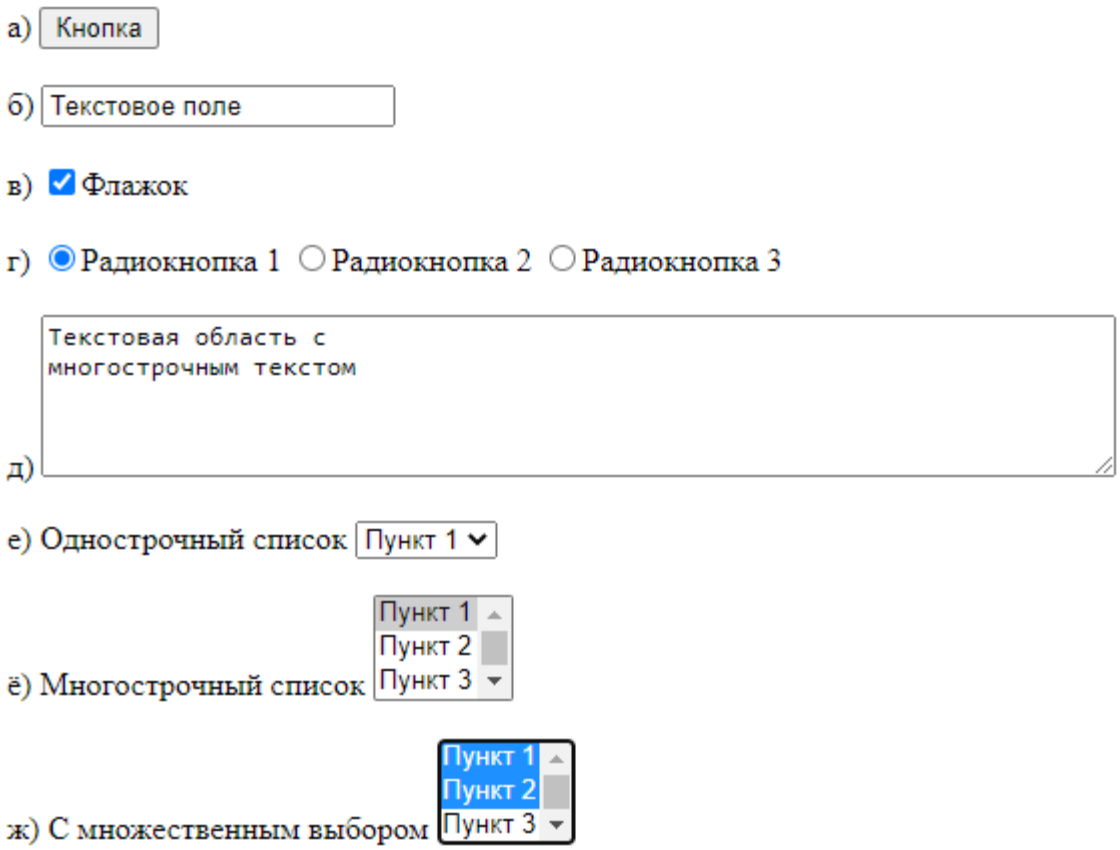


Рис. 123. Элементы управления

Код, по которому были созданы элементы управления на рис. 123, представлен далее:

```
а) <button>Кнопка</button> <BR><BR>
б) <input type="text" value="Текстовое поле"> <BR><BR>
в) <input type="checkbox" checked>Флажок <BR><BR>
г) <input type="radio" name="aaa" value="val1" checked>Радиокнопка 1
   <input type="radio" name="aaa" value="val2">Радиокнопка 2
   <input type="radio" name="aaa" value="val3">Радиокнопка 3
   <BR><BR>
д) <textarea cols=70 rows=5>Текстовая область с
```

```
многострочным текстом</textarea> <BR><BR>
е) Однострочный список
   <select size=1>
       <option selected>Пункт 1</option>
       <option>Пункт 2</option>
   </select> <BR><BR>
ё) Многострочный список
   <select size=3>
       <option selected>Пункт 1</option>
       <option>Пункт 2</option>
       <option>Пункт 3</option>
       <option>Пункт 4</option>
   </select> <BR><BR>
ж) С множественным выбором
   <select size=3 multiple>
       <option selected>Пункт 1</option>
       <option selected>Пункт 2</option>
       <option>Пункт 3</option>
       <option>Пункт 4</option>
   </select>
```

5.9. HTML-формы

Для того, чтобы элементы управления выполняли свою функцию, необходимо «обернуть» их тегами **<FORM>** и **</FORM>**, например:

```
<FORM>
  <select size=1 name="sort">
    <option value="rating">По рейтингу</option>
    <option value="price" selected>По возрастанию цены</option>
    ...
  </select>

  <input type="checkbox" name="available" checked>В наличии
  ...
  <input type="text" name="name">
  <button>Поиск</button>
</FORM> <BR>
```

В таком случае, после нажатия кнопки **Button** мы увидим изменения в адресной строке, в которой отобразятся введенные параметры запроса (рис. 124).

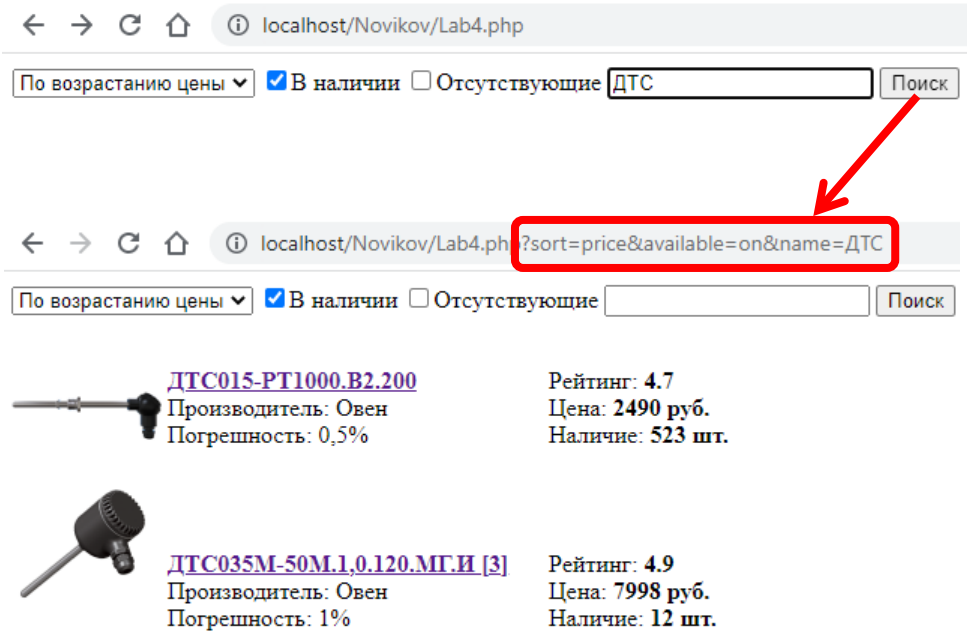


Рис. 124. Передача параметров через адресную строку

Стоит обратить внимание, что в этом случае атрибут **Name** становится обязательным для всех элементов управления, т. к. именно эти имена отображаются в адресной строке.

5.10. Получение значений форм

Получение значений из адресной строки уже упоминалось в разделе **4.8**. Рассмотрим его подробнее:

```
$id = !empty($_GET['id']) ? $_GET['id'] : 1;
```

Здесь **$id** – имя переменной, в которую сохраняется значение;

'id' – имя параметра в адресной строке;

1 – значение по умолчанию, присваиваемое переменной $id в случае, когда значение для 'id' в адресной строке не задано.

Для каждого из параметров, передаваемых через адресную строку, необходимо создать свою переменную.

5.11. Динамическое конструирование SQL-запросов

Трудно представить себе, чтобы сложный запрос можно было составить заранее (статически). Эти запросы должны конструироваться динамически, путем объединения требуемых строк при выполнении определенных условий. Для объединения строк в **PHP** используется точка. Условия записываются при помощи **if**.

Например, для добавления к запросу строки с сортировкой по возрастанию цены можно использовать следующий **PHP**-код:

```
//Сортировка по возрастанию цены
if ($sort == 'price') {$sql = $sql . " ORDER BY lab2.`Цена`";}
```

В данном случае переменная **$sort** со значением «**price**» должна быть получена из адресной строки (см. раздел **5.10**)

5.12. ЛАБОРАТОРНАЯ РАБОТА № 4 (часть 2)

«Сложные SQL-запросы»

Требуется дополнить разработанную страницу формой поиска, например, как на рис. 125, передающей введенные параметры через адресную строку. Добавить к форме поля, требуемые для выполнения отчета (см. содержание отчета ниже).

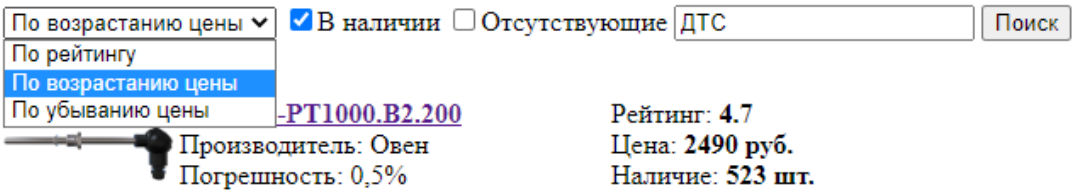


Рис. 125. Пример формы поиска

Дополнить **PHP**-код вводом параметров поиска через адресную строку (через **$_GET** – см. разделы **4.8** и **5.10**) и реализовать использование этих параметров в SQL-запросе.

Значения в элементах управления, доступные для выбора при поиске, могут быть вбиты «жестко» (т. е. их не нужно загружать из базы данных).

Отчет должен содержать:

- цель и задачи;
- скриншоты таблиц (из Лаб. 2, возможно немного доработанных), по которым осуществляются запросы;
- описание хода работы (что, как и в какой программе делалось? Со скриншотами);
- описание новых тегов HTML (использованных элементов управления и их атрибутов); описание использованных команд SQL и PHP;

- код созданных страниц (HTML+PHP+SQL) с выделением синтаксиса разными цветами и скриншоты созданных страниц, на которых видны заполненная форма поиска и результат выполнения этого запроса:
  - устройства в заданном диапазоне цен, выстроенные по рейтингу;
  - устройства с заданной погрешностью (либо указанного производителя, если погрешность для данных устройств неприменима), имеющиеся в наличии, выстроенные по возрастанию цены;
  - устройства, которых нет в наличии;
  - один запрос по выбору автора работы;
  - страница с сообщением «По вашему запросу ничего не найдено» (для случая, в котором по запросу было найдено **0** строк).
- описание параметров адресной строки;
- титульный лист, номера страниц, оглавление, список использованной литературы (включая Интернет-ресурсы и данную методичку), выводы/заключение и т. п.

5.13. *ЛАБОРАТОРНАЯ РАБОТА № 4 (часть 3)

«Сложные SQL-запросы»

В проделанной ранее Лабораторной работе необходимо модернизировать следующее:

- сохранять между запросами значения, заданные в элементах управления. Т. е. не очищать (повторно заполнять) поля ввода после нажатия кнопки «Поиск».
- изменить разработанную страницу так, чтобы значения в элементах управления были вбиты не «жестко», а подтягивались автоматически из вариантов, доступных в базе данных. Например, если в базе имеются устройства с корпусами, выполненными из пластика и из металла, то в выпадающем списке будет указано «пластик» и «металл». Если далее в базу будет добавлено устройство из дерева, то в выпадающем списке автоматически появится вариант «дерево».
- дополнить работу страницей сравнения двух устройств. Причем должна иметься возможность скрыть одинаковые для обоих устройств значения и отобразить только отличия.

В связи с тем, что предметом изучения является **SQL**, а не PHP, то, соответственно, при решении задачи необходимо максимально использовать возможности SQL, а не PHP.

В настоящем разделе содержатся дополнения к Лабораторной работе 4, предназначенные для углубленного изучения, не являющиеся обязательными для выполнения всеми студентами.

5.14. **ЛАБОРАТОРНАЯ РАБОТА № 5

«Работа с файлами в PHP»

Разработать несколько служебных модулей **PHP**, расположенных в папке «**Admin**», выполняющих следующие полезные функции:

- после запуска программа проверяет «битые ссылки» на изображения. Если в таблице (из Лаб. 2) содержатся ссылки на изображения, которых не существует в папке «**Img**», то выводится список этих файлов и номера строк в таблице (**id** датчика), которые ссылаются на несуществующие файлы. Если «битых ссылок» нет, то выводится соответствующее сообщение об успешной работе;
- предыдущий пункт лишь помогал определить проблему, но не исправлял ее. Далее нужно разработать программу, которая заменяет все «битые ссылки», на ссылку на одно изображение «**Null.png**». Также необходимо подготовить данное изображение, на котором в стилизованном виде будет написано «Нет изображения» (или т. п.);
- обратной задачей является проверка неиспользуемых изображений в папке «**Img**». Если на какое-то из изображений больше нет ссылки в базе данных (таблице из Лаб. 2), то его нужно удалить из папки «**Img**». Стоит также обратить внимание, что на файл «**Null.png**» может не быть действующих ссылок, но его в любом случае удалять не нужно.

6. *САМОСТОЯТЕЛЬНОЕ УГЛУБЛЕННОЕ ИЗУЧЕНИЕ

В данной Главе представлена информация, которая будет полезна студенту для самостоятельного углубленного изучения работы **WAMP**-сервера, **Apache**-сервера, **MySQL**-сервера, языка **PHP** и их настройки.

6.1. *Автозапуск WAMP-сервера

По умолчанию, для работы с **WAMP**-сервером его нужно запустить вручную (ярлык на рабочем столе). Автоматический запуск **WAMP**-сервера (**Apache**-сервера и **MySQL**-сервера) может быть полезен, например, после восстановления электропитания (после аварийного отключения). При этом если сервер настроен на работу в глобальной сети (Интернет) или на сохранение в БД показаний датчика, поступающих с контроллера – то такой автоматический перезапуск сервера просто необходим.

Настройку автозагрузки программ можно осуществить при помощи Планировщика заданий Windows. Для запуска **Планировщика заданий** необходимо нажать правую клавишу мышки на значке «Мой компьютер» («Этот компьютер») и выбрать пункт меню «**Управление**» (рис. 126). *Другой способ попасть в Планировщик заданий – это запустить «Панель управления» и в ней выбрать «Система и безопасность» → «Администрирование» → «Расписание выполнения задач».*

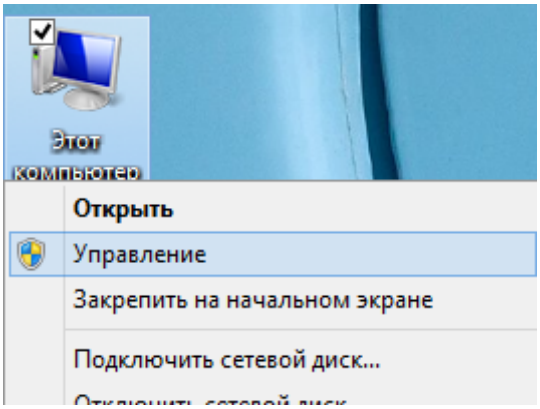


Рис. 126. Вызов Управления компьютером

Далее необходимо выбрать «**Планировщик заданий**» и нажать кнопку «**Создать задачу**» (рис. 127).

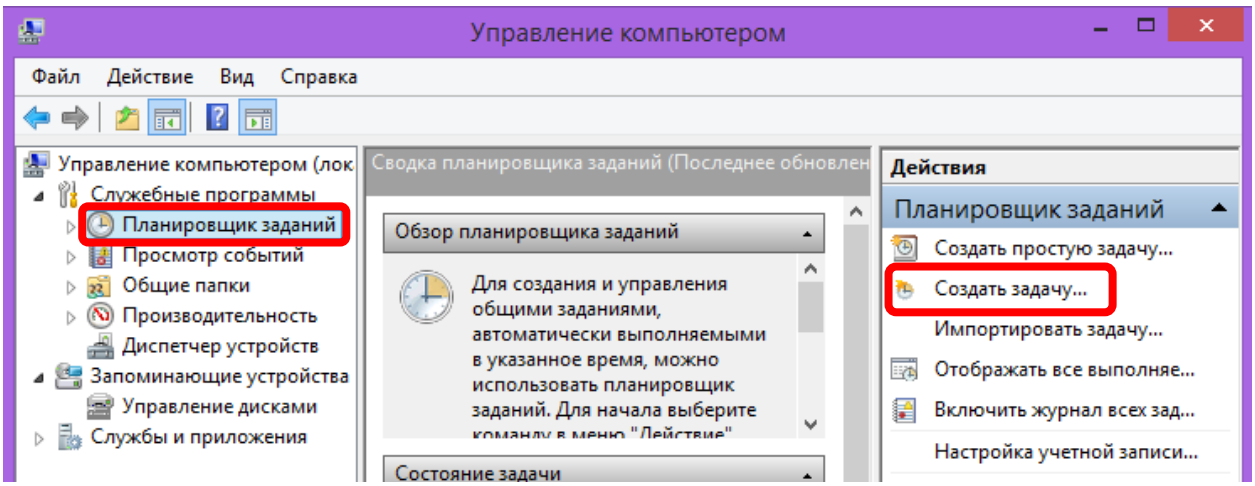


Рис. 127. Создание задачи в планировщике

В открывшемся окне необходимо указать «**Имя**» создаваемой задачи и поставить галочку «**Выполнить с наивысшими правами**» (рис. 128).

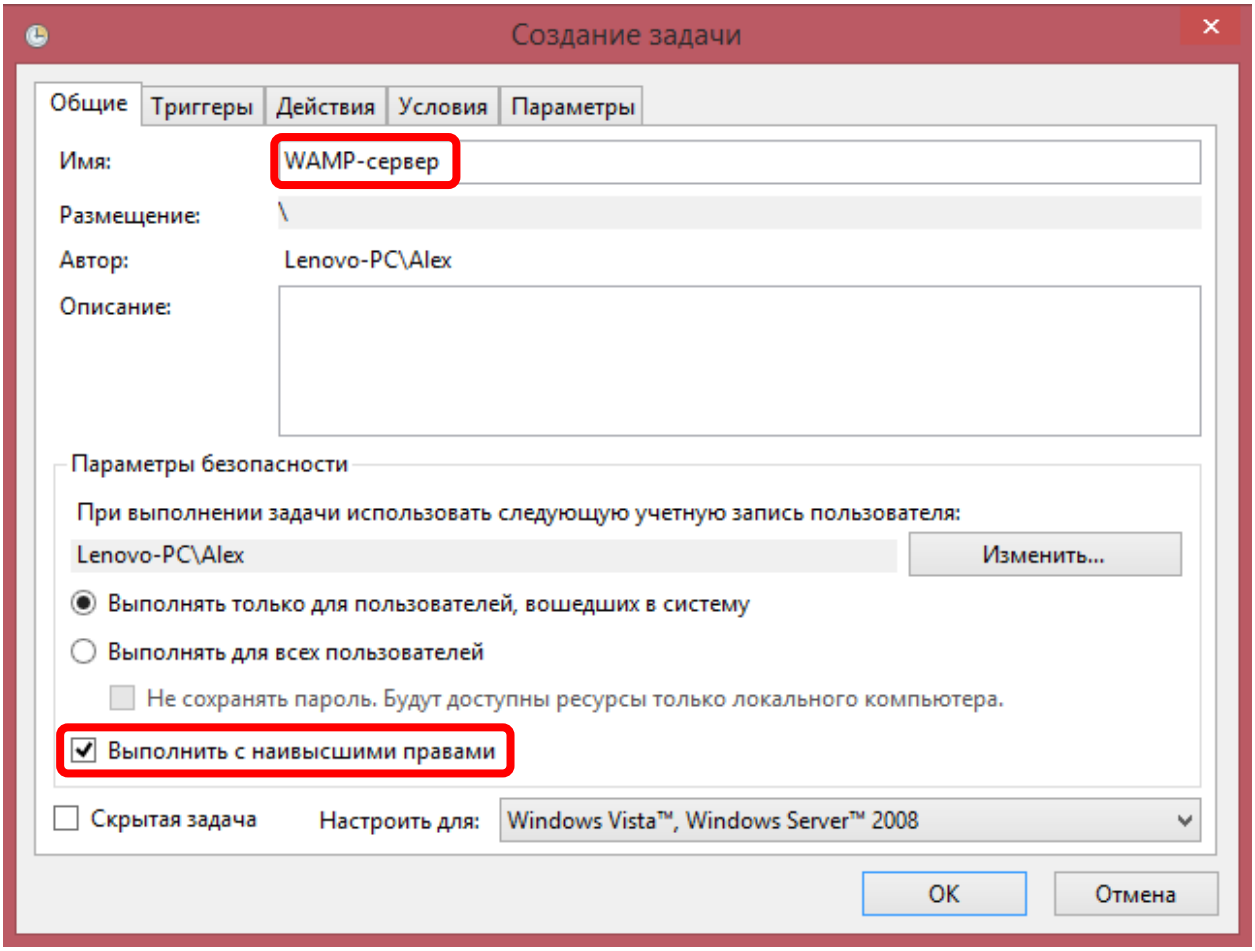


Рис. 128. Настройка создаваемой задачи

Далее переходим на вкладку «**Триггеры**» (рис. 129) и указываем условия запуска программы. Для этого необходимо нажать кнопку «**Создать**», и в появившемся окне указать «**При входе в систему**» и «**Любой пользователь**».

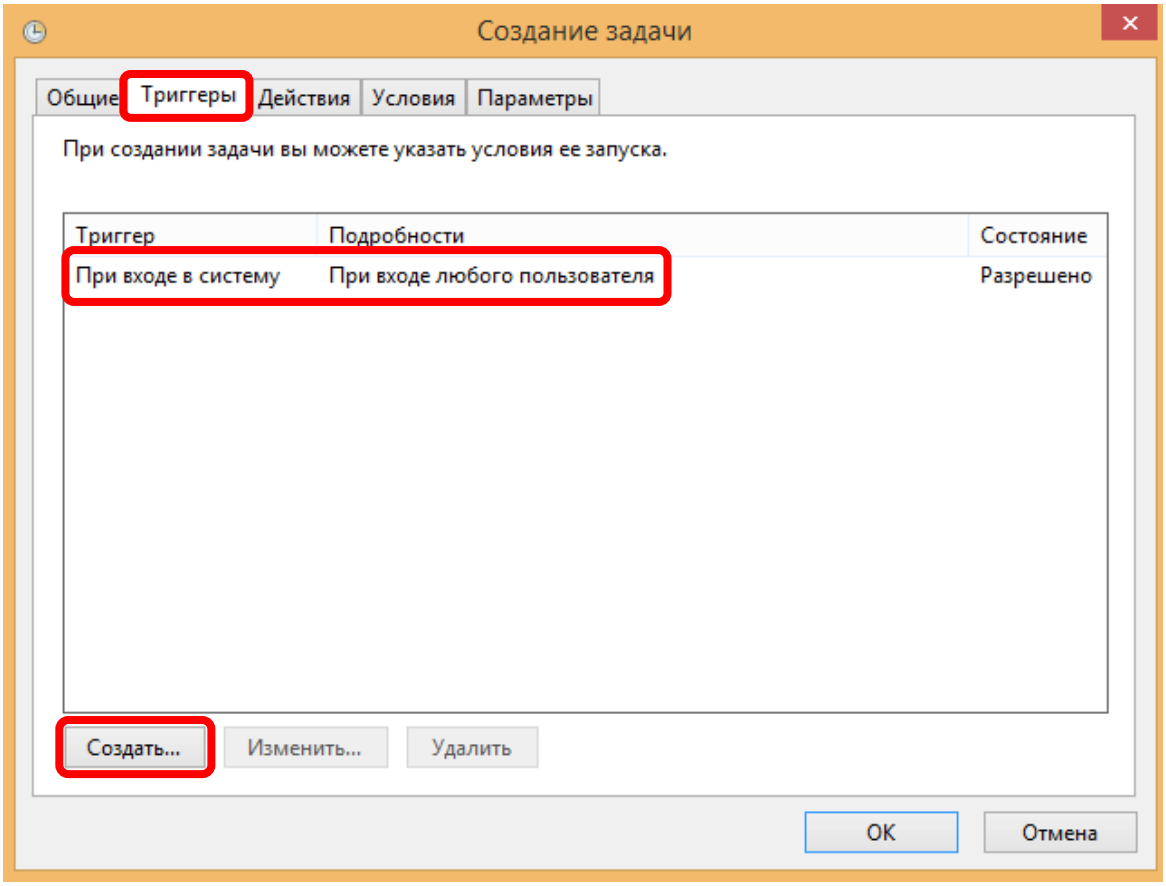


Рис. 129. Настройка условий запуска задачи

На вкладке «**Действия**» (рис. 130) указываем какую программу требуется запустить. Для этого необходимо нажать кнопку «**Создать**», в появившемся окне указать «**Запуск программы**» и, нажав кнопку «Обзор», выбрать адрес до WAMP-менеджера (при установке WAMP-сервера в папку по умолчанию это адрес «**C:\wamp64\wampmanager.exe**»).

На вкладках «**Условия**» (рис. 131) и «**Параметры**» (рис. 132) необходимо снять галочки, как показано на соответствующих рисунках.

После завершения создания задачи необходимо перезагрузить компьютер и убедиться в появлении иконки WAMP-сервера у часов (рис. 133).

Для просмотра и изменения (редактирования, отключения, удаления) созданной задачи необходимо зайти в Планировщик заданий и выбрать «**Библиотека планировщика заданий**» (рис. 134), после чего в списке (по алфавиту) найти созданную ранее задачу.

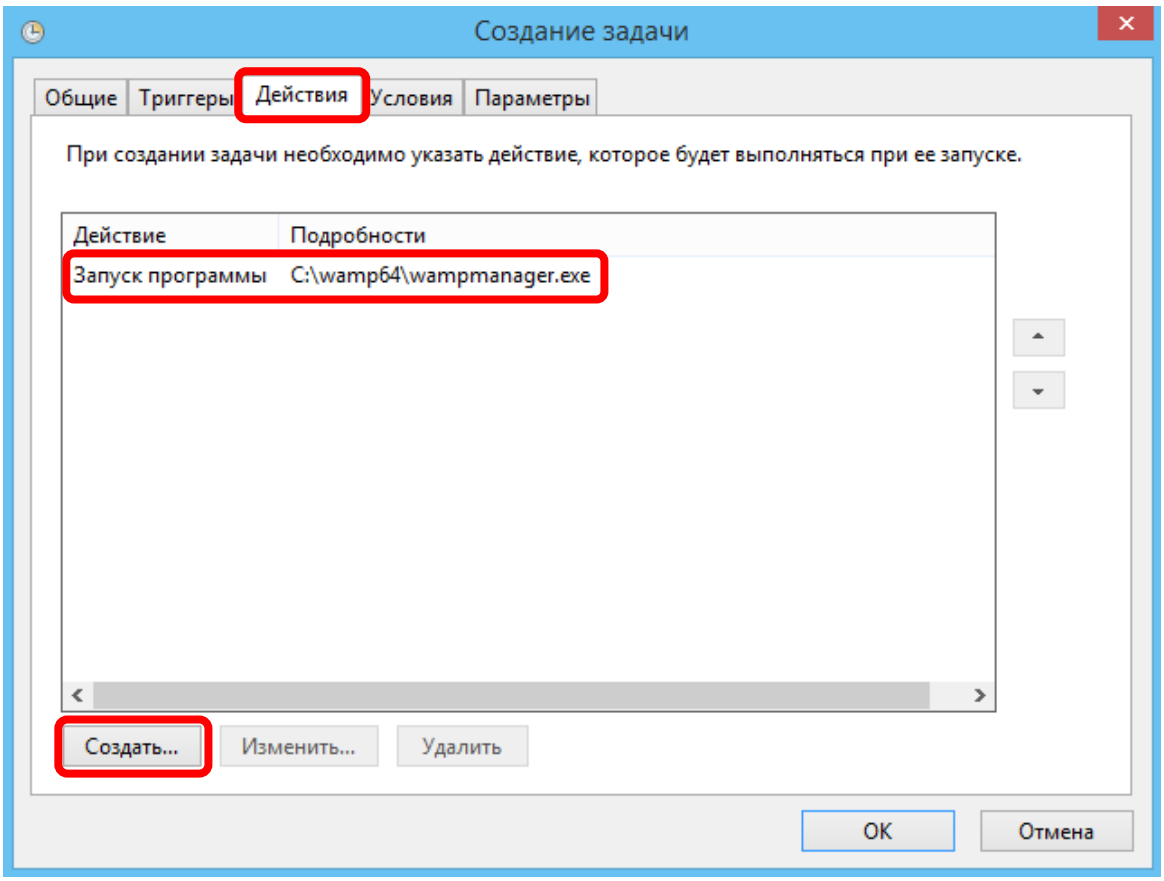


Рис. 130. Настройка действия, выполняемого задачей

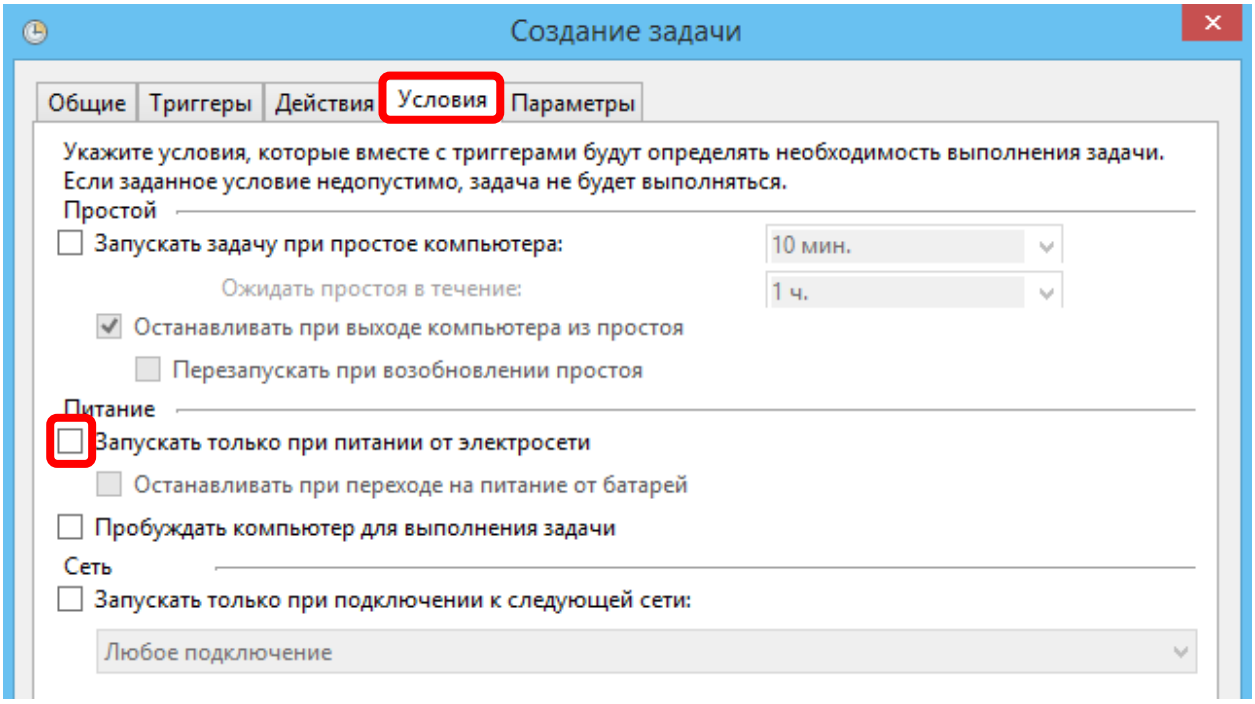


Рис. 131. Настройка дополнительных условий

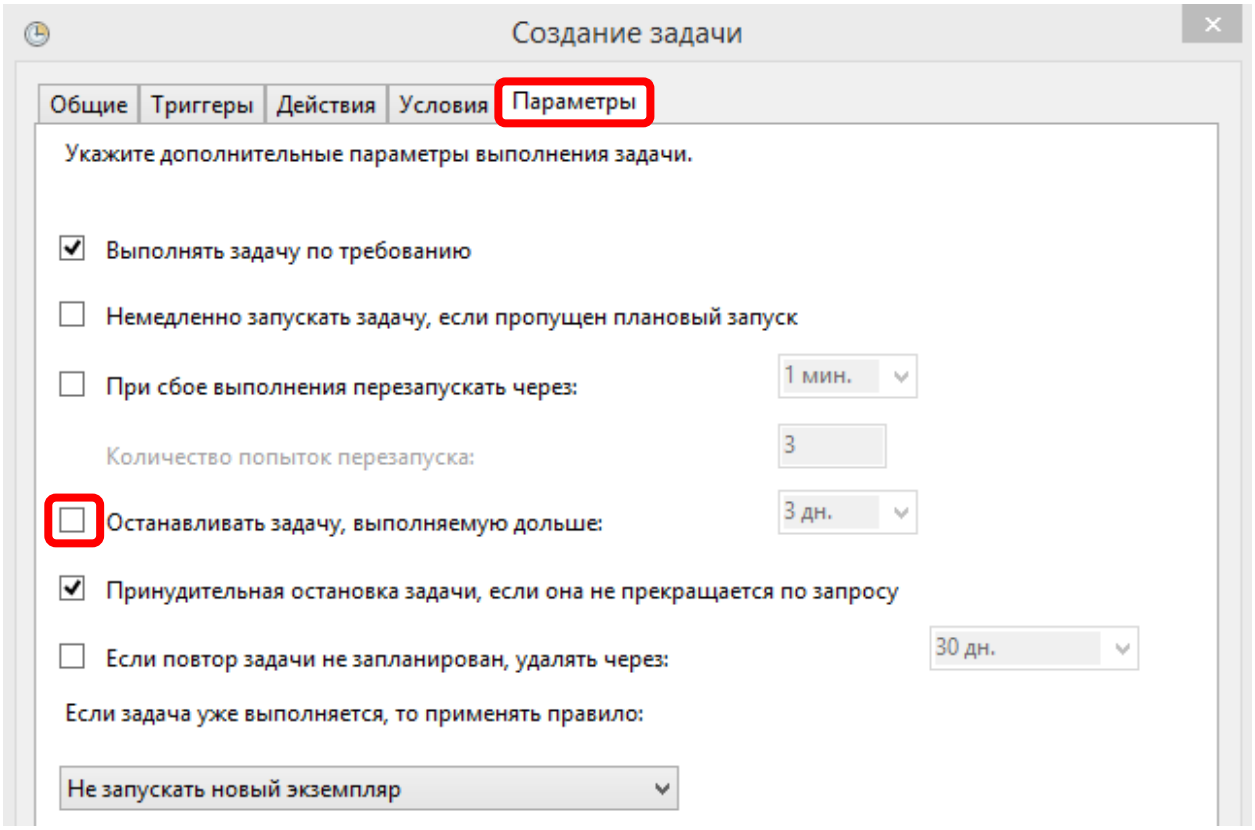


Рис. 132. Настройка дополнительных параметров

Рис. 133. Иконка WAMP-сервера у часов

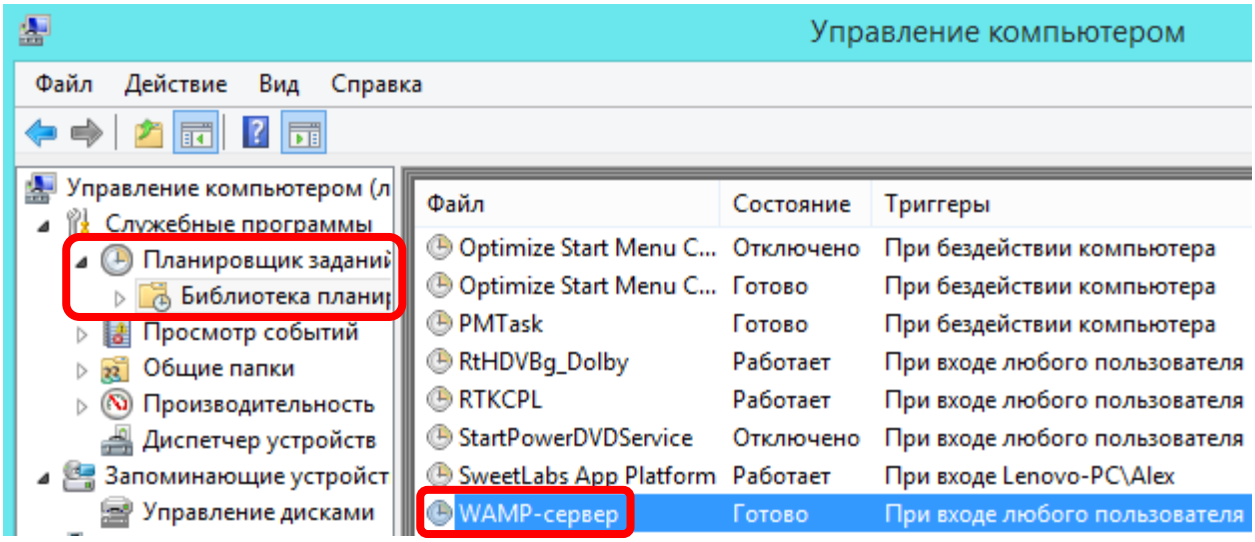


Рис. 134. Просмотр созданной задачи в Планировщике заданий

6.2. *Установка и настройка WAMP-сервера для компьютерного класса

Особенностью использования в компьютерном классе является то, что работа за компьютерами осуществляется из учетных записей без прав администратора. Но при этом для запуска WAMP-сервера права администратора необходимы.

Решением данной проблемы является настройка автоматического запуска WAMP-сервера с правами администратора, что можно осуществить при помощи Планировщика заданий Windows. При этом студент не будет иметь возможности изменять настройки WAMP-сервера (и даже не будет видеть его иконку у часов), но сможет осуществлять подключение к нему из клиентской программы (браузера), будет иметь доступ к базе данных MySQL через менеджер PhpMyAdmin.

1. Установка WAMP-сервера

Установка **WAMP**-сервера в основном производится с настройками по умолчанию. При этом стоит обратить внимание, что установка должна производиться в папку «**C:\wamp64**», т. е. только в корень диска (нельзя устанавливать данную программу в «C:\Program Files …»!). Путь не должен содержать пробелов и других специальных знаков, а также русских букв. Установка возможна только от имени **Администратора**.

!!! Нельзя устанавливать **WAMP**-сервер поверх имеющейся версии! Если ранее в данную папку уже был установлен **WAMP**-сервер, то необходимо вначале полностью удалить предыдущую версию при помощи стандартной функции Windows «Удалить или изменить программу», а при необходимости (если папка осталась) также удалить папку «C:\wamp64» вручную.

!!! При установке на **Windows 8** (и более ранние версии), необходимо выбрать для установки MySQL версии 5. При установке на **Windows 10** и более новые версии, необходимо выбрать для установки MySQL версии 8 (рис. 135).

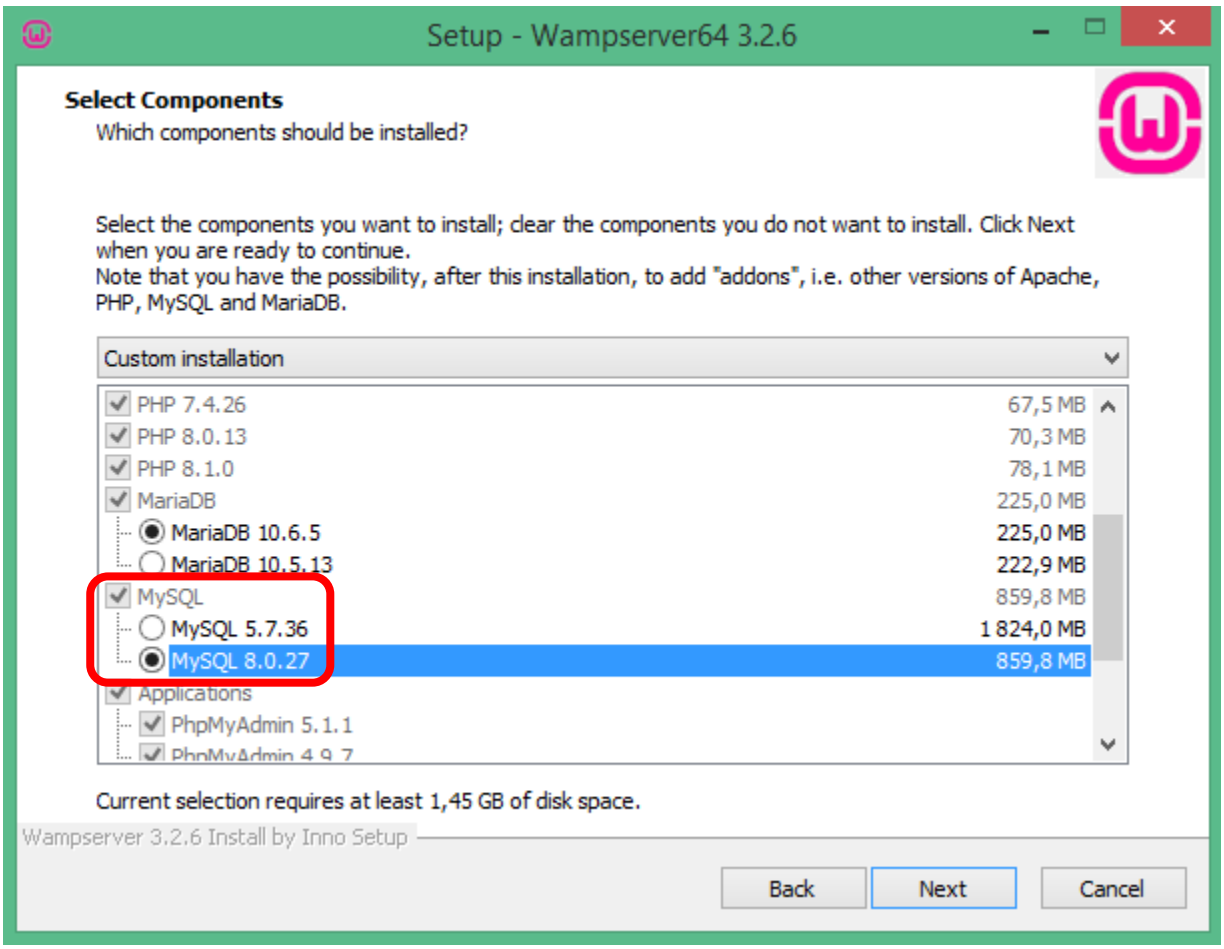


Рис. 135. Выбор версии MySQL для установки

2. Проверка работы MySQL-сервера

Для проверки правильности установки **WAMP**-сервера (и далее **MySQL**-сервера, входящего в его состав) необходимо проделать следующие действия:

- запустить появившийся на рабочем столе ярлык «**Wampserver64**»;

!!! Сервер стартует не мгновенно, после входа в учетную запись **необходимо подождать загрузки** сервера **не менее 1 минуты**. После запуска значок у часов должен стать зеленым (рис. 136).

Рис. 136. Значок WAMP-сервера у часов

- открыть браузер и ввести адрес **localhost**. Далее выбрать «**PhpMyAdmin**» (рис. 137);

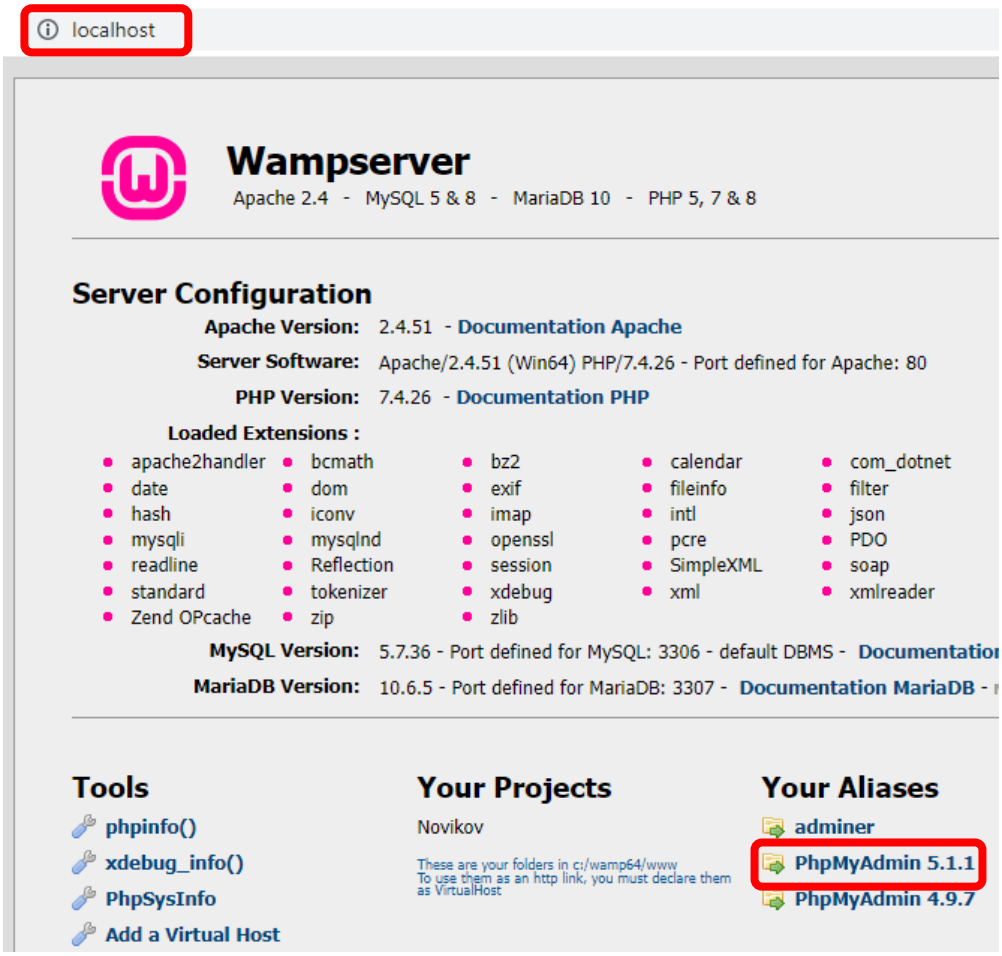


Рис. 137. Подключение к WAMP-серверу через браузер

- ввести имя пользователя **root** (без пароля) и подключиться к серверу **MySQL** (рис. 138).

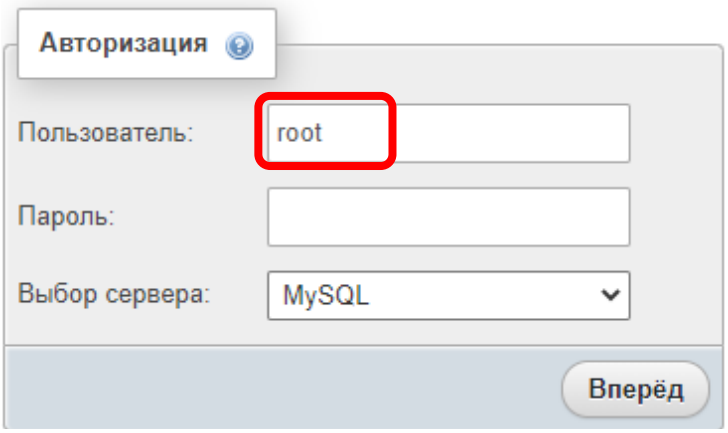


Рис. 138. Подключение к серверу MySQL

3. Настройка автоматического запуска WAMP-сервера

Если проверка работы **MySQL**-сервера прошла успешно, то далее необходимо настроить его автозапуск через Планировщик заданий.

Для запуска Планировщика заданий необходимо нажать правую клавишу мышки на значке «Мой компьютер» («Этот компьютер») и выбрать пункт меню «Управление» (рис. 139). *Другой способ попасть в Планировщик заданий – это запустить «Панель управления» и в ней выбрать «Система и безопасность» → «Администрирование» → «Расписание выполнения задач».*

Далее необходимо выбрать «Планировщик заданий» и нажать кнопку «Создать задачу» (рис. 140).

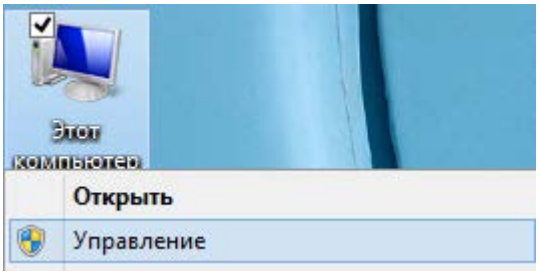


Рис. 139. Вызов Управления компьютером

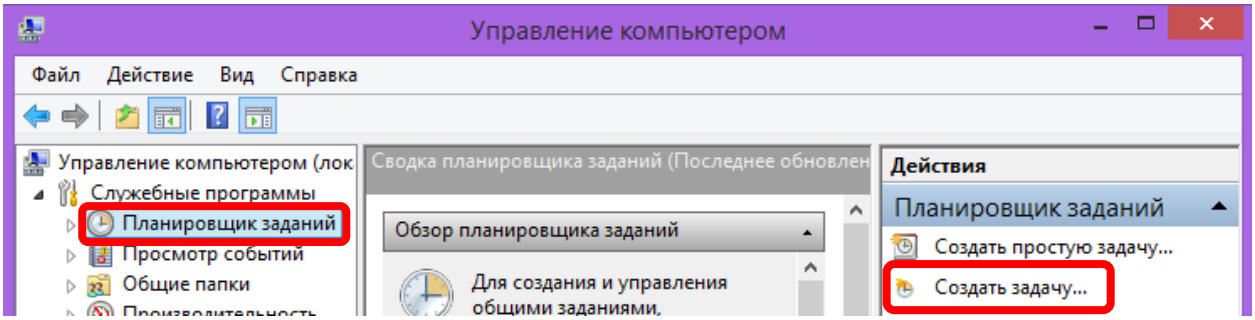


Рис. 140. Создание задачи в планировщике

В открывшемся окне произведем следующие настройки (рис. 141):

- указываем «Имя» создаваемой задачи;
- указываем описание создаваемой задачи (не обязательно);
- выбираем пункт «Выполнить для всех пользователей»;
- ставим галочку «Выполнить с наивысшими правами».

Далее переходим на вкладку «Триггеры» (рис. 142) и указываем условия запуска программы. Для этого необходимо нажать кнопку «Создать», и в появившемся окне (рис. 143) указать «При входе в систему» и «Любой пользователь».

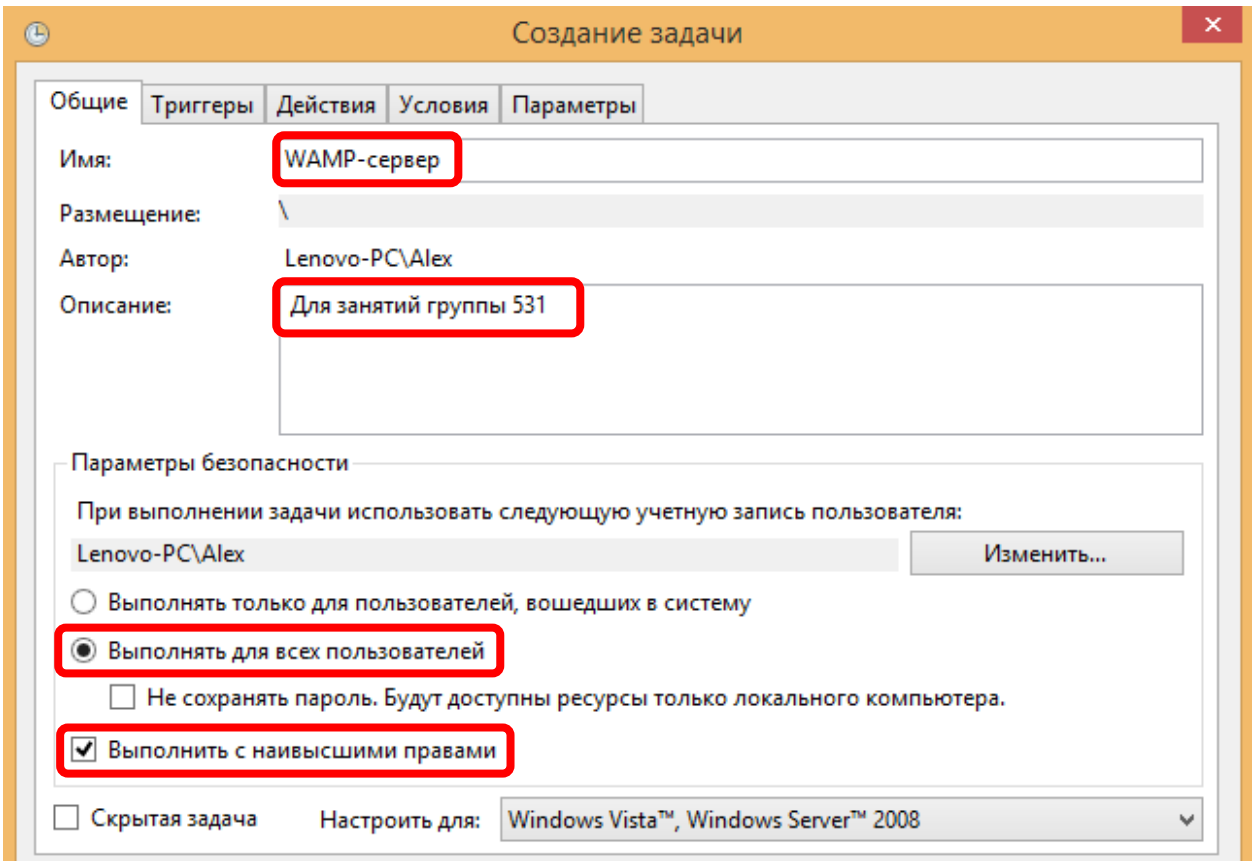


Рис. 141. Настройка создаваемой задачи

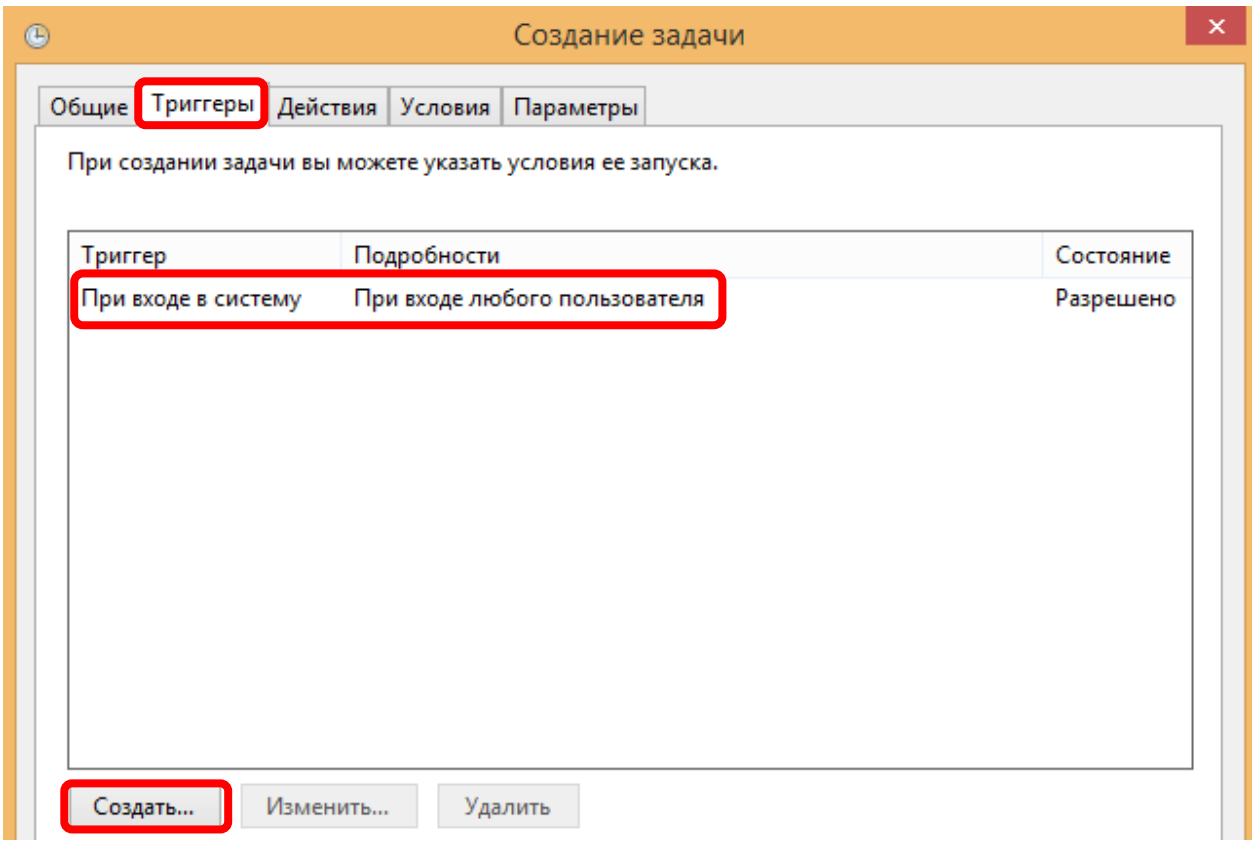


Рис. 142. Настройка условий запуска задачи

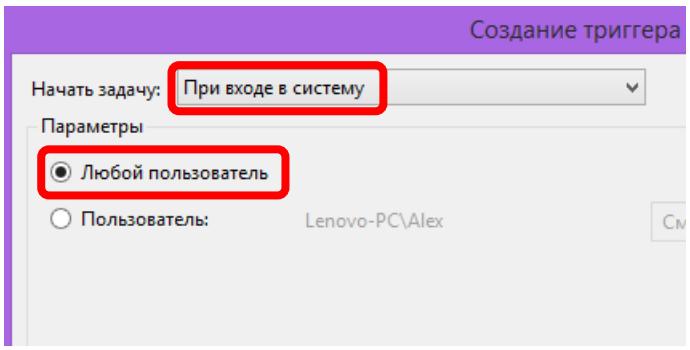


Рис. 143. Настройка запуска при входе в систему

На вкладке «Действия» (рис. 144) указываем какую программу требуется запустить. Для этого необходимо нажать кнопку «Создать», и в появившемся окне (рис. 145) указать «Запуск программы» и, нажав кнопку «Обзор», выбрать адрес до WAMP-менеджера (при установке WAMP-сервера в папку по умолчанию, это адрес «C:\wamp64\wampmanager.exe»).

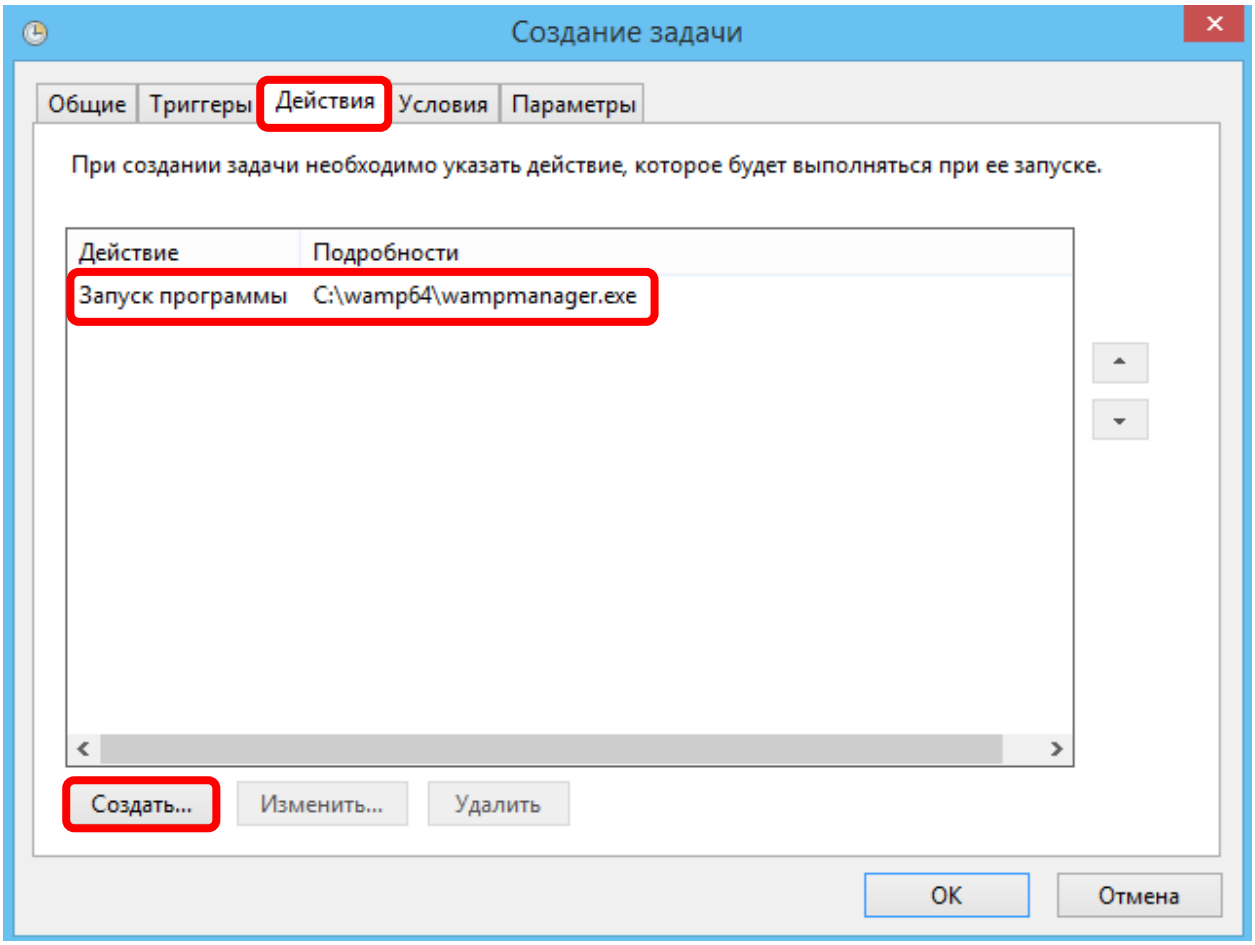


Рис. 144. Настройка действия, выполняемого задачей

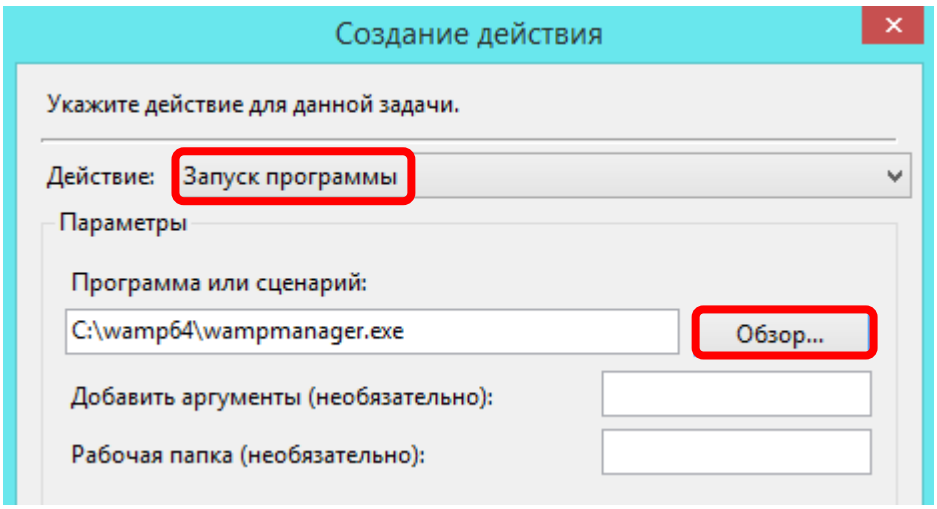


Рис. 145. Настройка запускаемой программы

На вкладках «Условия» (рис. 146) и «Параметры» (рис. 147) необходимо снять галочки, как показано на соответствующих рисунках.

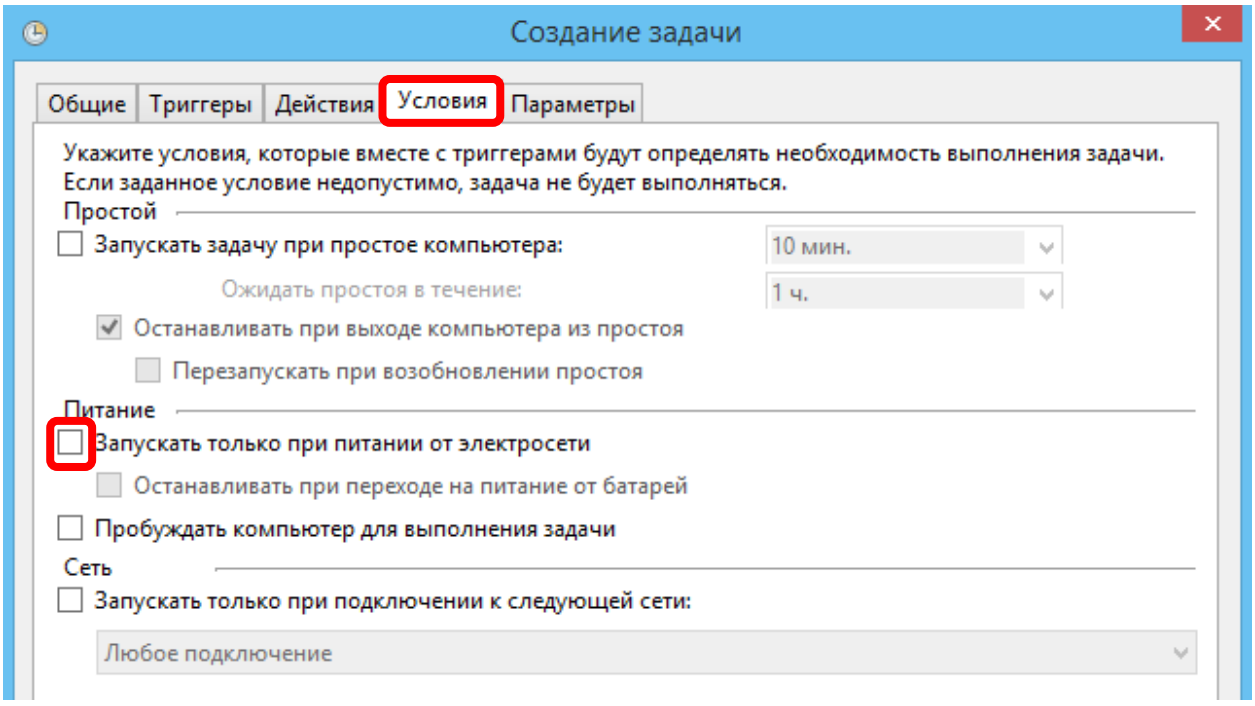


Рис. 146. Настройка дополнительных условий

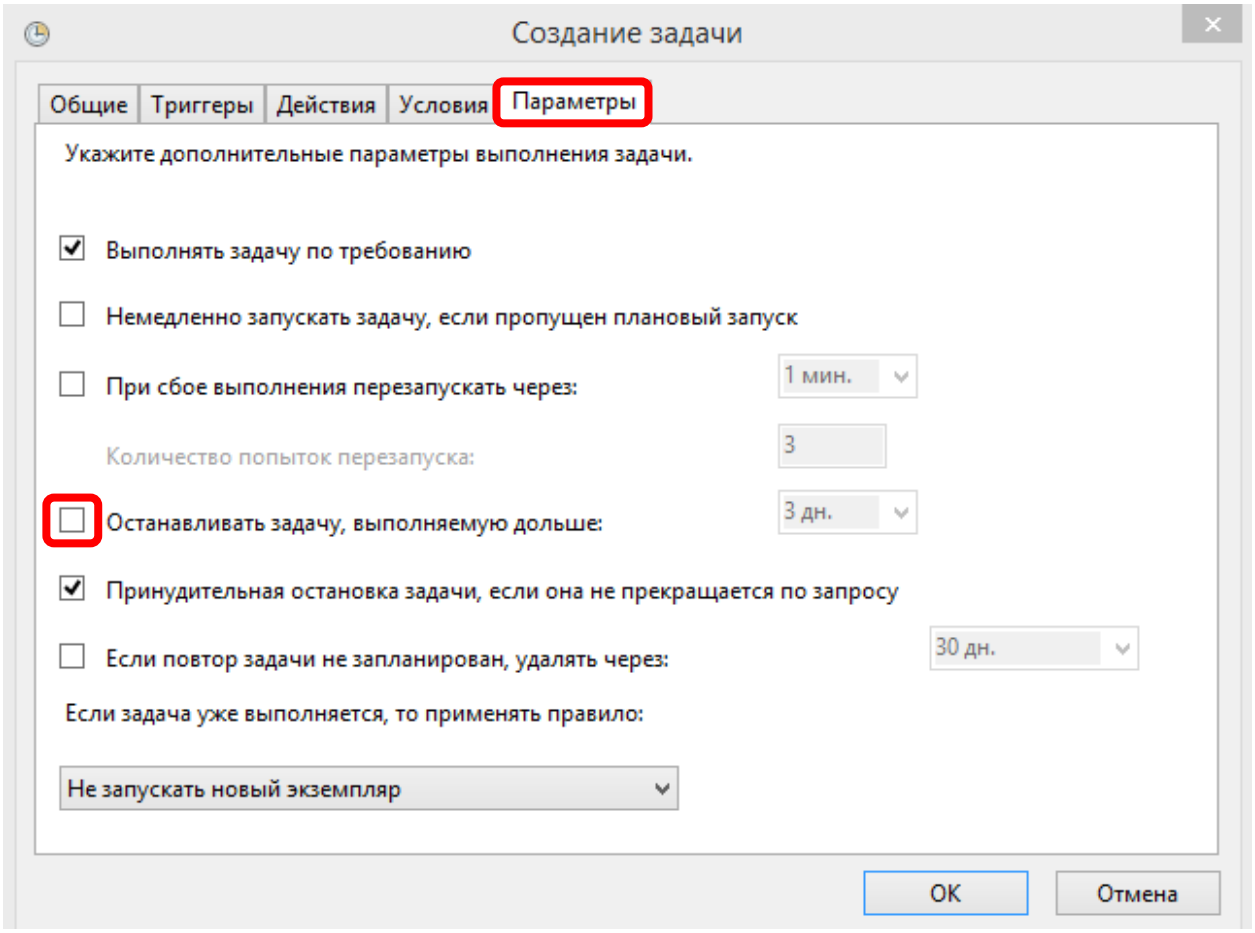


Рис. 147. Настройка дополнительных параметров

После нажатия кнопки «OK», появится окно (рис. 148), в которое нужно ввести пароль администратора. **На учетной записи администратора обязательно должен быть установлен пароль! (с пустым паролем работать не будет!!!).**

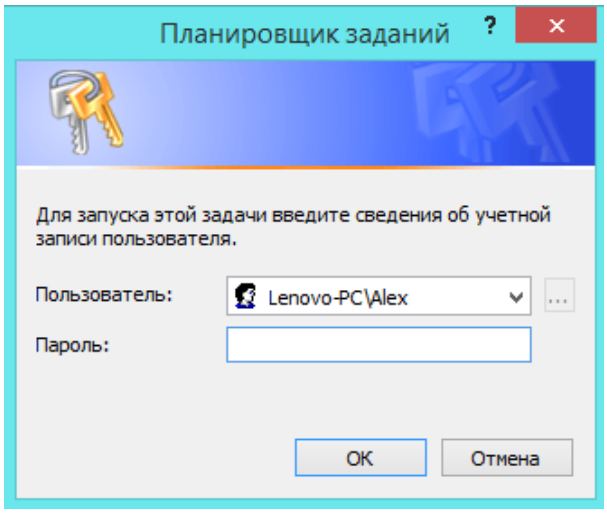


Рис. 148. Требование ввода пароля администратора

4. Проверка работы MySQL-сервера из учетной записи студента

После этого, нужно перезагрузить компьютер и, войдя из учетной записи студента, проверить работу WAMP-сервера *(как было описано ранее, за исключением того, что теперь не нужно запускать ярлык «**Wampserver64**» на рабочем столе)*.

Для проверки работы сервера необходимо открыть браузер и ввести адрес **localhost**. Далее выбрать «**PhpMyAdmin**» (рис. 137).

!!! Сервер стартует не мгновенно, после входа в учетную запись **необходимо подождать загрузки сервера не менее 1 минуты**.

Далее ввести имя пользователя **root** (без пароля) и подключиться к серверу **MySQL** (рис. 138).

5. Изменение и удаление созданной задачи

Для просмотра и изменения (редактирования, отключения, удаления) созданной задачи необходимо зайти (из учетной записи администратора) в Планировщик заданий и выбрать «Библиотека планировщика заданий» (рис. 149), после чего в списке (по алфавиту) найти созданную ранее задачу.

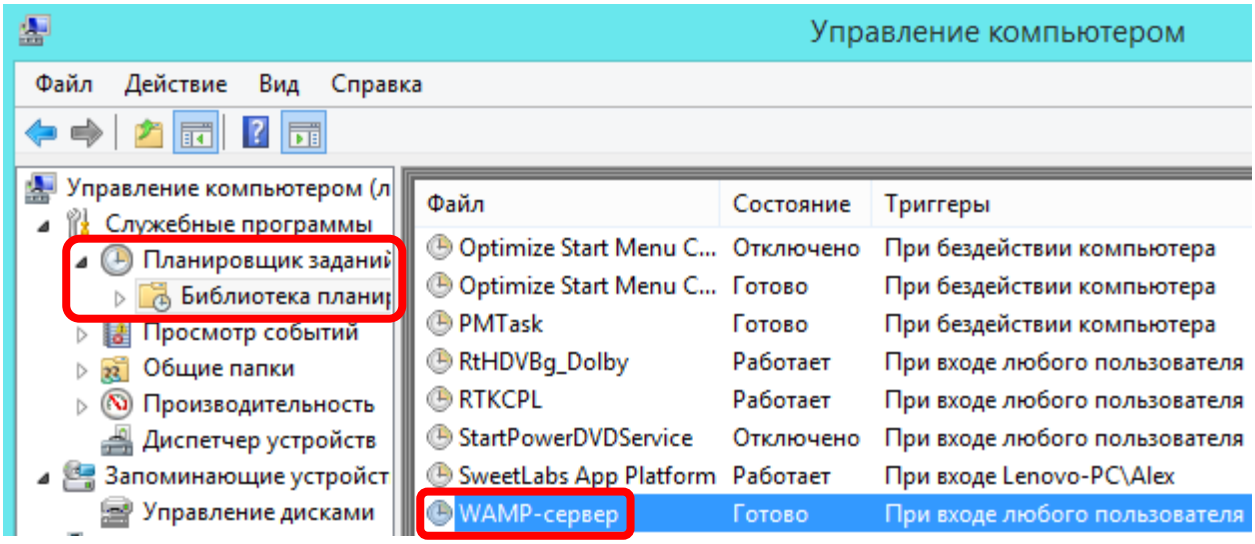


Рис. 149. Просмотр созданной задачи в Планировщике заданий

6.3. *Настройка прав доступа к MySQL

По умолчанию любой может получить доступ к **MySQL** через **PhpMyAdmin**, поэтому необходимо установить пароль на учетную запись «**root**». А также требуется разграничить права доступа для разных пользователей и разных баз данных.

1. Пароль учетной записи «root»

Для настройки пароля администратора (учетная запись «**root**») необходимо нажать надпись «Сервер: MySQL» вверху страницы (рис. 150).

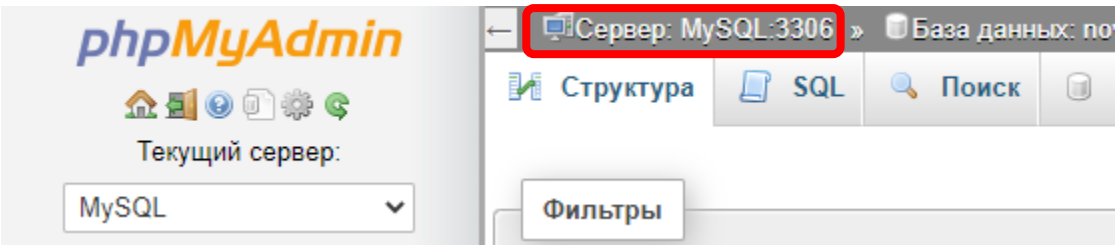


Рис. 150. Переход к настройкам сервера

Далее требуется выбрать вкладку «**Учетные записи пользователей**» (рис. 151) и кликнуть по названию учетной записи (в нашем случае «**root**»).

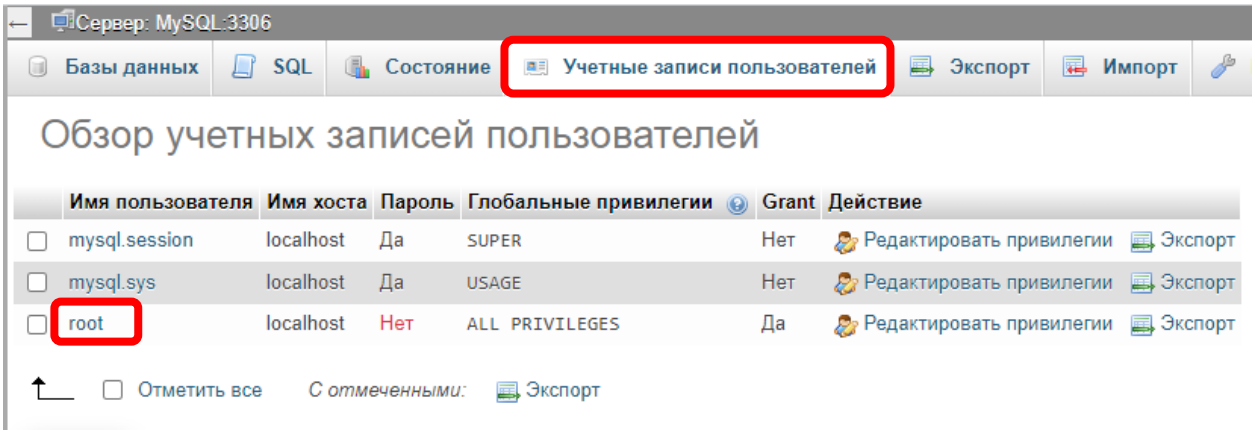


Рис. 151. Редактирование учетных записей пользователей

После этого необходимо перейти на вкладку «**Change password**» (рис. 152), где ввести новый пароль, а еще лучше нажать кнопку «**Генерировать**».

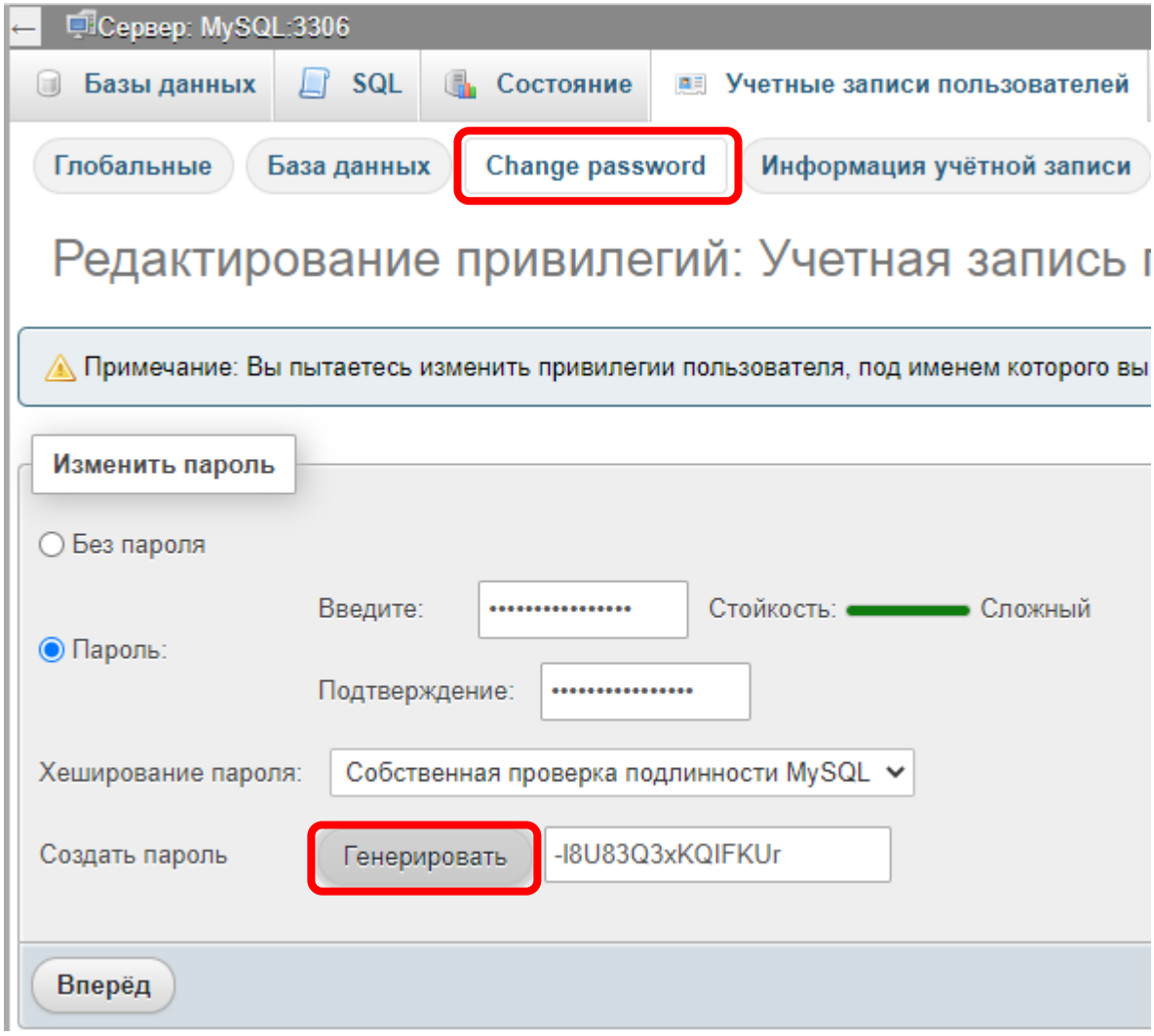


Рис. 152. Изменение пароля

2. Разграничение прав пользователей

При использовании **PHP**, все указанные в его коде пароли доступа к базе данных скрываются от конечного пользователя, запрашивающего страницу через браузер. Тем не менее, пароли все еще остаются доступны всем программистам, работающим над проектом, а также администраторам **HTTP**-сервера. Серьезными неприятностями будут удаление всех данных из базы или копирование базы данных конкурентами. Для предотвращения подобных ситуаций необходимо ограничить права учетных записей, пароли от которых используются в коде. Не рекомендуется использовать прямое подключение на стороне пользователя (без **PHP**-прослойки), например, через **JavaScript**, т. к. в этом случае пароль становится доступен всем пользователям. В таком случае необходимо разрешить доступ такой учетной записи только для выполнения запроса «**SELECT**».

В требуемой базе данных перейдем на вкладку «**Привилегии**» (рис. 153) и добавим новую учетную запись пользователя.

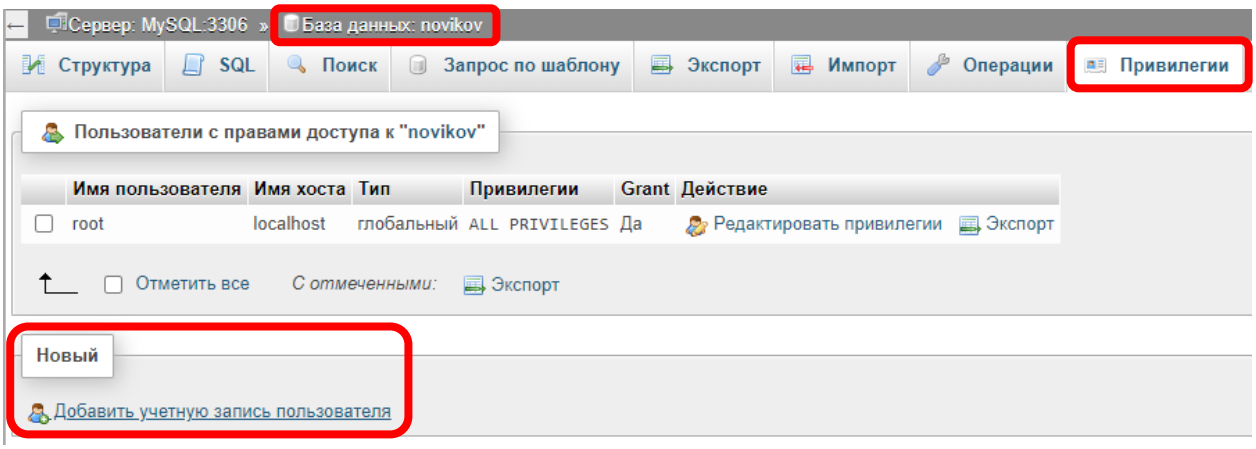


Рис. 153. Добавление нового пользователя

Нам необходимо создать две учетные записи «**view**» и «**edit**» для локального хоста (рис. 154).

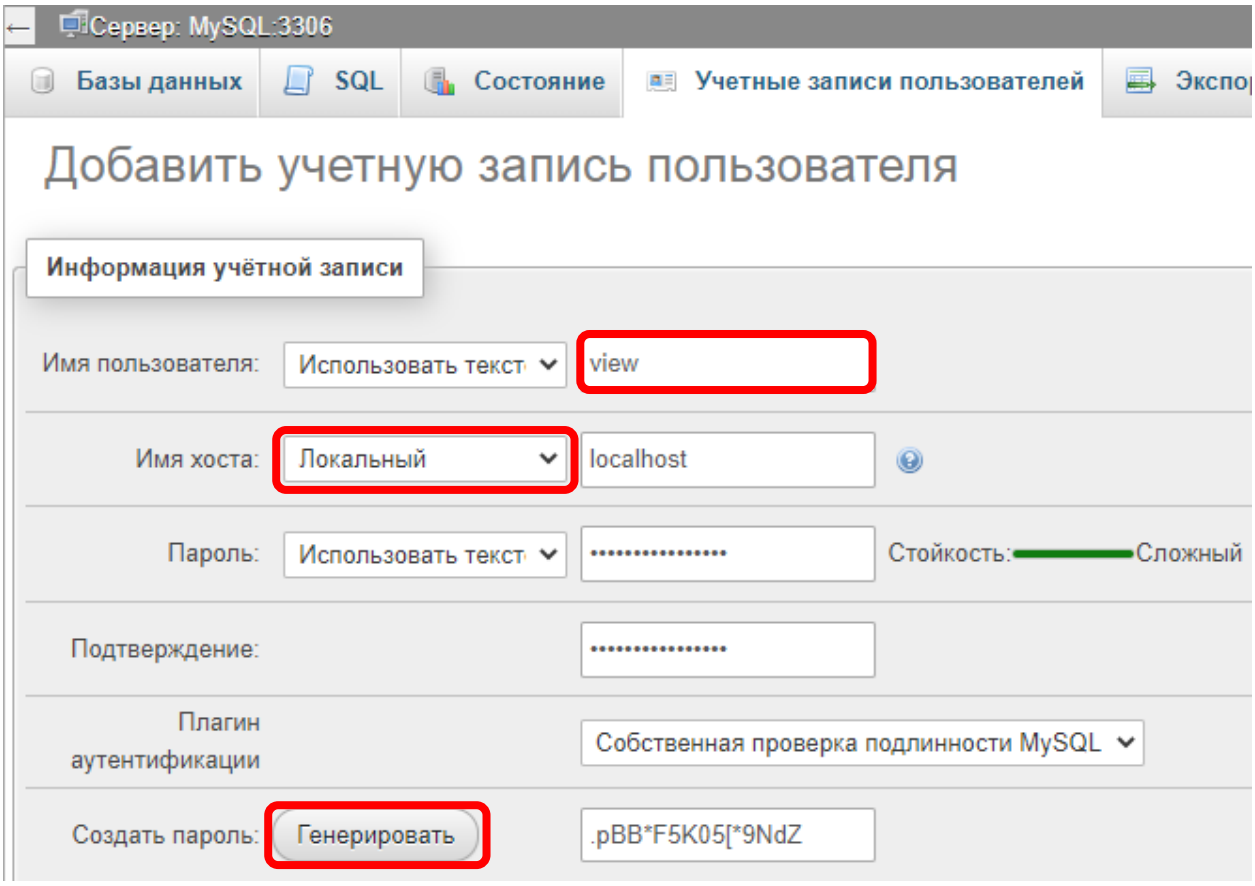


Рис. 154. Имя пользователя и пароль

Ниже необходимо указать права доступа. Для «**view**» это будет только «**SELECT**» (рис. 155), а для «**edit**» это будет «**SELECT**», «**INSERT**», «**UPDATE**» и, возможно, «**DELETE**».

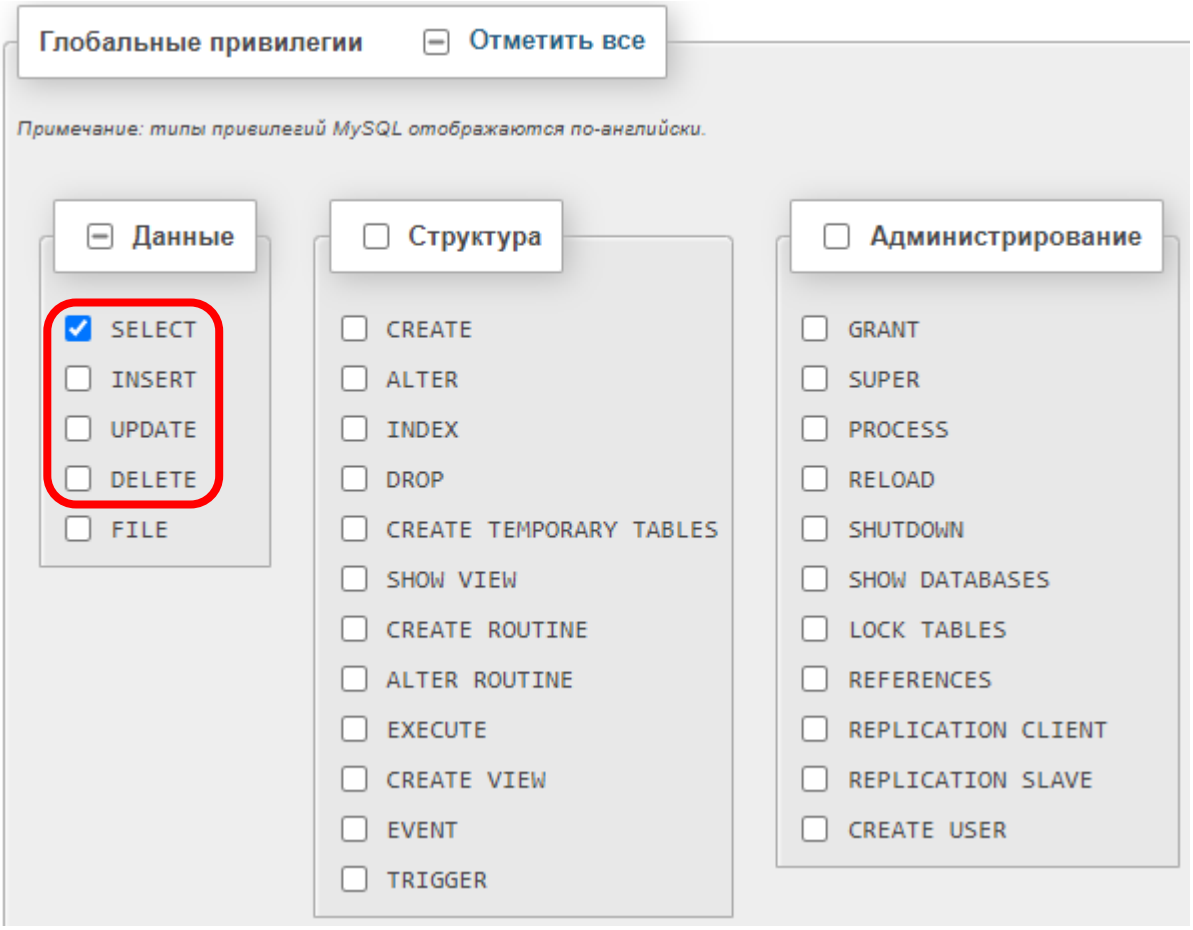
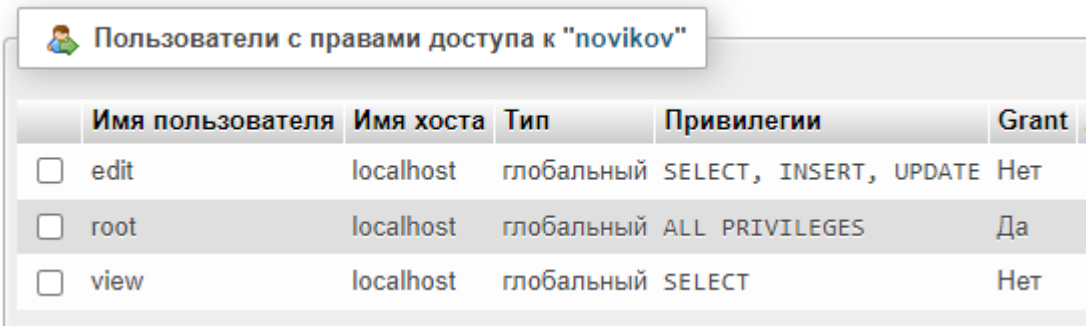


Рис. 155. Выбор прав доступа

В итоге у нас получится **3** учетные записи (рис. 156), причем создавать и удалять таблицы можно будет только из учетной записи «**root**», а редактировать содержимое таблиц можно будет также и пользователю «**edit**».



	Имя пользователя	Имя хоста	Тип	Привилегии	Grant
☐	edit	localhost	глобальный	SELECT, INSERT, UPDATE	Нет
☐	root	localhost	глобальный	ALL PRIVILEGES	Да
☐	view	localhost	глобальный	SELECT	Нет

Рис. 156. Добавленные учетные записи пользователей

6.4. *Оптимизация PHP-кода для MySQL

После того, как мы сменили Логин и/или Пароль доступа к базе данных, необходимо изменить их и во всех **PHP**-файлах в строке:

```
//Подключиться к базе данных
$connect = mysqli_connect("localhost", "root", "", "Novikov");
```

Если таких **PHP**-файлов много, то придется вручную в каждом из них менять Логин и Пароль, что может быть затруднительно, а также этот ручной труд может приводить к ошибкам.

Переделаем код **PHP** таким образом, чтобы Логин и Пароль можно было вводить один раз для всех **PHP**-файлов. В папке «C:\wamp64\www\Novikov**Admin**» создадим текстовый файл «**SQL_Connect.php**». В данный файл скопируем (из ранее созданных **PHP**-файлов) следующий код:

```
<?php
  //Подключиться к базе данных
  $connect = mysqli_connect("localhost", "root", "", "Novikov");
  //Завершить работу в случае ошибки
  if ($connect == false) {
      print("Невозможно подключиться к MySQL");
      exit;
  }
  //Установить кодировку
  mysqli_set_charset($connect, 'utf8');
?>
```

В этом коде также нужно указать новые Логин и Пароль.

Далее во всех **PHP**-файлах, в которых содержался данный код подключения к **SQL**, необходимо заменить этот код единственной командой:

```
//Подключиться к базе данных
include "Admin/SQL_Connect.php";
```

Теперь, когда необходимо будет изменить Логин и Пароль для подключения к базе данных, достаточно будет сменить их в единственном файле «**SQL_Connect.php**».

6.5. *Использование MariaDB вместо MySQL

Ранее мы использовали **MySQL** в качестве базы данных по умолчанию. Но в этой роли также может выступать и **MariaDB**.

В изначальной конфигурации, **MySQL** подключается к порту **3306**, а **MariaDB** к порту **3307**. Проверить номера портов можно на главной странице **WAMP**-сервера (рис. 157), зайдя по адресу «**localhost**».

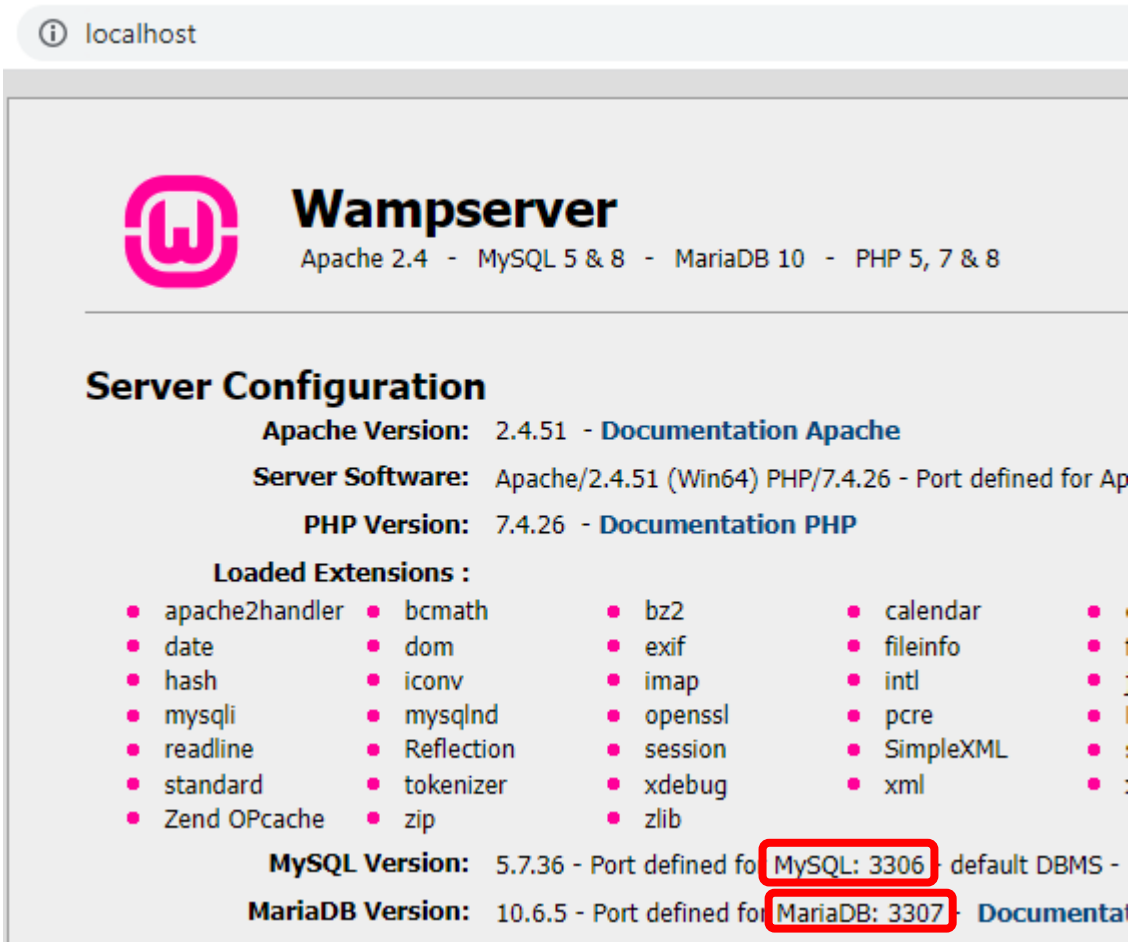


Рис. 157. Информация о портах MySQL и MariaDB

Для изменения конфигурации необходимо нажать правую клавишу мышки на значке **WAMP**-сервера (у часов) и в меню «*Tools*» выбрать пункт «**Invert default DBMS**» (рис. 158).

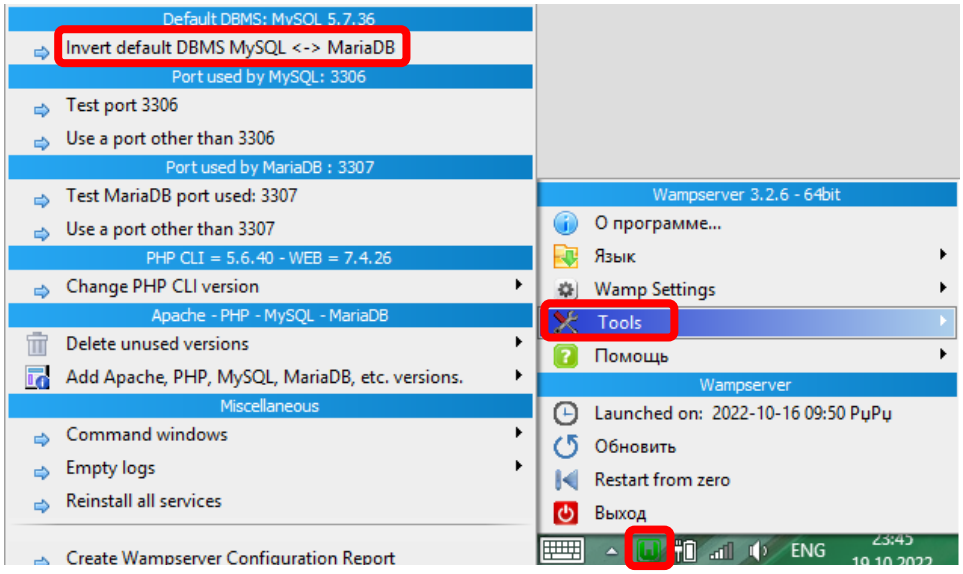


Рис. 158. Изменение БД по умолчанию

После этого порты изменятся (рис. 159) на **3306** – для **MariaDB**, и **3308** – для **MySQL**.

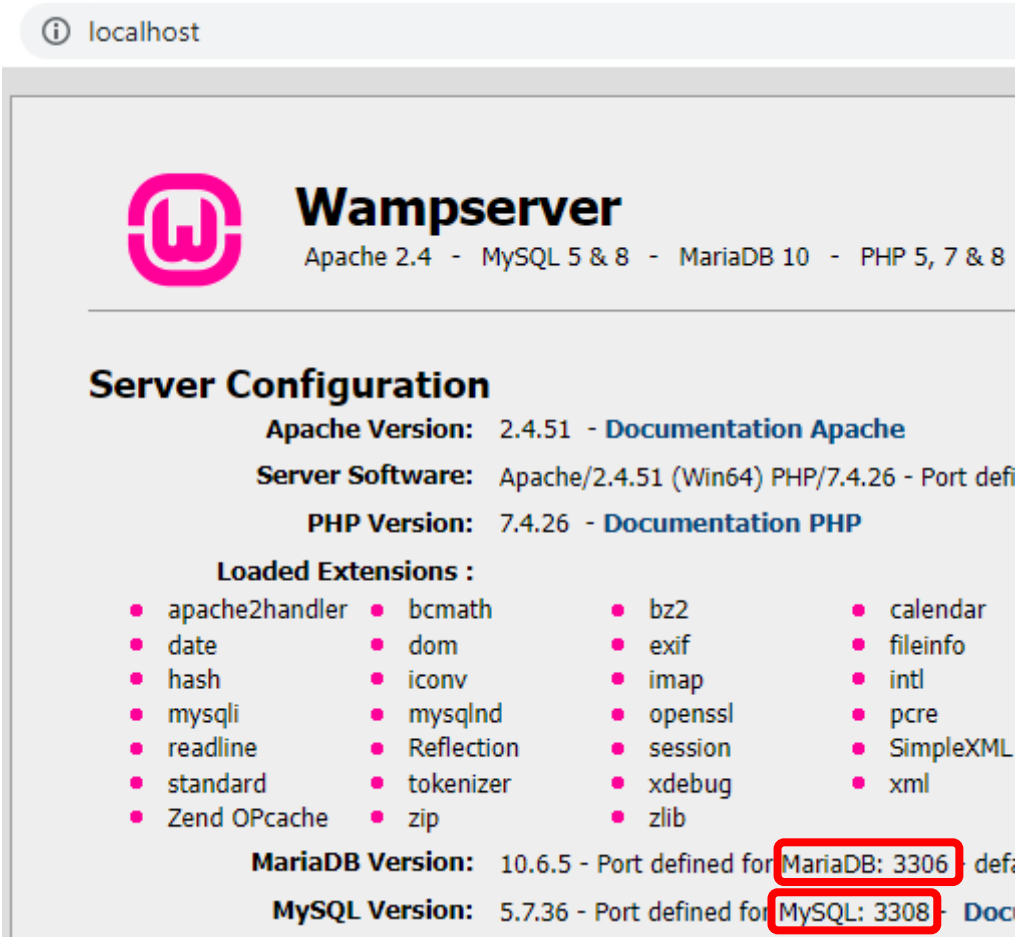


Рис. 159. Измененные порты MySQL и MariaDB

Кроме того, можно было вообще не «инвертировать» **СУБД**, а просто подключаться к необходимой по известному порту. Ранее, для подключения к БД из **PHP**, мы использовали команду:

```
//Подключение к СУБД по умолчанию
mysqli_connect("localhost", "root", "", "Novikov");
```

У этой-же команды есть и расширенный вариант, с указанием номера порта:

```
//Подключение к СУБД по умолчанию
mysqli_connect("localhost", "root", "", "Novikov", "3306");
//Подключение к MariaDB
mysqli_connect("localhost", "root", "", "Novikov", "3307");
//Подключение к MySQL
mysqli_connect("localhost", "root", "", "Novikov", "3308");
```

!!! Стоит обратить внимание, что одна и та-же **СУБД** не может занимать одновременно два порта, т.е. если она выбрана как «СУБД по умолчанию» (на порту **3306**), то на портах **3307** или **3308** она будет недоступна**!**

6.6. *Доступ к серверу с других устройств

Ранее мы подключались из браузера к **Apache**-серверу (**WAMP**-серверу), расположенному на том же ПК. При этом использовался адрес «**localhost**» или «**127.0.0.1**». Далее будет рассказано, как подключиться к серверу с других устройств, расположенных в одной Локальной сети с сервером. В наиболее классическом случае речь идет о том, что ПК с установленным **WAMP**-сервером и второй ПК (или смартфон) подключены к одному роутеру (не важно, по Wi-Fi или кабелем).

Для подключения к серверу с другого устройства требуется:

1. Узнать текущий **IP**-адрес сервера (в примере это будет «**192.168.0.120**»).
2. Настроить **Брандмауэр** Windows (и, возможно, Firewall и Антивирус) на сервере.
3. Настроить **Apache**-сервер.
4. Также рекомендуется настроить статический локальный **IP**-адрес для сервера.

1. IP-адрес сервера

Для того чтобы узнать текущий локальный **IP**-адрес (**IPv4**-адрес) компьютера, существует несколько способов. Рассмотрим основные из них.

1. Нажать комбинацию клавиш «**Windows+R**», ввести «**cmd**» (рис. 160), в появившемся окне ввести команду «**ipconfig**» (рис. 161).

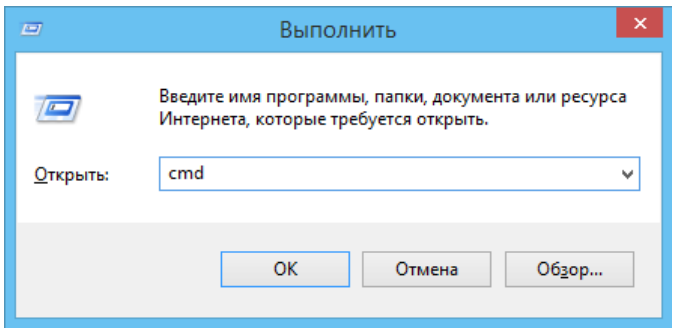


Рис. 160. Вызов «cmd»

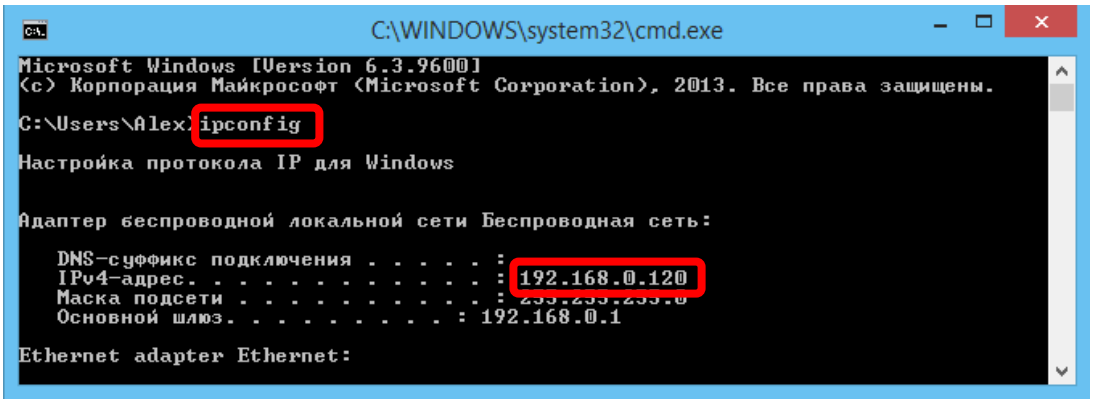


Рис. 161. Результаты команды «ipconfig»

2. Нажать комбинацию клавиш «**Windows+X**» и выбрать пункт «**Сетевые подключения**» (рис. 162). Дважды кликнуть мышкой по требуемому сетевому подключению и в открывшемся окне нажать кнопку «**Сведения**». В появившемся окне будет отображен текущий **IP**-адрес (рис. 163).

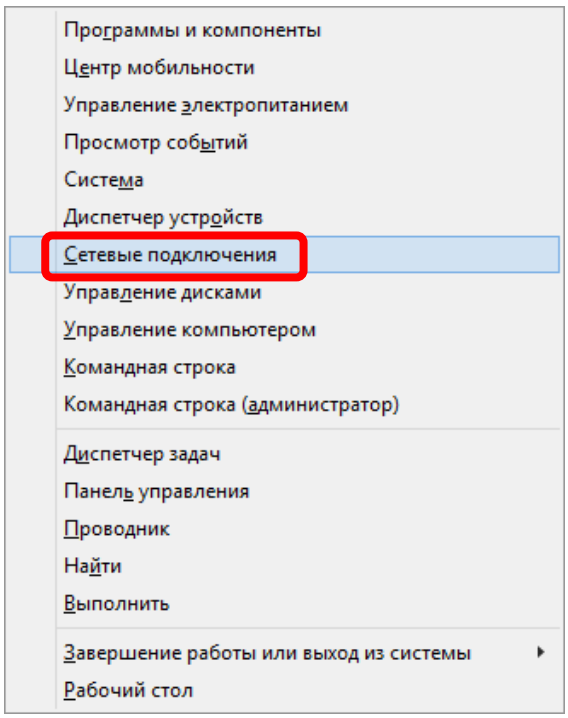


Рис. 162. Меню «Windows+X»

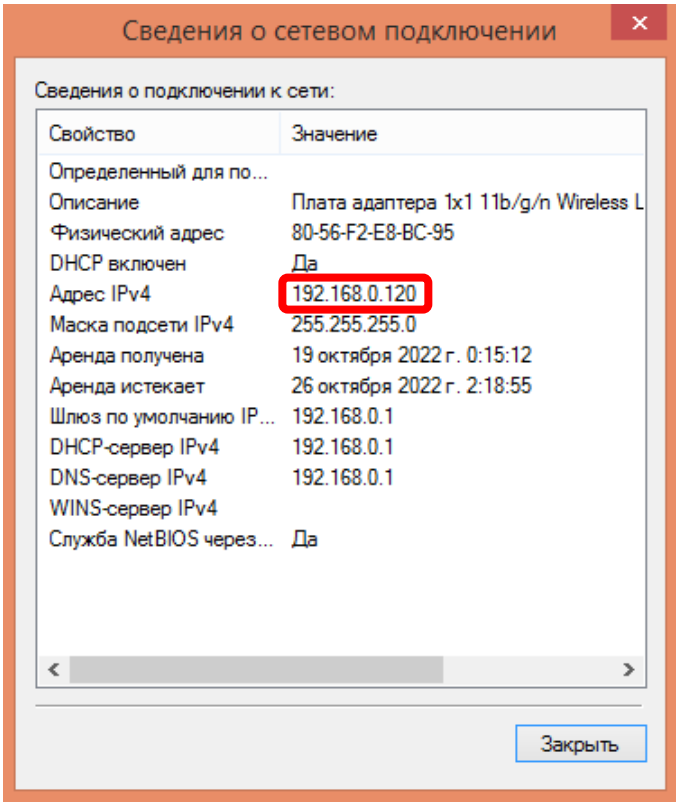


Рис. 163. Сведения о сетевом подключении

3. В углу у часов нажать правой кнопкой мышки на значок сети и выбрать «**Центр управления сетями и общим доступом**» (рис. 164). После чего выбрать «**Изменение параметров адаптера**» (рис. 165) и далее, как и в предыдущем способе, дважды кликнуть мышкой по требуемому

сетевому подключению и в открывшемся окне нажать кнопку «**Сведения**». В появившемся окне будет отображен текущий IP-адрес (рис. 163).

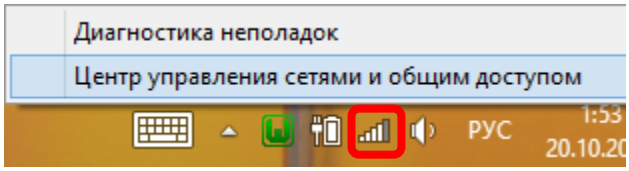


Рис. 164. Вызов Центра управления сетями

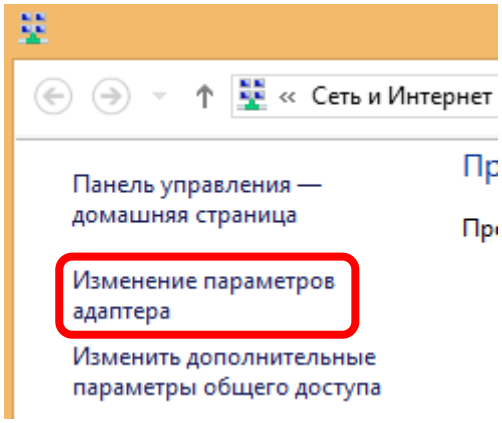


Рис. 165. Изменение параметров адаптера

Зайдем по найденному адресу через браузер (пока с того же компьютера, для которого мы определили этот адрес) и убедимся, что он работает (рис. 166). Также обратим внимание на то, что теперь браузер предупреждает нас, что подключение «Не защищено».

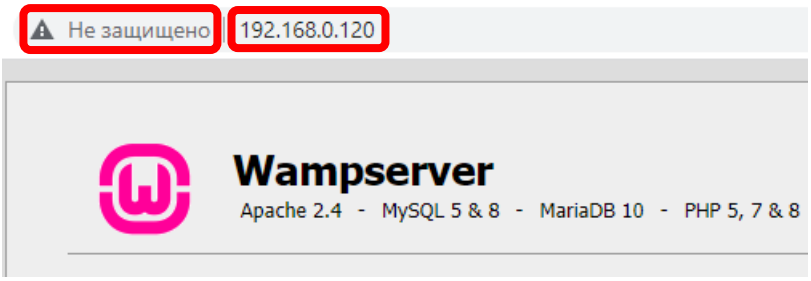


Рис. 166. Доступ по адресу 192.168.0.120

2. Настройка Брандмауэра

Попробуем зайти по этому же адресу (192.168.0.120) с любого другого устройства в этой Локальной сети, например со смартфона, подключенного по

Wi-Fi к тому же роутеру. Практически наверняка вначале мы получим ошибку «Не удается получить доступ к сайту» (рис. 167).

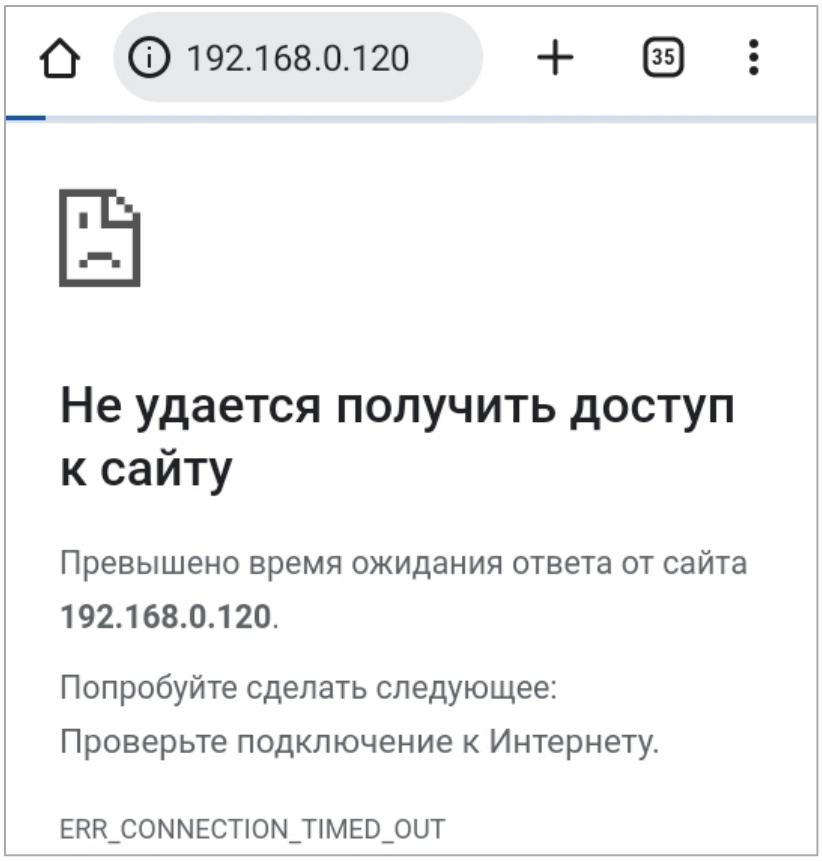


Рис. 167. Неудачное подключение со смартфона

Это означает, что соединение блокирует Брандмауэр Windows (и, возможно, Firewall и Антивирус). Для отключения Брандмауэра Windows необходимо перейти в «**Панель управления**\Система и безопасность**Брандмауэр Защитника Windows**» (рис. 168).

!!! Стоит обратить внимание, что отключение Брандмауэра (Firewall, Антивируса) здесь осуществляется лишь для демонстрации возможности подключения к серверу. Не следует применять отключение Брандмауэра (Firewall, Антивируса) на постоянной основе, в дальнейшем необходимо настроить его таким образом, чтобы он разрешал работу Apache-сервера, но не отключал другие запреты.

Если все запреты Брандмауэра (Firewall, Антивируса) отключены правильно, то теперь заходя по адресу 192.168.0.120, мы получим сообщение об ошибке «**403 Forbidden**» – «Доступ запрещен» (рис. 169).

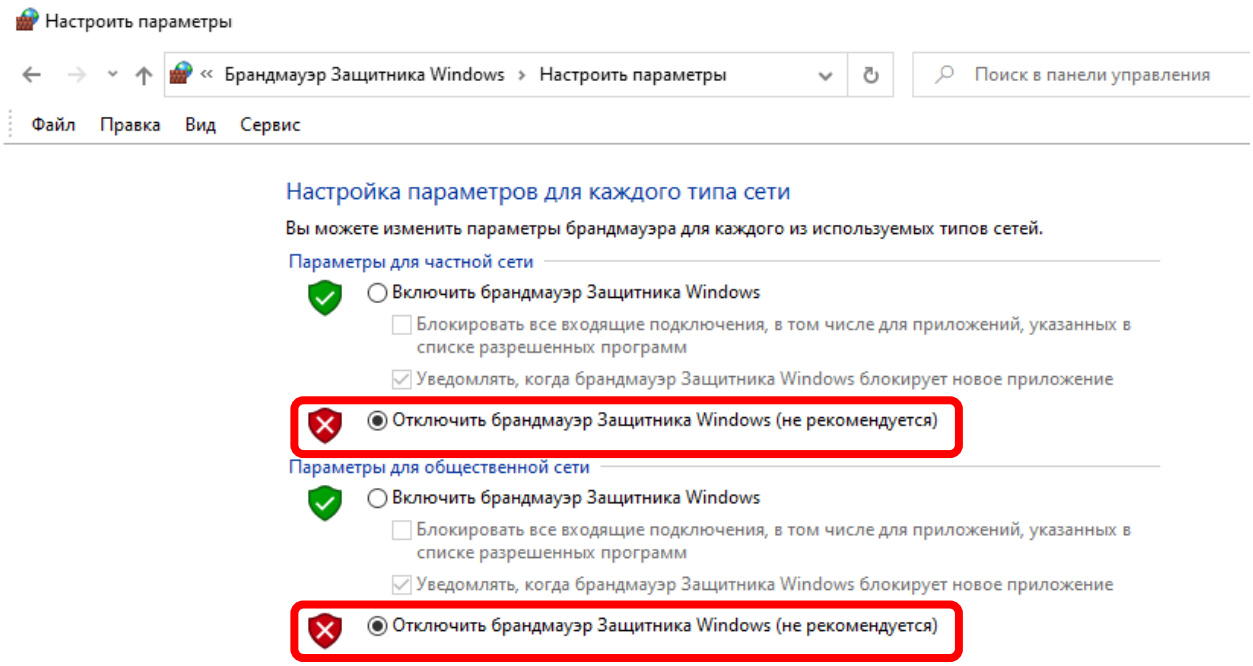


Рис. 168. Отключение Брандмауэра Windows

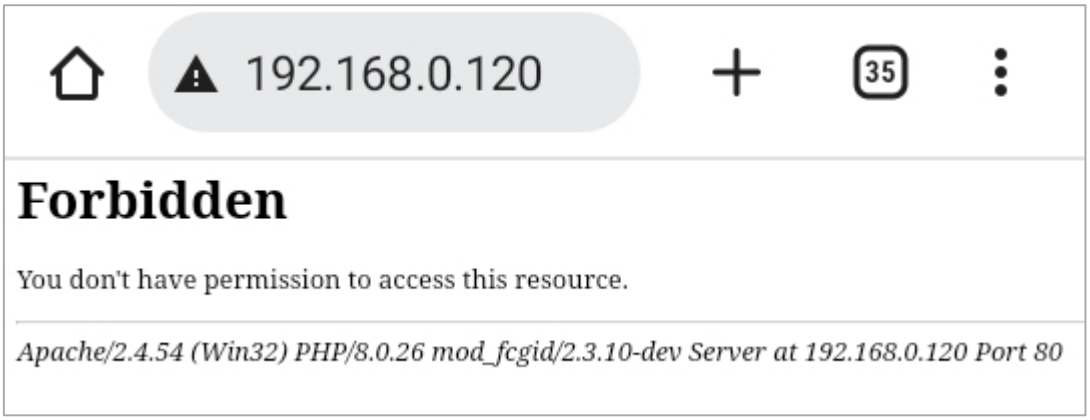


Рис. 169. Запрет доступа Apache-сервером

Это сообщение об ошибке «**403 Forbidden**» («Доступ запрещен») означает, что связь с Apache-сервером есть, но дальнейшая загрузка страницы блокируется самим сервером.

3. Настройка Apache-сервера

Когда мы наконец-то получили ответ от **Apache**-сервера, пусть и в виде ошибки «**403 Forbidden**» («Доступ запрещен»), пришло время настроить и его.

Откроем с помощью блокнота файл конфигурации, расположенный по адресу «C:\wamp64\bin\apache\apache2.4.51\conf\extra**httpd-vhosts.conf**», и исправим в нем соответствующую строчку на «**Require all granted**» (рис. 170). После внесения изменений может потребоваться перезагрузить **WAMP**-сервер.

*Чтобы обратно запретить доступ по сети, необходимо исправить значение на «**Require local**».*

```
# Virtual Hosts
#
<VirtualHost *:80>
  ServerName localhost
  ServerAlias localhost
  DocumentRoot "${INSTALL_DIR}/www"
  <Directory "${INSTALL_DIR}/www/">
    Options +Indexes +Includes +FollowSymLinks +MultiViews
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

Рис. 170. Разрешение доступа к Apache-серверу

Теперь заходя по адресу 192.168.0.120 мы получим доступ к требуемой странице (рис. 171).

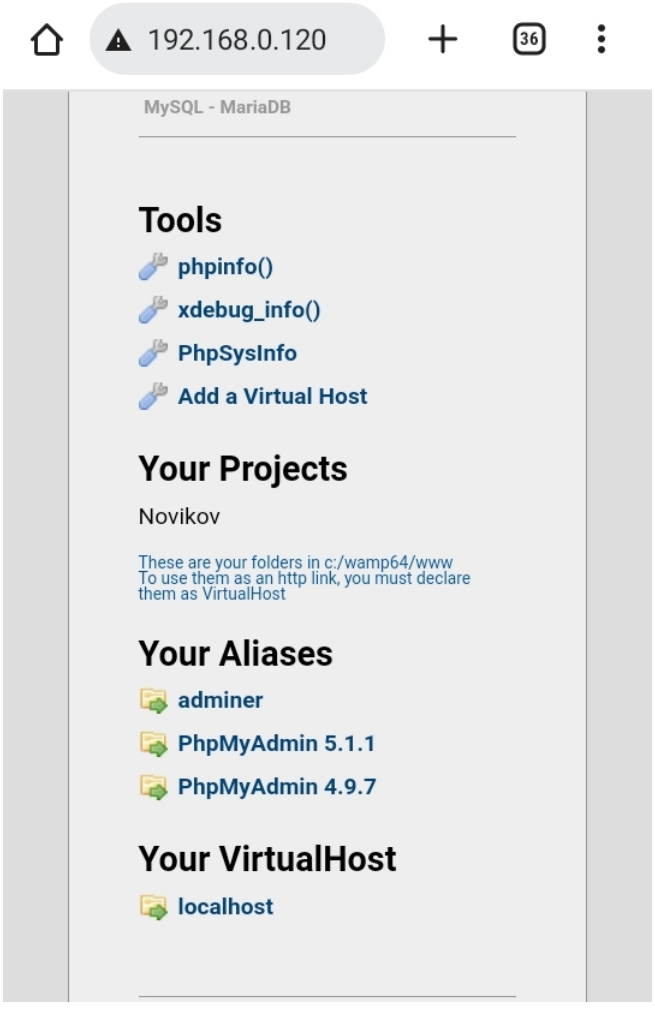


Рис. 171. Удачное подключение со смартфона

Если мы хотим иметь доступ и к **PhpMyAdmin**, то необходимо аналогично в файле «C:\wamp64\alias**phpmyadmin.conf**» исправить строчку на «**Require all granted**» (рис. 172).

```
1  Alias /phpmyadmin "c:/wamp64/apps/phpmyadmin5.1.1/"
2
3  <Directory "c:/wamp64/apps/phpmyadmin5.1.1/">
4    Options +Indexes +FollowSymLinks +MultiViews
5    AllowOverride all
6    Require all granted
7
8  # To import big file you can increase values
9    php_admin_value upload_max_filesize 128M
10   php_admin_value post_max_size 128M
11   php_admin_value max_execution_time 360
12   php_admin_value max_input_time 360
13 </Directory>
```

Рис. 172. Разрешение доступа к PhpMyAdmin

Стоит обратить внимание, что мы настроили доступ только к одной из версий **PhpMyAdmin** (самой новой из установленных). Для настройки других версий **PhpMyAdmin** (если их установлено несколько) необходимо внести аналогичные изменения в файлы настройки, у которых в имени файла указана версия, например, «C:\wamp64\alias**phpmyadmin4.9.7.conf**».

4. Настройка статического локального IP-адреса

Если ПК с установленным **WAMP**-сервером имеет динамический **IP**-адрес, т. е. адрес, который присваивается роутером, то такой адрес может периодически менять свое значение, что затруднит подключение к такому серверу с других устройств.

Для сервера имеет смысл применять статический **IP**-адрес. Для его настройки необходимо в свойствах сетевого соединения (рис. 173) прописать IP версии 4 (рис. 174).

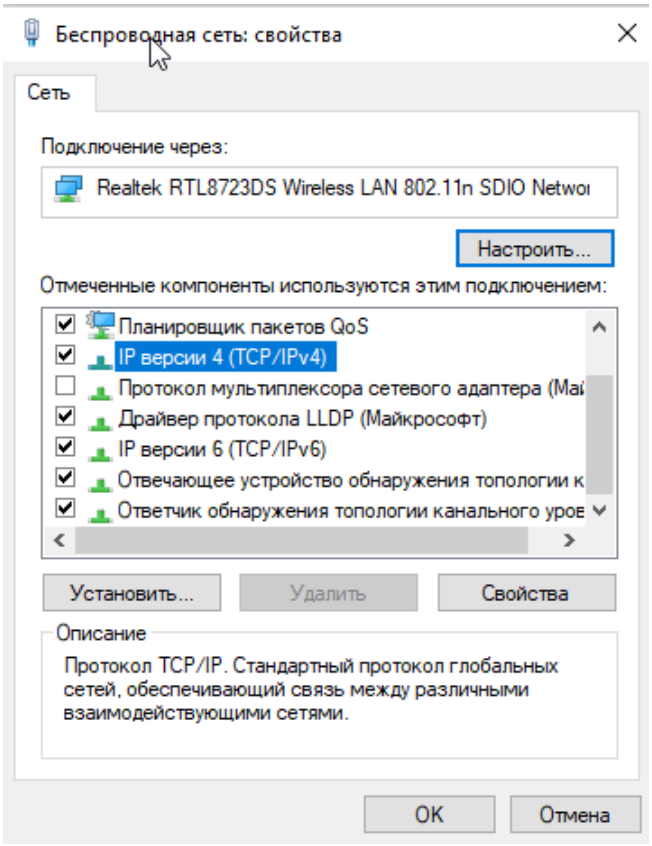


Рис. 173. Настройка IP-адреса

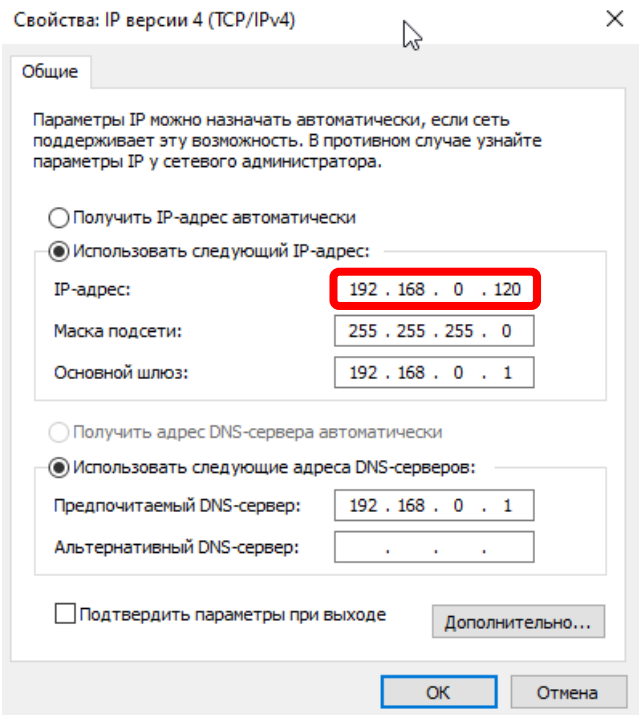


Рис. 174. Статический IP-адрес

6.7. *Шифрование и SSL-сертификат

SSL (Secure Sockets Layer – уровень защищённых сокетов) – криптографический протокол, который подразумевает более безопасную связь[16]. Использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. **SSL** считается устаревшим и должен быть исключён из работы в пользу **TLS**.

TLS (англ. Transport Layer Security – протокол защиты транспортного уровня)[17] – фактически, это новая версия **SSL**. В настоящее время под названием «SSL» обычно подразумевается как сам SSL, так и TLS.

SSL изначально разработан компанией Netscape Communications для добавления протокола **HTTPS** в свой веб-браузер Netscape Navigator. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший имя **TLS**.

HTTPS (HyperText Transfer Protocol Secure) – расширение протокола **HTTP** для поддержки шифрования в целях повышения безопасности[18]. Данные в протоколе HTTPS передаются поверх криптографического протокола **TLS** (или устаревшего в 2015 году SSL). В отличие от HTTP с TCP-портом 80, для HTTPS по умолчанию используется TCP-порт **443**. Протокол разработан компанией Netscape Communications для браузера Netscape Navigator в 1994 году.

OpenSSL – программа для генерации **SSL**-сертификатов.

Для включения функции шифрования необходимо выполнить следующие два действия:

1. Сгенерировать сертификат (пару файлов сертификат и ключ):
   - для публичного сервера, у которого имеется домен в интернете, для получения **SSL**-сертификата необходимо обратиться в один из центров сертификации (это платная услуга);
   - для частного сервера генерируется «самоподписанный сертификат». В составе **WAMP**-сервера для этого уже имеется программа **OpenSSL**, она расположена по адресу: «C:\wamp64\bin\apache\apache2.4.51\bin**openssl.exe**».
2. Установить сертификат и настроить **WAMP**-сервер (**Apache**-сервер).

6.8. *Доступ к серверу из глобальной сети

Коль скоро мы настроили **SSL**-шифрование и наши пароли теперь в безопасности, пришло время задуматься о подключении к серверу через сеть Интернет.

Обозначим основные направления, которые необходимо изучить для достижения этой цели:

- «**Проброс портов**» («перенаправление портов», «переадресация портов») – т. е. настройка роутера на поступление запросов извне (из Интернета на сервер в Локальной сети). По умолчанию используются следующие порты:
  - **HTTP** – порт **80**;
  - **HTTPS** – порт **443**;
  - **MySQL**/MariaDB - порт **3306** (3307, 3308);
  - **FTP** – порт **21** *(а также порт 20 и некоторые из портов динамического диапазона 49152–65535, например, для QNAP NAS диапазон портов пассивного FTP-режима **55536–56559**)*;
- **«Статический IP-адрес»** в глобальной сети *(выдается провайдером)*;
- либо «Динамический DNS» («**DDNS**»), для сервера без статического (глобального) IP-адреса, т. е. с динамическим IP-адресом (хотя DDNS может применяться как к динамическому, так и к статическому IP-адресу, позволяя использовать вместо трудно запоминаемого набора цифр, понятное символьное имя вида «MyName.ddns.net»). *Функция DDNS уже имеется на многих роутерах, остается только зарегистрироваться в одном из сервисов DDNS и правильно настроить роутер.*

!!! Для получения доступа через Интернет вначале должен быть настроен доступ по Локальной сети (см. **раздел 6.6**).

6.9. *Arduino и MySQL

Будем передавать данные с контроллера **Arduino** в базу данных **MySQL**. Для начала создадим саму таблицу при помощи **PhpMyAdmin**, а также «прослойку» на **PHP**, через которую **Arduino** будет связываться с базой данных.

При помощи **PhpMyAdmin** создадим таблицу «**arduino_tbl**», структура которой представлена на рис. 175.

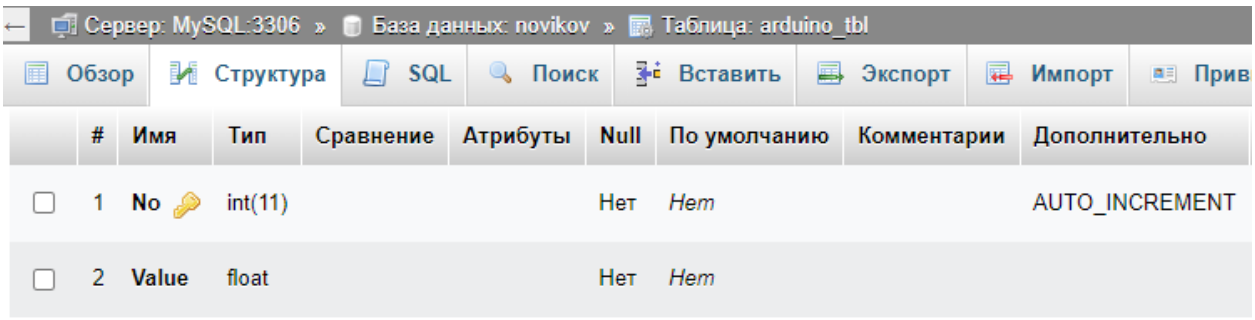


Рис. 175. Создание таблицы

Далее создадим файл «**Arduino.php**», со следующим кодом:

```
<?php
  //Значение, которое нужно добавить в базу
  //(например: localhost/Novikov/Arduino.php?val=7.5 )
  $val = $_GET['val'];


  //Подключиться к базе данных
  $connect = mysqli_connect("localhost", "root", "", "Novikov");
  //Завершить работу в случае ошибки
  if ($connect == false) {
      print("Невозможно подключиться к MySQL");
      exit;
  }
  //Установить кодировку
  mysqli_set_charset($connect, 'utf8');


  //Добавить значение в базу данных
  $sql = "INSERT INTO arduino_tbl VALUES (null, $val);";
  $result = mysqli_query($connect, $sql);
  //Завершить работу в случае ошибки
  if ($result == false) {
      print("Ошибка при выполнении запроса<BR>$sql");
      exit;
  }

  print("Ok!");


  //Завершить соединение с базой данных
  mysqli_close($connect);
?>
```

Теперь после каждого входа по адресу вида «**localhost/Novikov/Arduino.php?val=7**» в таблицу «**arduino_tbl**» (рис. 176) будет добавляться новое значение, указанное в адресной строке.

!!! В дальнейшем необходимо снабдить таблицу колонкой с метками времени, чтобы знать точные дату и время сохранения каждого значения.

No	Value
1	1
2	2
3	3.3
4	9
5	9.1
6	7

Рис. 176. Таблица arduino_tbl

В случае удачной записи значения, в браузере отобразится «**Ok!**» (рис. 177).

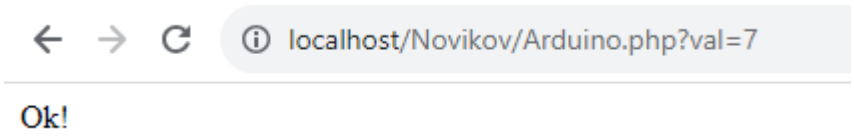


Рис. 177. Успешная запись значения

Далее, научим контроллер самостоятельно заходить по данному адресу. Для этого в **Arduino IDE** напишем следующий код, который необходимо загрузить в контроллер **Arduino**:

```
#include <SPI.h>
#include <Ethernet.h>
EthernetClient client;


//Адрес сервера, к которому осуществляется подключение.
char server[] = "192.168.0.132";
String address = "/Novikov/Arduino.php?val=";


//Настройка адреса текущего устройства.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };


void setup() {
  Serial.begin(9600);
  #define LED 8
  pinMode(LED, OUTPUT);

  //Запускаем Ethernet-соединение.
  //IP-адрес получаем по DHCP.
```

```
  Serial.println("Initialize Ethernet with DHCP:");
  if (Ethernet.begin(mac) == 0) {
    Serial.println("  Error DHCP");
    while (true) {delay(1);}
  } else {
    Serial.print("  DHCP assigned IP ");
    Serial.println(Ethernet.localIP());
  }

  delay(1000);
}

void loop() {
  //В случае успешной передачи, светодиод мигает.
  //В случае ошибки, светодиод горит постоянно.
  digitalWrite(LED, HIGH);

  //Подключение к серверу (порт 80).
  Serial.print("Connecting to ");
  Serial.print(server);
  Serial.println("...  ");
  if (!client.connect(server, 80)) {
    //Ошибка подключения к серверу.
    Serial.println("  Error");
  }

  //В случае успешного подключения...
  if (client.connected()) {
    //Считываем текущее значение с аналогового датчика A0.
    int val = analogRead(A0);
    Serial.print("  val = ");
    Serial.println(val);

    //Выполняем HTTP-запрос.
    client.print("GET ");
    client.print(address);
    client.println(val);
    client.println();

    //Отключиться от сервера.
    client.stop();
    digitalWrite(LED, LOW);
  }

  delay(500);
}
```

Данный код каждые **0,5 сек**. замеряет показания датчика, подключенного к **A0**, и передает это значение при помощи **HTTP**-запроса. Далее программа «**Arduino.php**» помещает данное значение в базу данных.

Задействованное в проекте «железо»:

1. Контроллер Arduino UNO.
2. Модуль Ethernet Shield.
3. Светодиод (LED), подключенный к D8.
4. Аналоговый датчик, подключенный к A0 (измеряемый параметр может быть любым).

Контроллер подключается кабелем к роутеру и после включения питания получает свой **IP**-адрес по **DHCP**.

!!! Контроллер нужно подключать именно к роутеру, т. к. при прямом соединении с ПК он не сможет получить адрес по **DHCP**.

!!! Нужно не забывать отключать Брандмауэр перед подключением контроллера к базе данных.

Информация, отображаемая в «Мониторе порта» **Arduino IDE** во время работы контроллера, представлена на рис. 178.

PhpMyAdmin позволяет отображать данные не только в виде таблиц, но и в виде графика. Для этого необходимо нажать соответствующую кнопку (рис. 179).

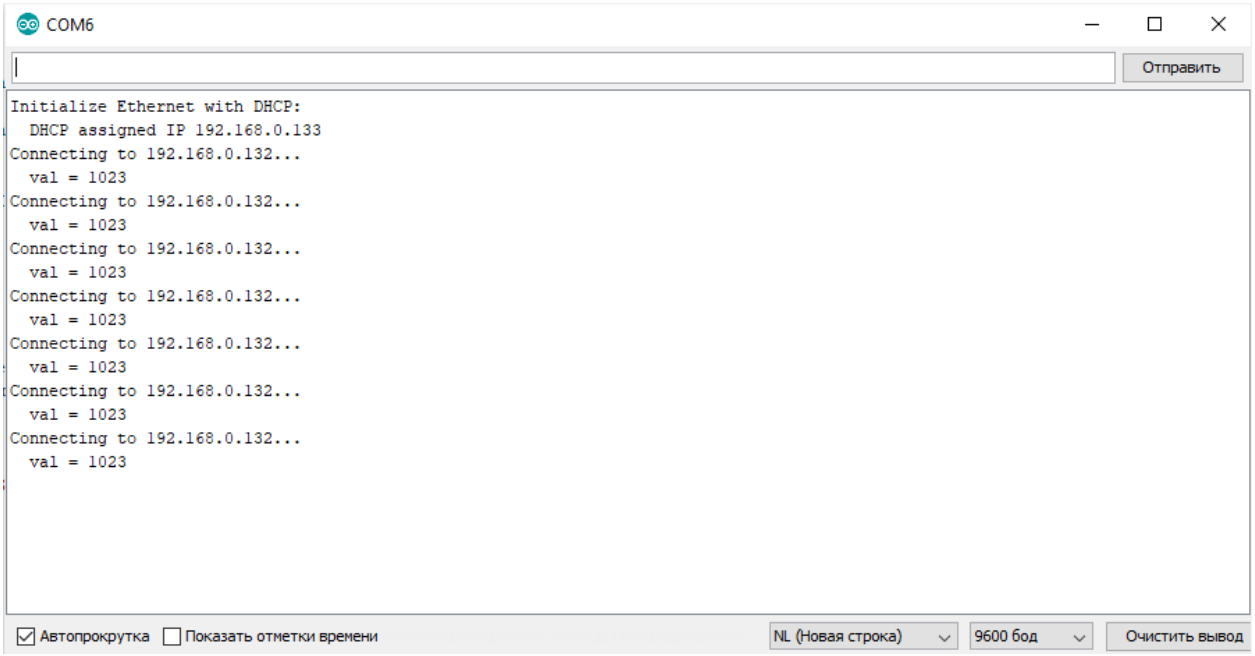


Рис. 178. Монитор порта

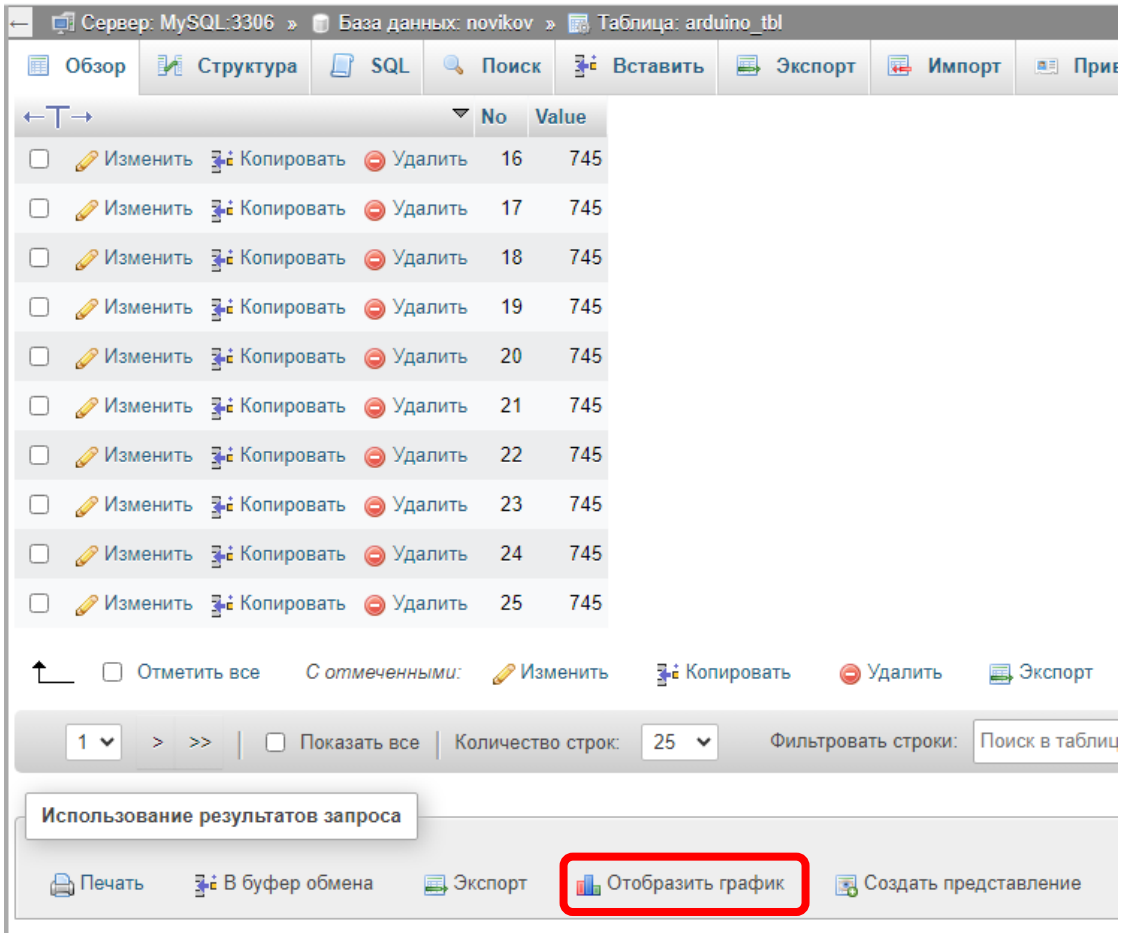


Рис. 179. Построение графика

Пример графика, построенного по данным собранным контроллером, представлен на рис. 180.

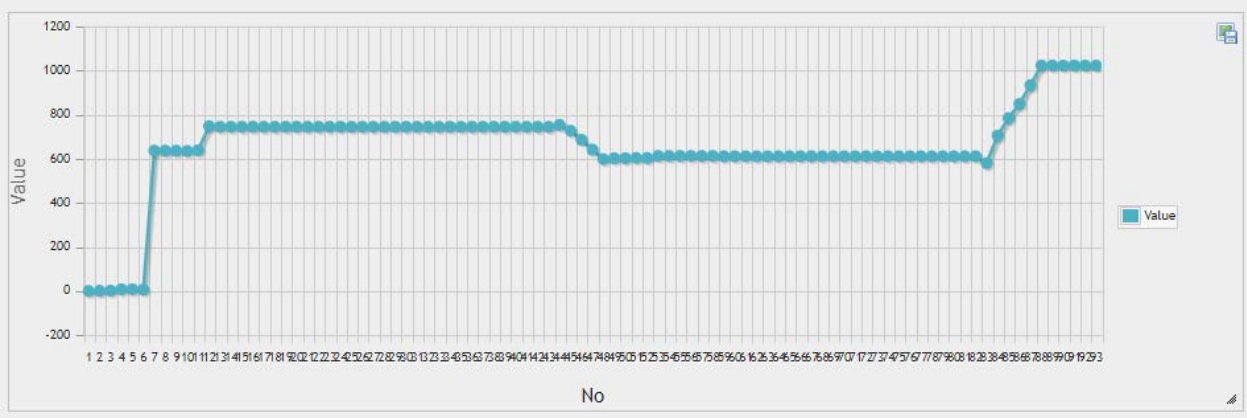


Рис. 180. График в PhpMyAdmin

Естественно, что нет смысла только сохранять значения в базу и нигде их не отображать. Поэтому желательно просматривать данные не только при помощи **PhpMyAdmin**, и в дальнейшем имеет смысл создать отдельную **PHP**-страницу, которая отображает последнее (текущее) значение, а также позволяет просмотреть архив значений.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните функцию перечисленных ключевых слов SQL, приведите пример использования (один пример может пояснять работу сразу нескольких ключевых слов).
    1. CREATE
    2. VALUES
    3. AS
    4. ASC
    5. SELECT
    6. DROP
    7. ORDER BY
    8. INSERT
    9. INNER
    10. PRIMARY KEY
    11. LIKE
    12. JOIN
    13. DELETE
    14. VARCHAR
    15. UPDATE
    16. NOT NULL
    17. SET
    18. OUTER
2. Какие из перечисленных выше ключевых слов являются первым словом в запросе? Какие являются первым словом в строке?
3. Расшифруйте и поясните каждый из перечисленных терминов. Если термины имеют сходное значение или написание, объясните их отличая.
    1. СУБД
    2. БД
    3. Access
    4. Apache
    5. SQL
    6. MySQL
    7. WAMP
    8. DB
    9. MariaDB
    10. HTML
    11. PHP
    12. PhpMyAdmin
    13. HTTP-клиент
    14. HTTP-сервер
    15. Web-сервер
    16. WAMP-сервер
    17. SQL-сервер
    18. Apache-сервер

!!! Приведенный список терминов является минимальным обязательным для студента. Необходимо знать все из перечисленных терминов! *Для смягчения условий оценки студентов преподаватель может разрешить студенту допустить одну ошибку, но не более.*

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Система управления базами данных [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Система_управления_базами_данных (дата обращения: 05.09.2023).
2. Реляционная СУБД [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Реляционная_СУБД (дата обращения: 16.09.2023).
3. Microsoft Access [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Microsoft_Access (дата обращения: 05.09.2023).
4. MySQL [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/MySQL (дата обращения: 05.09.2023).
5. MariaDB [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/MariaDB (дата обращения: 05.09.2023).
6. PhpMyAdmin [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/PhpMyAdmin (дата обращения: 05.09.2023).
7. Динамический сайт [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Динамический_сайт (дата обращения: 05.09.2023).
8. PHP [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/PHP (дата обращения: 05.09.2023).
9. WampServer [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/WampServer (дата обращения: 05.09.2023).
10. WampServer [Электронный ресурс]: Официальный сайт. – URL: https://www.wampserver.com/ (дата обращения: 05.09.2023).
11. Notepad++ [Электронный ресурс]: Официальный сайт. – URL: https://notepad-plus-plus.org/ (дата обращения: 05.09.2023).
12. SQL INNER JOIN Keyword [Электронный ресурс]: W3Schools Online Web Tutorials. – URL: https://www.w3schools.com/sql/sql_join_inner.asp (дата обращения: 05.09.2023).
13. SQL LEFT JOIN Keyword [Электронный ресурс]: W3Schools Online Web Tutorials. – URL: https://www.w3schools.com/sql/sql_join_left.asp (дата обращения: 05.09.2023).
14. SQL RIGHT JOIN Keyword [Электронный ресурс]: W3Schools Online Web Tutorials. – URL: https://www.w3schools.com/sql/sql_join_right.asp (дата обращения: 05.09.2023).
15. SQL Server COALESCE() Function [Электронный ресурс]: W3Schools Online Web Tutorials. – URL: https://www.w3schools.com/sql/func_sqlserver_coalesce.asp (дата обращения: 05.09.2023).
16. SSL [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/SSL (дата обращения: 05.09.2023).
17. TLS [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/TLS (дата обращения: 05.09.2023).
18. HTTPS [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/HTTPS (дата обращения: 05.09.2023).

Учебное издание

Новиков Александр Игоревич

Системы управления базами данных в АСУ

Учебно-методическое пособие

Редактор и корректор Е. О. Тарновская
Техн. редактор Е. О. Тарновская Темплан 2023 г., поз. 5266

Подписано к печати 20.11.23. Формат 60x84/16. Бумага тип № 1.
Печать офсетная. Печ.л. 8,7. Уч.-изд. л. 8,7.
Тираж 30 экз. Изд. № 5266. Цена «С». Заказ №

Ризограф Высшей школы технологии и энергетики СПбГУПТД,
198095, Санкт-Петербург, ул. Ивана Черных, 4.